Transformación de ontologías OWL a UML para la indexación y recuperación mediante esquema basado en grafos.

Universidad Carlos III de Madrid

PROYECTO FIN DE CARRERA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Autor: Javier Ortega Sánchez

Tutor: Anabel Fraga Vázquez

Agradecimientos

A mi madre:

Gracias por mostrarme el camino del amor.

Resumen del proyecto

Título:	Transformación de ontologías OWL a UML para la indexación y recuperación mediante esquema basado en grafos.
Alumno:	Javier Ortega Sánchez
Tutor:	Anabel Fraga Vázquez
Descripción:	El objetivo de este proyecto es conseguir transformar la información disponible en ontologías OWL en modelos UML para la recuperación e indexación mediante esuqemas basados en grafos. Para ello, realizaremos una comparación detallada OWL-UML de forma teórica, donde se establecerán las reglas y operaciones necesarias para realizar la equivalencia entre OWL y UML, desarrollando a continuación el sistema que realice esta tarea. Esta comparación OWL-UML supondrá el grueso del proyecto, realizándose un estudio sobre trabajos similares y publicaciones existentes al respecto, y una vez establecida la transformación teórica más apropiada a nuestras necesidades. Esta aplicación estará integrada en un proyecto más grande, que es UMLModels, donde además se podrá realizar la transformación a otros lenguajes como RSHP a través de librerías previamente desarrolladas.
Summary:	The main purpose of this project is to develop a new application in order to transform OWL ontologies to UML models (based on XMI language). In order to achieve this goal, we need to present a comparison between OWL (Web Ontology Language) and UML model, as a necessary step to design the rules for the transformation. In addition, on this project we will be focused on the comparison of OWL – UML , by analyzing papers, academic publications and documentation about this topic. After that, we will decide which transformation best fits our transformation system and we will develop an application that transforms OWL ontologies into UML. Furthermore, its application is included in a bigger project called UMLModels, that allows transformations to other representation models such as RSHP (Relationship Language).

Contenido

1 Introducción 12
1.1 Visión general 13
1.2 Objetivos 13
1.3 Estructura y contenidos del documento 14
2 Estado del arte 16
2.1 La World Wide Web 17
2.2 La Web Semántica 23
2.2.1 Definición de la Web Semántica 23
2.3 Ontologías 30
2.3.1 Introducción 30
2.3.2 Características de las ontologías 31
2.3.3 Modelado de ontologías 32
2.3.4 Lenguajes utilizados para la representación de ontologías 34
2.3.5 Razonamiento en ontologías 35
2.4 Lenguajes para la representación de ontologías Web 35
2.4.1 Introducción 35
2.4.2 Capas de lenguajes en la Web Semántica 36
2.4.3 RDF (Resource Description Framework) 37
2.4.4 RDFS (Resource Description Framework Schema) 40
2.4.5 OWL, (Web Ontology Language) 41
2.5 Herramientas para representación y transformación de ontologías en la actualidad 54
2.5.1 Características a considerar sobre las aplicaciones del mercado 56
2.5.2 Descripción de herramientas similares 58
2.6 Unified Modeling Language, UML 61
2.7 RSHP (Relationship Language) 66
2.8 UMLModels 71
3 Estudio de viabilidad del sistema 73
3.1 Identificación del alcance del sistema 74
3.2 Estudio de alternativas de solución 74
3.3 Análisis de la transformación de elementos OWL a UML 77
4 Análisis 105
3.1 Introducción 106
3.2 Definición del sistema 107

3.2.2 Identificación del entorno tecnológico 107
3.3 Establecimiento de requisitos 110
3.3.1 Obtención de requisitos 113
3.3.1.1 Requisitos funcionales 113
3.3 3.2.1.2 Requisitos de interfaz 128
3.4 3.2.1.3 Requisitos de comprobación 129
3.4.1 Especificación de casos de uso 129
3.4.2 Descripción de la interacción de objetos 148
3.4.3 Análisis de clases 150
3.4.4 Identificación y especificación de clases 150
3.4.5 Identificación de las clases asociadas a un caso de uso 151
3.5 Definición de Interfaces de Usuario 153
3.5.1 Definición de Interfaces de Usuario 153
3.5.2 Se describe a continuación la interfaz de usuario utilizada. Siendo coherentes con el resto de aplicaciones utilizadas dentro de UML Models. 153
3.5.3 Especificación de Principios Generales de la Interfaz 153
3.5.4 Identificación de Perfiles y Diálogos 153
3.6 Especificación de Formatos Individuales de la Interfaz de Pantalla 154
3.7 Especificación del Comportamiento Dinámico de la Interfaz 154
3.8 Especificación de Formatos de Impresión 155
3.9 Definición del Alcance de las Pruebas 156
3.10 Definición de Requisitos del entorno de Pruebas 156
3.11 Definición de pruebas de aceptación del sistema 157
4 Diseño 163
4.1 Introducción 164
4.2 Definición de arquitectura del sistema 164
4.2.1 Definiciones de niveles de arquitectura 164
4.2.2 Especificación de excepciones 165
4.2.3 Especificación de estándares y normas de diseño y construcción 166
4.3 Diseño de la arquitectura de diseño 166
4.3.1 Diseño de subsistemas de diseño 166
4.4 Diseño de clases 168
4.4.1 Identificación de clases 169
4.4.2 Especificación de clases 170
4.5 Especificación técnica del plan de pruebas 179
4.5.1 Especificación del entorno de pruebas 179

4.5.2 Especificación técnica de niveles de prueba 179
5 Planificación 189
6 Presupuesto 197
7 Resultados 201
7.1 Transformación 1: Jerarquías y atributos 203
7.2 Relaciones y cardinalidad 204
7.3 Herencia en propiedades, propiedad funcional, intersección y restricciones de valor 206
7.4 Elementos UnionOf 208
7.5 Transformación del elemento InverseOf 209
7.6 Enumeración de instancias/individuals 211
7.7 Intersección de multiples clases, herencia múltiple 213
7.8 SameAs, DifferentFrom: Relaciones entre instancias 214
8 Conclusiones y trabajos futuros 216
8.1 Conclusiones 217
8.1.1 Resultados de la planificación 217
8.1.2 Evaluación del trabajo realizado 218
8.1.3 Trabajos futuros 219
9 Anexo 220
9.1 Glosario y definiciones 221
9.2 Referencias 223

Indice de ilustraciones

Ilustración 1. Diagrama original de arquitectura pra la World Wide Web [6] 17
Ilustración 2. Diagrama intercambio de archivos en la WWW 19
Ilustración 4. Diagrama Venn de las categorías de esquemas URI [4] 20
Ilustración 3. Funcionamiento básico en la WWW 20
Ilustración 5. Identificación y representación de recursos en RDF 21
Ilustración 6. Esquema para la Web 1.0 22
Ilustración 7. Esquema para la Web 2.0 23
Ilustración 8. Evolución de la Web. Tecnologías y aplicaciones [54] 25
Ilustración 9. Arquitectura de capas en la Web Semántica [46] 36
Ilustración 10. Elementos básicos en sentencias RDF 38
Ilustración 11. Esquema de utilización de recursos en RDF 39
Ilustración 12. Lenguajes OWL 42
Ilustración 13. Esquema sobre espacios de nombres e importación de ontologías [46] 43
Ilustración 14. Estructura básica de ontologías OWL 45
Ilustración 15. Jerarquía en UML 48
Ilustración 16. Representación conceptual de clases en OWL 48

Ilustración 17. Jerarquía de propiedades en OWL 49
Ilustración 18. Esquema conceptual de propiedades en OWL 50
Ilustración 19. Herramientas de transformación OWL similares 56
Ilustración 20. Captura de pantalla OWL2UML en Protege 58
Ilustración 21. Captura de pantalla de visualización de ontologías en OWLGrEd 60
Ilustración 22. Esquema del sistema ATL Eclipse 60
Ilustración 23. Esquema de funcionamiento de la aplicación de Xu et al 61
Ilustración 24. Artefacto RSHP 67
Ilustración 25. Representación por términos 67
Ilustración 26. Termino en RSHP 68
Ilustración 27. estructura de relaciones en RSHP 68
Ilustración 28. Estructura de propiedades en RSHP 70
Ilustración 29. Modelo de representación RSHP [66] 71
Ilustración 30. Esquema conceptual de la aplicación a diseñar 104
Ilustración 31. Tareas de la fase de Análisis 106
Ilustración 32. Diagrama general de casos de uso 130
Ilustración 33. Diagrama de casos de uso: OWL classes and properties transformation 131
Ilustración 34. Diagrama de casos de uso: OWL Property characteristics 131
Ilustración 35.Diagrama de casos de uso: Individuals 132
Ilustración 36. Diagrama de secuencia, Recuperar una clase 149
Ilustración 37. Diagrama de actividad, Recuperar una actividad 149
Ilustración 38. Diagrama de actividad recuperar una descripción 150
Ilustración 39. Mensaje de resumen 154
Ilustración 40. Mensaje de error 154
Ilustración 41. Formulario principal 155
Ilustración 42. Representación de capas 165
Ilustración 43. Diseño de componentes del sistema 167
Ilustración 44. Diagrama de clases 168
Ilustración 45. Muestra de pantalla de pruebas 180
Ilustración 46. Diagrama de Gantt, Planificación 196
Ilustración 47. Presupuesto por roles 198
Ilustración 48. Distribución de horas en roles y actividades 198
Ilustración 49. Presupuesto para materiales y licencias 199
Ilustración 50. Resumen presupuesto 200
Ilustración 51. Resultado transformaciónOWL2UML 215
Ilustración 52. Debrief tiempos real 218

Indice de tablas

Tabla 1. Recursos en FDF 39
Tabla 2. Algunos constructores en RDF 41
Tabla 3. Espacio de nombres en cabeceras OWL 46
Tabla 4. Elementos de información en ontologías OWL 47
Tabla 5. Comparativa de herramientas de transformación OWL 58
Tabla 6. Elemento clase en UML 62
Tabla 7. Elemento generalización en UML 63
Tabla 8. Composición en UML 63

Tabla 9. Agregación en UML 64
Tabla 10. Dependencia en UML 64
Tabla 11. Asociación en UML 65
Tabla 12. Escala de valores a utilizar en evaluación 75
Tabla 13. Criterios de evaluación 75
Tabla 14. Resultados de la evaluación 76
Tabla 15. Coste de herramientas a utilizar 77
Tabla 16. Ejemplo tabla transformación 80
Tabla 17. Transformación owl:Class 81
Tabla 18. Transformación rdfs:subClassOf 82
Tabla 19. Transformación owl:equivalentClass 83
Tabla 20. Transformación owl:disjointWith 84
Tabla 21. Transformación owl:oneOf 85
Tabla 22. Transformación owl:intersectionOf 86
Tabla 23. Transformación owl:unionOf 87
Tabla 24. Transformación owl:complemntOf 88
Tabla 25. Transofrmación owl:hasValue 89
Tabla 26. Transformación owl:allValuesFrom 90
Tabla 27. Transformación owl:someValuesFrom 91
Tabla 28. Transformación owl:maxCardinality 92
Tabla 29. Transformación owl:minCardinality 93
Tabla 30. Transformación owl:cardinality 94
Tabla 31. Transformación owl:ObjectProperty 95
Tabla 32. Transformación owl:DatatypeProperty 96
Tabla 33. Transformación owl:inverseOf 97
Tabla 34. Transformación owl:subPropertyOf 98
Tabla 35. Transformación owl:equivalentProperty 99
Tabla 36. Transformación owl:FunctionalProperty 100
Tabla 37. Transformación owl:inverseFunctionalProperty 101
Tabla 38. Transformación owl:TransitiveProperty 102
Tabla 39. Transformaación owl:SymmetricProperty 103
Tabla 40. RSF-000, modelo de tabla de requisitos 113
Tabla 41. RSF-001 Precesamiento de la ontología 114
Tabla 42. RSIS-001 Recuperación de clases OWL 114
Tabla 43.RSIS-002 Recuperación de jerarquías en OWL 115
Tabla 44. RSIS-003 Recuperación de clases equivalentes 115
Tabla 45. RSIS-004 Recuperación de clases disjuntas 116
Tabla 46. RSIS-005 Recuperación de elementos de enumeración (owl:oneOf) 116
Tabla 47. RSIS-006 Recuperación de elementos de tipo intersección (owl:intersectionOf) 117
Tabla 48. Recuperación de elementos tipo unión (owl:unionOf) 117
Tabla 49. RSIS-008 Recuperación de elementos de tipo restricción (owl:allValuesFrom) 118
Tabla 50. RSIS-009 Recuperación de elementos de tipo restricción (owl:hasValue) 118
Tabla 51. RSIS-010 Recuperación de cardinalidad máxima (owl:maxCardinality) 119
Tabla 52. RSIS-011 Recuperación de cardinalidad mínima (owl:minCardinality) 119
Tabla 53. RSIS-012 Recuperación de cardinalidad exacta (owl:cardinality) 120
Tabla 54. RSIS-013 Recuperación de propiedades de relación (owl:OjectProperty) 120

Tabla 55. RSIS-014 Recuperación de propiedades de relación (owl:DatatypeProperty) 121
Tabla 56. RSIS-015 Recuperación de propiedades de especialización (rdfs:subPropertyOf) 121
Tabla 57. RSIS-016 Recuperación de propiedades de relación inversa (owl:inverseOf) 122
Tabla 58. RSIS-017 Recuperación de propiedades funcionales (owl:FunctionalProperty) 122
Tabla 59. RSIS-018 Recuperación de propiedades simétricas (owl:symmetricProperty) 123
Tabla 60. RSIS-019 Recuperación de instancias 123
Tabla 61. RSIS-020 Verificación del archivo OWL..... 124
Tabla 62. RSIS-021 Inserción en modelo UML..... 124
Tabla 63. RSIS-022 Inserción en modelo RSHP 125
Tabla 64. RSIS-023 Recuperación de atributos..... 125
Tabla 65. RSIS-024 Recuperación de instancias mediante declaración de clase..... 126
Tabla 66. RSIS-025 Inserción de comentarios en elementos UML..... 126
Tabla 67. RSIS-026 Recuperación de elementos tipo owl:sameAs 127
Tabla 68. RSIS-027 Recuperación de elementos tipo owl:differentFrom 127
Tabla 69. RIN-001 Utilización de librerías CAKE para transformación..... 128
Tabla 70. RIN-002 Utilización de librerías nUML para inserción en modelo 129
Tabla 71. RCOM-001 Verificación de ontologías..... 129
Tabla 72.CU-001 Recuperar clase 132
Tabla 73. CU-002 Obtener jerarquía 133
Tabla 74. CU-003 Obtener clases equivalentes 134
Tabla 75. CU-004 Obtener clases disjuntas 135
Tabla 76. CU-005 Recuperar enumeración 136
Tabla 77.CU-006 Recuperar propiedad intersección..... 136
Tabla 78. CU-007 Recuperar restricción allValuesFrom..... 137
Tabla 79. CU-008 Recuperar cardinalidad mínima..... 138
Tabla 80. CU-009 recuperar cardinalidad exacta 139
Tabla 81. CU-010 recuperar propiedad unión..... 140
Tabla 82. CU-011 Recuperar instancia o valor 141
Tabla 83. CU-012 Recuperar owl:ObjectProperty..... 142
Tabla 84. CU-013 Recuperar owl:DatatypeProperty..... 143
Tabla 85. CU-014 Recuperar owl:subPropertyOf 144
Tabla 86. CU-015 Recuperar owl:inverseOf 145
Tabla 87. CU-016 Recuperar owl:FunctionalProperty..... 146
Tabla 88. CU-017 Recuperar owl:sameAs 147
Tabla 89. CU-018 Recuperar owl:differentFrom 148
Tabla 90. Definición de clase 151
Tabla 91. Definición de información de usuario..... 152
Tabla 92. Definición de clase de transformación 152
Tabla 93. PA-001 Verificar el documento ontología..... 157
Tabla 94. PA-002 Recuperar una jerarquía 158
Tabla 95. PA-003 Recuperar una relación ObjectProperty 158
Tabla 96. PA-004 Recuperar un elemento owl:equivalentClass 159
Tabla 97. PA-005 Recuperar un elemento owl:oneOf..... 160
Tabla 98. PA-006 Recuperar un elemento owl:maxCardinality..... 161
Tabla 99. PA-007 Recuperar un elemento owl:maxCardinality..... 161
Tabla 100. PA-008 Recuperar instancias 162

Tabla 101. Descripción de clase UmlElement 170
Tabla 102. Descripción de clase (Modelo UML) 171
Tabla 103. Descripción de clase relación 172
Tabla 104. Descripción de clase generalización 173
Tabla 105. Descripción de clase instancia 173
Tabla 106. Descripción de clase atribut 174
Tabla 107. Descripción de clase enumeración 174
Tabla 108. Descripción de clase ontología 175
Tabla 109. Descripción de clase OWL2UmlTransformer 178
Tabla 110. PU-001 Recuperación de clases 181
Tabla 111. PU-002 Recuperación de owl:objectPeroperty 182
Tabla 112. PU-003 Recuperación de rdfs:subClassOf 183
Tabla 113. PU-004 Recuperación de owl:oneOf 184
Tabla 114. PU-005 Tratamiento de cardinalidad 185
Tabla 115. PU-006 Recuperación de elementos owl:DatatypeProperty 185
Tabla 116. PU-007 Recuperación de instancias 186
Tabla 117. PU-008 Recuperación de elementos owl:inverseOf 187
Tabla 118. Recuperación de relación entre instancias (owl:sameAs; owl:differentFrom) 187
Tabla 119. Recuperación de elementos owl:DatatypeProperty 188
Tabla 120. Resumen de tareas 191
Tabla 121. Horas por mes 191
Tabla 122. Planificación agosto 192
Tabla 123. Planificación septiembre 192
Tabla 124. Planificación octubre 193
Tabla 125. Planificación noviembre 193
Tabla 126. Planificación diciembre 193
Tabla 127. Planificación enero 194
Tabla 128. Planificación febrero 194
Tabla 129. Planificación marzo 195
Tabla 130. Planificación abril 195
Tabla 131. Planificación mayo 195

1 Introducción

En este capítulo se introducirán los temas principales a tratar en este proyecto, así como la descripción a alto nivel en cuanto a los objetivos a conseguir. Se incluirá también una definición de la estructura de este documento a fin de orientar en la lectura y utilización de este documento.

1.1 Visión general

La gestión, el tratamiento y utilización del conocimiento se han convertido en actividades de gran relevancia, no sólo dentro del ámbito profesional y académico en la ingeniería informática, sino también en las actividades cotidianas de la sociedad de la información.

La gran expansión y crecimiento de la Web (WWW), así como de los sistemas y tecnologías utilizados por la sociedad para realizar multitud de tareas o gestiones que llegan a considerarse de mucha utilidad, nos hacen plantearnos nuevas herramientas o posibilidades para entregar mejores productos o funcionalidades a los usuarios. El crecimiento de la cantidad de información disponible en la Web, así como los distintos usos, han crecido enormemente con respecto a lo inicialmente esperado en su creación, y se hace indispensable automatizar la gestión y utilización de la información que se encuentra disponible en la Web.

A fin de poder conseguir estos objetivos, se han ido planteando cambios en la evolución de la Web, ya que en un principio partíamos de un gran repositorio de documentos enlazados, por decirlo de alguna manera, y esto no permite cumplir con éxito las expectativas de los productos diseñados hasta cierto momento. Este hecho, vinculado al crecimiento de la Web y a la mayor exigencia de los usuarios en la utilización de las funcionalidades, motivaron dotar a la Web de mayor semántica, y por tanto mayor significado, además de añadir una estructura o formato que pueda ser legible por las máquinas.

Para conseguir esto, surgieron nuevos recursos en la Web, apoyados en disciplinas como la inteligencia artificial, y en concreto la representación del conocimiento mediante ontologías. El lenguaje más importante y representativo en este ámbito (apoyado o construido a partir de otros anteriores) es Ontology Web Language (OWL).

1.2 Objetivos

Este proyecto forma parte de una aplicación más grande y compleja, que es UMLModels. Esta aplicación, explicando su funcionamiento a alto nivel, consta de un crawler que realiza búsquedas en la Web, extrayendo información en los formatos que más tarde recuperarán en modelos UML. En esta labor de indexación, trabajan aplicaciones que se han ido incluyendo como módulos que han ido completando la aplicación UMLModels, para recuperar información desde distintos formatos, como imágenes, lenguaje XML y bases de datos Access.

El objetivo de este proyecto, es *incluir un módulo más dentro de UMLModels, que permita la recuperación en UML de ontologías OWL* que puedan ser recuperadas desde la Web.

Aunque los resultados que entregará UMLModels, mediante la utilización del nuevo módulo OWL2UML serán modelos UML (archivo XMI), imágenes mediante el uso de otra aplicación y modelos RSHP, el objetivo fundamental de este proyecto, *es establecer y diseñar las reglas necesarias para extraer la información disponible en las ontologías OWL transformando esta información en un modelo intermedio normalizado, que pueda ser representado en UML*.

Una vez se hayan diseñado las reglas y operaciones necesarias para recuperar la información de ontologías OWL, la creación de los ficheros y modelos XMI y RSHP se realizará mediante otras librerías ya creadas previamente a este proyecto.

1.3 Estructura y contenidos del documento

En este documento encontraremos distintos apartados que nos sirven como soporte, ayuda y herramienta para el diseño del sistema a desarrollar en este proyecto.

- **Introducción**: En este apartado se realiza una introducción sobre el tema a tratar, así como los objetivos a alto nivel que se tratarán en el proyecto. No se entrará en detalle en este apartado, pero ayudará a entender a nivel conceptual cuales son los elementos y objetivos fundamentales.

- **Estado del arte**: En este apartado, se describen las tecnologías, campos de estudio y recursos necesarios para llevara a cabo el proyecto que se lleva a cabo. Es necesario para completar este apartado realizar tareas de investigación y análisis sobre distintos aspectos, a fin de comprender a todos los elementos involucrados, así como las posibles soluciones que se puedan plantear en otras fases.

- **Análisis**: En este apartado se establecen los requisitos y funcionalidades de la aplicación. Se da una primera aproximación a las características del sistema, siendo desarrolladas estas más en detalle en el apartado de Diseño.

- **Diseño**: Partiendo de los requisitos recogidos en el apartado anterior, se trabajará en este apartado en el diseño de la aplicación teniendo en cuenta

la arquitectura de la misma, así como las indicaciones a alto nivel sobre su construcción y pruebas.

- **Planificación y presupuesto**: En este apartado se detalla la planificación inicialmente creada para el proyecto, así como la propuesta económica para el sistema y proyecto a desarrollar.

- **Conclusiones y resultados**: En estos apartados se incluirán los puntos más importantes a destacar como conclusiones, partiendo de los resultados y trabajo realizado. Se mostrarán los resultados obtenidos durante la realización del proyecto.

2 Estado del arte

En este capítulo se realiza un resumen y exposición de la información encontrada en las tareas de investigación en este proyecto. Esto servirá como base para la comprensión de todos los elementos y tecnologías involucrados en el trabajo, siendo imprescindible para la realización del resto de etapas del proyecto. Se investigan y expone información sobre la Web, la Web Semántica, ontologías, lenguaje de modelado UML y lenguaje basado en relaciones RSHP.

2.1 La World Wide Web

Antes de comenzar a explicar otros conceptos y elementos necesarios para desarrollar este proyecto, debemos ver detenidamente qué es la web, y su evolución hacia la web semántica.

La Web ha provocado un gran cambio en nuestra sociedad (acceso a la información, gestión de la información, entretenimiento…) [6], y en la forma de comunicarnos, pero a la par hemos visto cómo la cantidad de información disponible en la misma, así como las utilidades de que disponemos en ella se han incrementado enormemente.

En 1989 Tim Berners-Lee inventó la World Wide Web [1]. Inicialmente, el objetivo principal para la Web era conseguir que la interacción persona – hipertexto fuese intuitiva de tal manera que las máquinas pudiesen leer y representar los pensamientos, interacciones y patrones de utilización de las personas, pudiendo conseguir análisis muy valiosos mediante la utilización de máquinas (software, dispositivos…). Para estos análisis, se tendría en cuenta la similitud de los problemas o necesidades entre personas de distintas organizaciones [6].

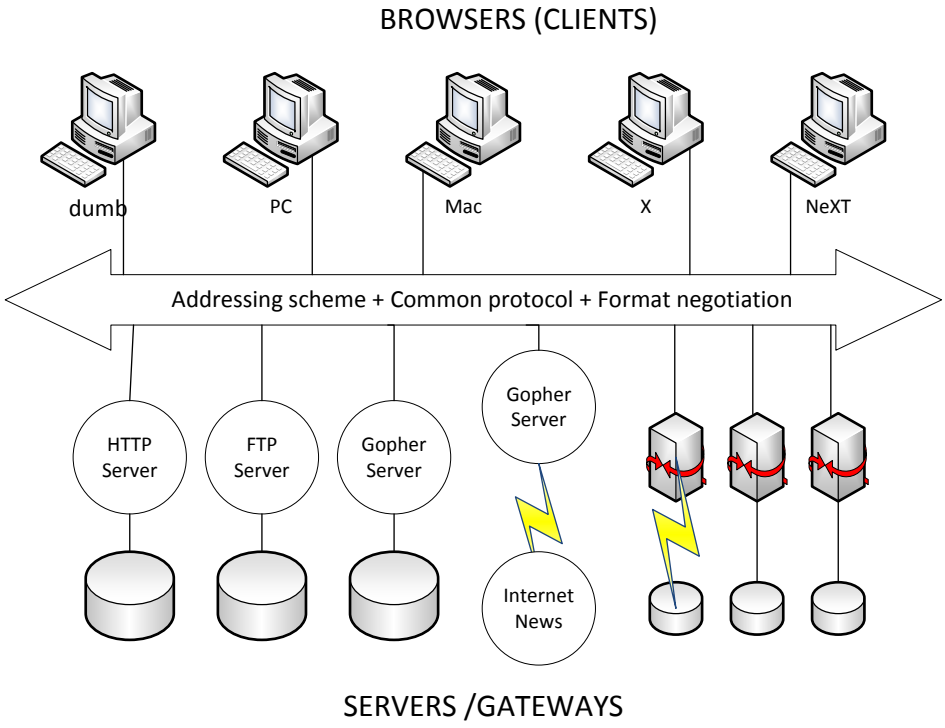


Ilustración 1. Diagrama original de arquitectura pra la World Wide Web [6]

Además, la web se diseñó de acuerdo a los siguientes principios [6] según su creador Tim Berners-Lee:

- **Independencia de especificaciones**: Uno de los elementos más importantes fue la flexibilidad. Cada nueva especificación que quisiese ser añadida necesitaba asegurar su compatibilidad con el resto. Estas nuevas especificaciones, debían cumplir con una estricta modularidad e independencia de unas respecto de otras, con el fin de poder modificarlas o eliminarlas en el futuro sin causar problemas. En estos primeros pasos, se debía demostrar que la Web podía crecer, incluyendo nuevas especificaciones manteniéndose estable y con un crecimiento sostenido.

- **Universal Resource Identifiers (URI):** Un enlace podría apuntar a cualquier documento, o dicho de forma más genérica, a un recurso. Esto requería un espacio global de de identificadores, que son uno de los elementos más importantes de la World Wide Web. Una URI es universal, teniendo en cuenta que podremos representar cualquier nuevo espacio en la web (recurso), de acuerdo a su identificación, nombre o dirección. Las propiedades de la URI dependerán de las características del espacio al que apunta esta, pudiendo ser "nombre" o "localización". Además, esta URI podría ir cambiando de acuerdo a los protocolos utilizados y sintaxis.

- **Opacidad de los identificadores:** Una cuestión importante es que las URI's son tratadas como "strings opacos", en los que el software cliente no podría ver el contenido internamente o realizar cambios en los recursos solicitados.

- **URI's genéricas:** Las URI's podrían representar objetos de forma "genérica". Esto es que podríamos tener una URI distinta para representar un documento en un determinado idioma, otra URI para un idioma distinto, y otra URI para acceder a un streaming.

- **HTTP (Hypertext Transfer Protocol):** Los protocolos de comunicación, siendo utilizados para acceder a datos de forma remota, ofrecían en un comienzo el protocolo FTP, pero este se descartó al poco tiempo, considerándolo lento para la transferencia de hipertexto, siendo poco óptimo para la Web, por lo que se diseñó el HiperText Transfer Protocol (HTTP). Utilizando HTTP las URI's se resolverían en el propio documento, haciéndose en dos mitades. La primera mitad sería utilizada para saber el nombre del servidor al que se hace la petición (Domain Name Server, DNS), y la segunda sería un string de forma "opaca" que sería gestionado por el servidor. Una de las utilidades de HHTP, sería entregar un documento/recurso al cliente acuerdo a criterios de lenguaje o formato. Por ejemplo en el caso de URI's genéricas, permitiría al servidor seleccionar el

recurso más apropiado. Uno de los elementos importantes sería la negociación del formato (el cliente indica necesidades/requisitos, y el servidor intenta ajustarse a ello), haciendo hincapié en la independencia entre HTTP y HTML. Esta negociación de acuerdo a HTTP/1.1, podría ser gestionada por el servidor, entregando al cliente el recurso "suponiendo" que los parámetros serían correctos (idioma, etc…), o gestionado por el cliente, siendo el propio navegador quien conozca de alguna manera cuales son los parámetros, o siendo el usuario quien inserte esta información en el navegador manualmente [6] [8].

- **HTML (Hypertext Markup Language):** Para el intercambio de hipertexto (herramienta software que permite crear, enlazar, agregar y compartir información de varias fuentes mediante enlaces asociativos [9]), se eligió HTML (Hypertext Markup Language). Inicialmente se pensó que sería complejo para los usuarios utilizar este nuevo sistema de información, por lo que se adoptó HTML como un lenguaje similar a SGML (Standard Generalized Markup Language), a fin de que los usuarios de esta comunidad y otras pudiesen adaptarse con más facilidad. No se aceptó SGML por ser algo complejo, y no tener bien definida su sintaxis.

Aunque no se va a describir en este trabajo de forma detallada el funcionamiento de la Web, sí que es importante indicarlo a alto nivel:

- Los nombres de servidores que utilizamos en las URI's son resueltos a fin de obtener la dirección IP del servidor que recibe la petición para entregar los paquetes de datos de la página. Estos nombres, son resueltos mediante servidores DNS, que contienen información (tablas) sobre la correspondencia nombre de dominio-IP.

- A continuación se realizará una solicitud HTTP solicitando el recurso correspondiente, como por ejemplo una página web. Se solicitará el código HTML de la página, que será analizado por el navegador a fin de determinar si es necesario seguir solicitando más recursos al servidor.

- Una vez son recibidos los recursos mediante paquetes de datos, el navegador representa la información, de acuerdo al código HTML, hojas de estilo CSS, u otros elementos que determinen la apariencia de la información.

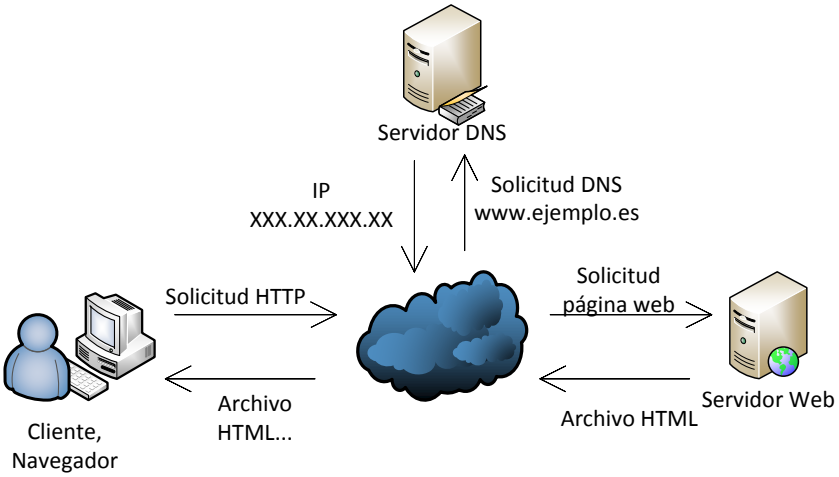


Ilustración 3. Funcionamiento básico en la WWW

Las URI's en la Web:

URI's, que son identificadores universales pudiendo ser URL's (identificador de la localización del recurso) ó URN's (su nombre independientemente de la localización) [2].

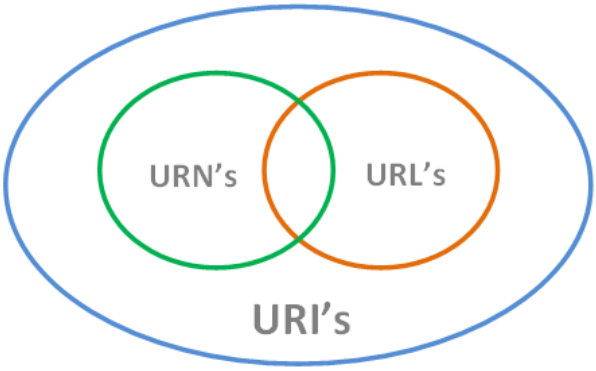


Ilustración 4. Diagrama Venn de las categorías de esquemas URI [4]

Las URN's y URL's son subconjuntos de URI's, y son generalmente disjuntos, pero no se considera así estrictamente ya que todas las URI's pueden tratarse como nombres y podemos encontrar esquemas con las dos categorías o ninguna de ellas [5] [2].

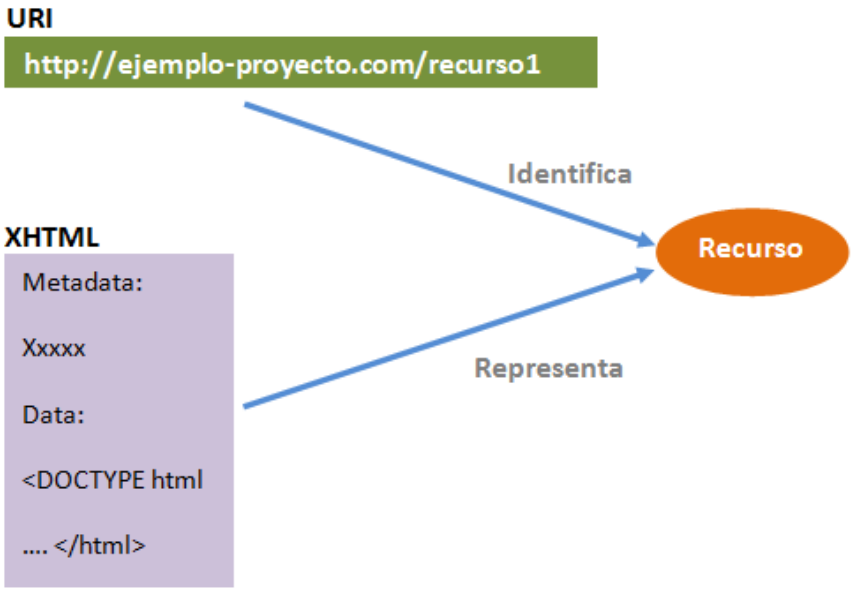


Ilustración 5. Identificación y representación de recursos en RDF

Una URI está compuesta por:

- **Esquema**: especificación para asignar identificadores (urn:, tag:), o indicar el protocolo (http, ftp,…) [5]

- **Autoridad**: elemento jerárquico para identificar la autoridad de nombres (www.webejemplo.com)

- **Ruta**: información generalmente organizada de forma jerárquica para identificar el ámbito del esquema URI y la autoridad de nombres (/webdomain/example)

- **Consulta**: información con estructura no jerárquica (con pares "clave=valor") que identifica el recurso en el ámbito del esquema URI y autoridad de nombres. El comienzo de este elemento es con el carácter "?".

- **Fragmento**: permite identificar una parte del recurso principal o vista de una representación del mismo. Comienza con el carácter "#".

Hemos hablado hasta ahora del nacimiento de la Web, su funcionamiento en términos generales, y de su evolución hasta nuestros días. En los inicios de esta tecnología (los años 90), la Web era estática, es decir, los usuarios utilizaban la Web como medio de consulta, de acceso a la información. En los documentos de hipertexto, se comenzaban a acumular grandes cantidades de información, pero los usuarios no colaboraban o interactuaban en ese contenido.

A medida que la Web iba creciendo, la cantidad de información disponible aumentaba, y los usuarios de la misma también se incrementaban, se daban paso nuevas capacidades y funcionalidades para la interacción con la Web.

Desde el inicio de la Web en los años noventa, era considerada la Web 1.0, caracterizada por ser estática, sin interacción ni contribución de los usuarios. Los documentos, o páginas web son mantenidas y modificadas por un grupo restringido de usuarios, y estos recursos son utilizados o consultados por el resto de usuarios.

Ilustración 6. Esquema para la Web 1.0

Teniendo en cuenta lo anterior, no existe un momento exacto o concreto en que la Web pasó a denominarse Web 2.0, pero según varias fuentes [10] [11] [12], este término empezó a utilizarse en una conferencia de Tim O'Reilly en el año 2004 sobre la Web 2.0, en la que colaboraba en esta discusión Dale Dougherty.

Este nuevo escalón en la Web, estaría apoyado en la capacidad de los usuarios de colaborar para añadir contenido y actuar de forma colaborativa, como por ejemplo mediante el uso de redes sociales, plataformas de almacenamiento de vídeos, foros, gestiones o consultas online, así como plataformas en el mundo empresarial que ofrecen grandes opciones de colaboración a los miembros de una organización.

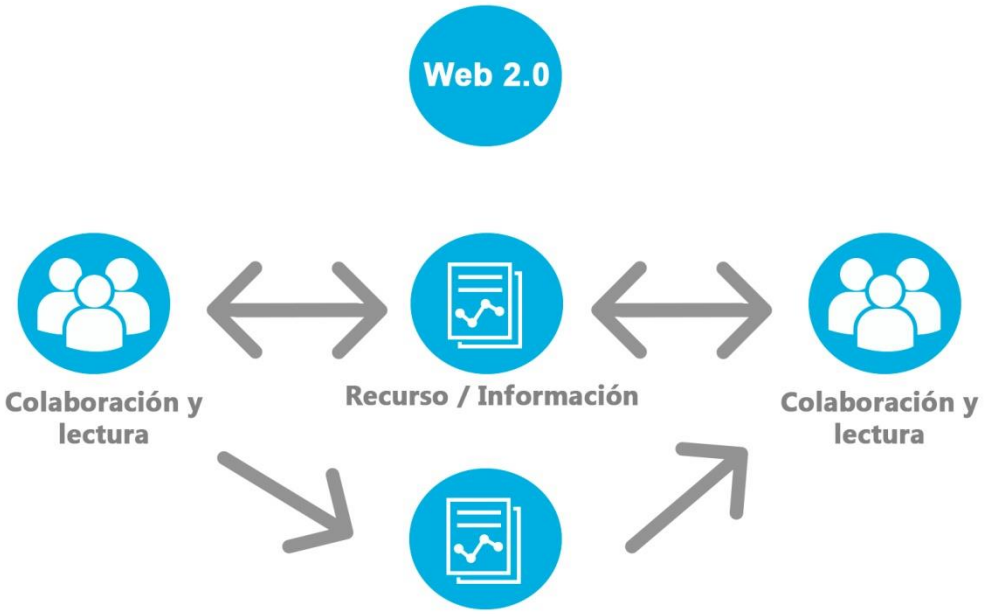


Ilustración 7. Esquema para la Web 2.0

Este cambio de "escalón", o "versión" no quiere decir que las tecnologías o estándares de la Web hayan cambiado en un momento concreto, sino más bien en una "lenta transición" desde la Web estática, hacia la colaborativa y dinámica que hay actualmente.

2.2 La Web Semántica

2.2.1 Definición de la Web Semántica

La Web Semántica es considerada la "web de los datos", y una extensión de la Web actual. En esta nueva extensión de la Web, se incluyen más datos, y por tanto más semántica con el fin de que los agentes inteligentes (programas software que pueden utilizar la información disponible en la Web), realicen un mejor uso de los datos que existen en la Web. La Web que se ha utilizado hasta la época actual, fue diseñada para ser utilizada y leída por los seres humanos [20], pero no por programas o agentes que puedan utilizarla de forma "inteligente" de acuerdo a su significado. Mediante el uso de ordenadores y aplicaciones (navegadores, etc…), se realizan tareas como mostrar los datos con u determinado estilo o efectos, se incluyen enlaces a otros documentos o páginas, etc…pero en la gran mayoría de los casos, los agentes no pueden "entender" o procesar los datos de manera adecuada, de tal forma que pudiesen entregar mejores resultados en actividades cotidianas como realizar búsquedas. Mediante el uso de la Web Semántica, se puede aportar de más significado a la Web, y

gracias a ello las máquinas podrían realizar tareas más complejas "comprendiendo" documentos específicos sobre semántica y datos, pero no habría que confundirlo con documentos de texto enfocados hacia la lectura y comprensión de las personas.

Otro de los objetivos fundamentales, es hacer la Web más universal, es decir, que el significado de las páginas web, independientemente de la cultura, el idioma, o terminología utilizada en un determinado campo, pueda ser utilizada y compartida por los agentes/aplicaciones que realizan tareas de forma automática. Hasta la actualidad la Web ha evolucionado de tal forma, que la mayor parte del desarrollo ha estado dirigido hacia el ámbito de los usuarios, o su interacción y comprensión de las páginas Web, pero no tanto hacia la evolución de la Web para que pueda ser mejor utilizada por las máquinas, y esto es uno de los objetivos fundamentales de la Web Semántica.

La representación del conocimiento, es utilizada en la Web Semántica de tal manera que las máquinas puedan tener acceso a repositorios de información clasificada o estructurada, además de reglas de inferencia, a fin de que se puedan realizar tareas de razonamiento automático [20].

La tendencia en la Web actual, ha sido el uso de conceptos o términos, de tal manera que tuviesen que ser utilizados de forma exacta en distintos documentos (páginas Web), para poder ser procesados por las máquinas, pero debido al crecimiento exponencial de la Web actual esta tarea se convierte en algo tedioso y difícil, sin llegar incluso a obtener los resultados esperados en algunas aplicaciones utilizadas.

Mediante la utilización de lógica (reglas de inferencia) en la Web, se pretenden realizar consultas y tareas, y aunque estos recursos matemáticos y de ingeniería pueden ofrecer buenas soluciones en estos casos, podríamos encontrar problemas a la hora de decidir en casos complejos como pueden ser las paradojas.

Desde un principio, las tecnologías utilizadas para desarrollar la Web Semántica fueron XML (eXtensible Markup Language) y RDF (Resource Description Framework). XML aporta un grado de libertad en la definición de las etiquetas utilizadas (por ejemplo, <Apellido>), y esto resulta problemático a la hora de ser interpretado por las máquinas, ya que habría que consensuar el nombre de los distintos elementos utilizados en las definiciones de este lenguaje para que los programas pudiesen entender las etiquetas para realizar diferentes tareas. Por tanto, tal y como recalca el autor en uno de los primeros documentos en los que se describe la Web Semántica [20], XML ofrece a los usuarios una estructura de etiquetas que pueden ser definidas de manera arbitraria, por lo que no podría ser comprendido correctamente por las máquinas.

RDF nos permite expresar el significado, incluyéndolo en triplas que están compuestas de sujeto, predicado y objeto, pudiendo ser representadas utilizando XML. Estas relaciones objeto-predicado/verbo – objeto son entendidas como afirmaciones sobre distintos elementos de un dominio, que se utilizan para describir los datos que serán procesados por las máquinas. Los sujetos , objetos se identifican como URIs (Uniform Resource Identifier), y los verbos también son representados utilizando el mismo elemento (URI), pudiendo los usuarios aportar nuevos conceptos o verbos en la Web. Las palabras en utilizadas en la Web dejarían de ser simplemente "palabras", para convertirse en elementos relacionados con otros, que aporten más información y recursos a las máquinas que los utilizan.

Para seguir profundizando en la definición de Web Semántica, pasamos a ver la utilización de ontologías. Como se ha comentado anteriormente, uno de los objetivos fundamentales es la universalización de la Web. Uno de los problemas que nos podemos encontrar en el uso de la Web, es que en algunas ocasiones hay palabras o términos que son escritos o representados de forma distinta, aunque conceptualmente representen una misma cosa y, si un programa utiliza estos recursos no podría realizar tareas complejas debido a que no puede "entender" la semejanza entre dichas palabras.

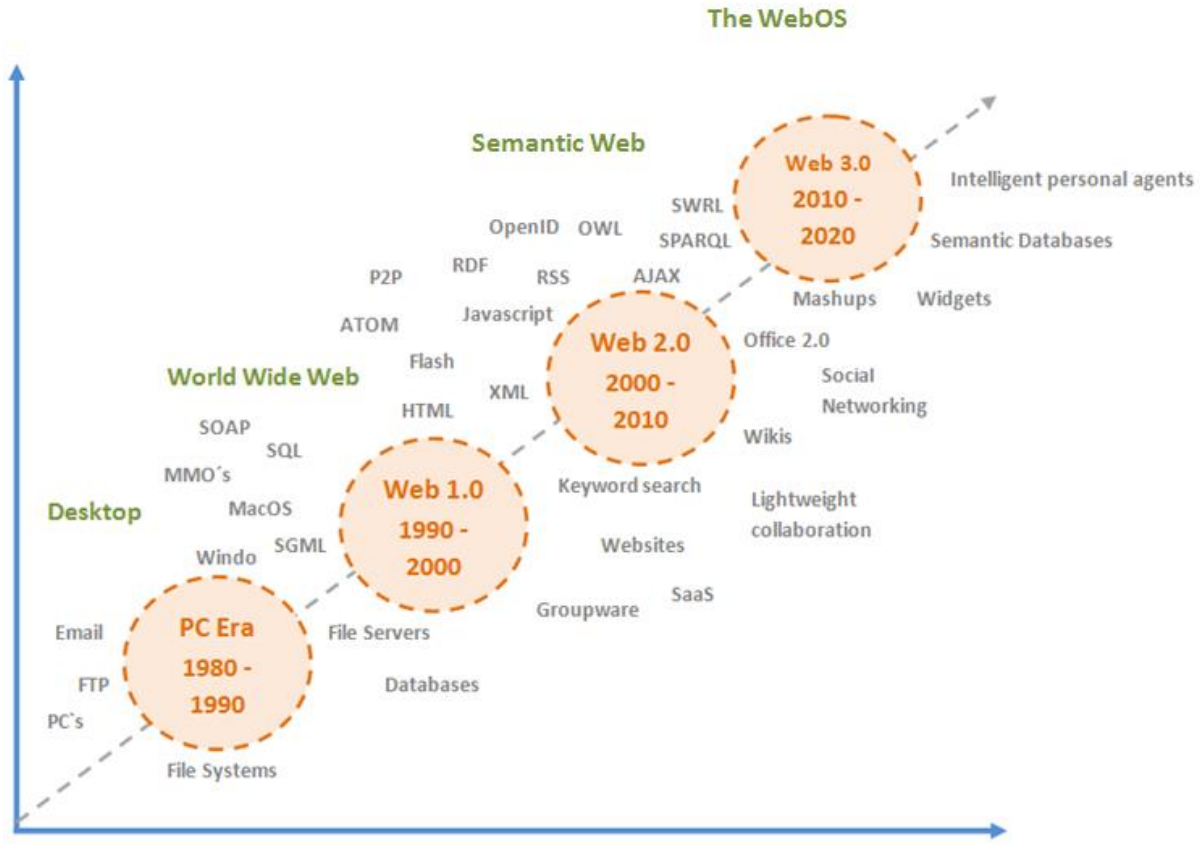


Ilustración 8. Evolución de la Web. Tecnologías y aplicaciones [54]

La Ontología es considerada en filosofía una rama de la metafísica que estudia lo que hay, o la existencia de las cosas. Además la Ontología también estudia la relación entre las entidades que existen [21]. En el documento de Berners-Lee citando anteriormente [20], el autor nos indica el significado de ontología como "un documento que formalmente nos define las relaciones entre términos", añadiendo además que la ontología más común utilizada en la Web es una taxonomía [1] y un conjunto de reglas de inferencia. Según Berners-Lee la taxonomía define clases de objetos y la relación entre ellos[20], mientras que también podemos encontrar definiciones de taxonomía que indican que son un conjunto organizado de palabras o frases para la organización de la información [22]. Se pensó desde un principio, que las posibilidades que ofrecía el uso de clases, subclases y relaciones con respecto a la Web sería de gran utilidad, aprovechando también elementos como la herencia de clases, y propiedades. En capítulos posteriores, se explicará de forma más detallado el concepto de ontología, así como la visión de distintos autores e investigadores en este campo.

Además de la representación de la información mediante el uso de ontologías, es de gran importancia para la Web Semántica la utilización de reglas de inferencia. Como se ha explicado anteriormente, mediante lógica se pueden llegar a deducir relaciones que implican "nuevo" conocimiento. Si pensamos qué tipos de tareas se podrían llevar a cabo con la utilización de la Web Semántica, podemos un simple ejemplo como el siguiente: Si un hospital está relacionado con una zona sanitaria, y esa zona sanitaria está relacionada con una provincia, mediante reglas de inferencia podremos deducir que el hospital está relacionado con una provincia, y por tanto se podrán realizar distintas tareas automáticamente, de acuerdo a las leyes o características concretas de esa provincia, sin necesidad de incluir todos los datos. Tal como detalla el creador de la Web, Tim Berners-Lee, las máquinas no llegan a entender en ningún momento el significado de los conceptos que se están tratando [20], pero pueden llegar a resolver tareas complejas que implican un gran valor para las actividades cotidianas de las personas, mediante la utilización de ontologías y reglas de inferencia.

En la definición inicial de la Web Semántica [20], se explica también la importancia de los agentes (software) que utilizarán los datos de la Web Semántica para procesarlos, e intercambiar información con otros programas. Combinando un buen funcionamiento de este software con los recursos que ofrece la Web Semántica permitirá hacer la Web más legible, y procesar mejor los datos (con más significado), compartiendo incluso información o resultados entre los propios agentes que realicen las tareas de procesamiento. Los agentes considerados consumidores y productores, podrían intercambiar ontologías, que serían los vocabularios necesarios para "entender" y procesar los datos. Podrían también añadir nuevas capacidades de razonamiento al conocer nuevas ontologías.

[1] Conjunto organizado de palabras o frases para la organización de la información [22].

Existe debate actualmente, sobre la cuestión de haber transitado, en este caso hacia la Web 3.0. En primer lugar, hay que resaltar que no existe una definición concreta sobre lo que debe ser la Web 3.0, pero sí se sabe que ésta podría estar basada en la utilización de distintas tecnologías según W3C [13], a describir en los siguientes puntos:

- **Consultas**: Según la visión del W3C [13], la web debería avanzar hacia la "web de los datos", en lugar de hacia la "web de los documentos" actual (web 1.0 y 2.0). Esto tendría que ver con la forma en que se ordenan los datos en el gran repositorio de información, que representa la Web. Por ello, según la idea de esta organización, un objetivo fundamental sería establecer las condiciones y los sistemas que den soporte a la utilización de forma automática de la información por parte de máquinas, de forma segura y fiable (utilización en diferentes capas). Además, una tecnología importante en este apartado sería la realización de consultas mediante SPARQL, que es un lenguaje estandarizado para realizar consultas sobre RDF. En un próximo apartado, se explicará en profundidad RDF, pero se puede decir a alto nivel, que es un modelo de datos estándar para Web, que facilita la integración o relación entre distintos tipos de esquemas[2]. RDF permite definir lo que se suele denominar "triples", que estarían formados por sujeto, predicado y objeto, permitiendo representar relaciones entre sujeto y objeto. Utilizando este modelo, RDF permite integrar, mezclar y reutilizar información de diferentes sistemas [14], mediante la utilización de la información por los agentes[3] ó máquinas. Por ello, debido a la necesidad de integrar RDF para avanzar en la Web Semántica, es necesario incluir un lenguaje de consultas adecuado para RDF, como es SPARQL[4].

- **Ontologías / vocabularios**: Una de las definiciones de ontología más utilizada y aceptada es la que realizó Tom Gruber en 1992 [15], en la cual se dice que una ontología es una descripción conceptual de un dominio. Añade además, que una ontología es una descripción de conceptos y relaciones que pueden existir o ser utilizados por los agentes software (software agents). No existe una clara diferenciación a la hora de utilizar el término vocabulario u ontología, pero en general se considera que la ontología es una definición más extensa y compleja de la definición conceptual de un dominio, mientras que los vocabularios no presentan tales

[2] Esquema (Schema): describen el contenido y la estructura de la información, de una forma precisa [63]
[3] Agentes: Aplicaciones informáticas con capacidad de decidir cómo deben aztuar para alcanzar sus objetivos [64]
[4] SPARQL: SPARQL Protocol and RDF Query Language. Se trata de un lenguaje estandarizado para la consulta de grafos RDF [65]

restricciones como en las ontologías, y constituyen los elementos básicos para la inferencia en la Web semántica. Estos vocabularios, que pueden ser muy complejos incluyendo miles de relaciones y conceptos, son usados por distintos agentes software.

La utilización de ontologías tiene grandes ventajas para la Web semántica, ya que mediante la utilización de este recurso se mejora la comprensión de la información disponible en la Web por parte de los agentes software.

Como ejemplo, podemos considerar la utilización del concepto “coche” y “vehículo”, que mediante una ontología representada con los lenguajes disponibles en la Web semántica (RDF, RDFS, SKOS, RIF y OWL), podemos relacionar ambos conceptos con una relación “same-as”. Este simple ejemplo, permitiría a los agentes software hacer una mejor utilización de la información disponible en la Web, consiguiendo mejores consultas en los buscadores, inferencia de resultados, y reutilizar la información disponible en distintas ontologías.
Teniendo en cuenta que uno de los elementos más importantes para la realización de este proyecto son las ontologías, se desarrollará y ampliará la información sobre ontologías en otro apartado de esta memoria [1.3, Ontologías].

- **Relación entre datos:** Entre los objetivos fundamentales de la Web semántica, es dar mayor “sentido” y comprensibilidad a los datos que existen en la Web a fin de que los agentes o máquinas puedan obtener mejores resultados, y esto se puede realizar mediante en enlace entre datos, conceptos, y la utilización de vocabularios u ontologías. En este caso, haciendo hincapié en la web con datos relacionados, podemos obtener mejores resultados del uso de la Web, ya que cuando un agente “utiliza” un dato, no solo tiene la información individual sobre éste, sino que puede disponer de la información de todos sus datos relacionados. Según el creador de la Web, Tim Berners-Lee [16], la diferencia entre la Web de los datos (Web semántica) y la Web anterior, basada en documentos hipertexto, es que en ésta última encontramos enlaces a otros documentos HTML, mientras que en la Web semántica tenemos relaciones de los datos con otros datos arbitrarios definidos en RDF.

  Berners-Lee, describe cuatro reglas [16], pero no de forma estricta, es decir, las describe como “oportunidades” que nos puede ofrecer la Web de los datos. El hecho de no cumplir una de las reglas, no implica que se esté utilizando mal la Web semántica:

  - Utilizar URIs como si fuesen nombres de elementos o cosas.

- Utilizar URIs HTTP (Teniendo en cuenta que una URI HTTP es una URL [2]).
- Cuando se analice o utilice una URI, considerando este elemento como un nodo de un grafo, se debe entregar información útil sobre ese nodo, mediante las herramientas que ofrece RDF y SPARQL. Esto se haría obteniendo todos los nodos para los cuales el actual representa un sujeto u objeto.
- Incluir enlaces a otras URIs, de tal modo que se pueda descubrir más.

Como una de las características más importantes, la relación entre datos está incluida en la gran mayoría de aplicaciones de la Web semántica. Uno de los ejemplos más mencionados es DBpedia, que incluye el contenido existente en Wikipedia pero en este caso disponible en RDF.

- **Inferencia:** Como definición genérica de inferencia (o deducción), encontramos que es sacar una consecuencia o deducir una cosa a partir de otra [17]. Este concepto aplicado a la web semántica, permite encontrar nuevas relaciones en el "universo" con el que se está trabajando. Teniendo en cuenta que en la web semántica los datos son modelados como un conjunto de relaciones, mediante inferencia podemos crear nuevas relaciones basándonos en reglas (RIF), y en la representación con vocabularios/ontologías (RDF(S), OWL) [18]. Estas nuevas relaciones inferidas podrían ser añadidas explícitamente a las ya existentes, o ser devueltas directamente a algún tipo de consulta. Estos hechos se obtienen mediante la utilización u observación de un dominio determinado.

  Se puede decir que las ontologías clasifican la información, utilizando clases, subclases, relaciones y atributos. Por otra parte, las reglas nos permiten obtener inferencias (o deducciones) válidas en el universo con el que se está trabajando. En la Web Semántica, la utilización de la inferencia nos puede permitir un mejor análisis de los datos, así como una mejor gestión del conocimiento existente en la Web. Es posible por tanto, utilizando esta característica, obtener más información que mejoraría los resultados de muchas de las aplicaciones utilizadas hoy día por los usuarios de la Web (libros, vuelos, hoteles, búsquedas, etc…).

- **Aplicaciones verticales:** En W3C denominan aplicaciones verticales, a aquellas para las que existe colaboración con distintas organizaciones, instituciones y empresas con el fin de explorar los posibles usos y funcionalidades que podrían desarrollarse utilizando la Web Semántica. Los campos más destacados podrían ser los relacionados con la salud, la biotecnología, ciencias naturales, gestión de documentos, librerías, etc…

En esta colaboración entre W3C y otras organizaciones, se avanza en la investigación de vocabularios u ontologías adaptados a su dominio del problema, tipos de consultas necesarias, así como posibles inferencias a desarrollar para obtener mejores resultados en la Web Semántica con respecto a la Web actual. Uno de los grupos más destacados por el W3C en este ámbito, es "The Health Care and Life Science Interest Group", donde se han conseguido avances en la usabilidad de la Web Semántica, la gestión de pacientes, procedimientos de aprobación de medicamentos, y gestión del conocimiento en las publicaciones científicas.

2.3 Ontologías

2.3.1 Introducción

En los últimos años ha aumentado el uso de las ontologías en el campo de la Inteligencia Artificial y la Informática, utilizándose este recurso en tecnologías Web, integración de bases de datos, sistemas multi-agente, procesamiento del lenguaje natural, educación, recuperación de información en la web o buscadores, etc..

Se puede partir de la idea de que la representación formal de conocimiento está basada en la conceptualización, es decir, la utilización de objetos y conceptos que pueden estar relacionados entre sí sobre un área de interés [33], esta definición la amplía Gruber, quien propuso una de las definiciones más utilizadas y aceptadas, (1993) [34] añadiendo que una ontología es una especificación formal y explícita de una conceptualización compartida". Una conceptualización es una visión abstracta o simplificada del mundo que deseamos representar para algún cometido. Además habría que tener en cuenta que los sistemas basados en conocimiento sólo pueden procesar aquello que está representado. Gruber también añade que, cuando el conocimiento del dominio es representado de forma declarativa y formal, se puede decir que el conjunto de objetos representados es llamado universo del discurso. Este conjunto de objetos junto con las relaciones entre ellos y el vocabulario de representación representan conocimiento [33].
Además Gruber añade que en las ontologías existen definiciones que relacionan las entidades o elementos del universo del discurso (clases, relaciones, funciones u otros objetos) con los textos tradicionales utilizados por las personas, incluyendo también axiomas que restringen la correcta interpretación de los términos utilizados.

Podemos encontrar también la definición de ontología aportada por Hendler (2001), que dice: "una Ontología es un conjunto de términos de conocimiento, que incluye un vocabulario, relaciones y un conjunto de reglas lógicas y de inferencia sobre un domino en particular". Ésta última definición se ajusta más al tipo de ontología desarrollada en OWL (Hendler ha participado en los proyectos W3C sobre OWL), donde las relaciones entre elementos y reglas lógicas y de inferencia

son de gran importancia para la representación de conocimiento utilizando este lenguaje.

2.3.2 Características de las ontologías

Pasamos a describir en este apartado algunos elementos y características más destacados sobre las ontologías. Según comenta el autor Iván J. Flores [40], citando también a otras fuentes [37], [38], [39]:

- Las Ontologías proveen un vocabulario común y sin ambigüedades para referirse a los términos en el área aplicada, pudiéndose compartir o reutilizar estos entre diferentes aplicaciones que hagan uso de la Ontología.

  Esto es el caso de los agentes software (en el ámbito de las tecnologías Web), que pueden reconocer adecuadamente los elementos de una ontología siempre y cuando se cumplan las condiciones anteriores.

- Además de un vocabulario común, especifican una taxonomía o herencia de conceptos que establecen una categorización o clasificación de las entidades del domino. Una buena taxonomía es simple y fácil de recordar, separa sus entidades de forma mutuamente excluyente, y define grupos y subgrupos sin ambigüedad.

- El vocabulario y la taxonomía representan un Marco de Trabajo Conceptual para el análisis, discusión o consulta de información de un domino.

- Una Ontología incluye una completa generalización/especificación de sus clases y subclases, las cuales están formalmente especificadas (incluyendo sus relaciones e instancias) asegurando la consistencia en los procesos deductivos.

- Las Ontologías son implementadas en un lenguaje específico de representación Ontologica (*ontology representation languages*) de manera que la especificación de sus clases, relaciones entre éstas y sus restricciones dependerán de las características de dicho lenguaje.

  En este proyecto nos centraremos en la representación de conocimiento utilizando ontologías OWL, que se explicará en detalle en otro apartado.

2.3.3 Modelado de ontologías

A la hora de modelar ontologías podemos encontrarnos diferentes técnicas de modelado y lenguajes [40], ya que pueden ser altamente informales, expresadas en lenguaje natural, semi-informales (expresadas en un lenguaje natural restringido y estructurado), semi-formales (expresadas en un lenguaje artificial y formalmente definido: RDF, OWL, etc) y rigurosamente formales si poseen términos definidos meticulosamente con semántica formal, teoremas y propiedades que les dan un sentido completo.

Según los autores [40] [41], a principios de los 90, las Ontologías fueron construidas utilizando principalmente técnicas de representación de conocimiento en Inteligencia Artificial (IA), como son los Marcos y la Lógica de Primer Orden. En los últimos años, se han usado otras técnicas de representación de conocimiento basadas en lógica descriptiva para modelarlas y se implementan utilizando distintos lenguajes para tal fin. Otras maneras que también se han aprovechado para modelar Ontologías, son el Lenguaje de Modelado Unificado (UML) en Ingeniería de Software y Diagramas E/R, utilizados en Base de Datos para modelar conceptos y las relaciones entre estos.

Como se puede observar en lo descrito anteriormente por otros autores, existen lenguajes de modelado de ontologías como pueden ser OWL y UML, que aunque surgieron para objetivos distintos (OWL, vocabularios/ontologías Web Semántica, y UML Ingeniería del Software diseño de sistemas), podemos encontrar puntos de coincidencia entre ambos y reutilización de recursos que ofrecen ambos lenguajes para continuar el análisis que se realiza en este proyecto para la recuperación de información desde OWL a lenguajes de modelado como UML o modelos de representación de información como son RSHP.

Pasamos a describir el modelado de ontologías:

- **Modelado de ontologías utilizando marcos y lógica de primer orden.**

Algunos autores como Gruber (1993) [38] ,describieron que las ontologías disponían de los siguientes elementos, explicados también por el autor J. Flores [40]:

- **Clases**: Son los elementos principales de la ontología ya que representan conceptos del dominio. Suelen ser organizadas en taxonomías pudiendo utilizar mecanismos de herencia.

- **Relaciones**: Representan las interacciones entre los objetos del dominio. Generalmente en las ontologías se utilizan relaciones entre dos elementos, siendo normalmente uno de ellos denominado dominio y el otro rango.

- **Funciones**: Son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de una Ontología.

- **Instancias**: Representan objetos determinados de un concepto o clase.

- **Axiomas**: Se usan para modelar sentencias que son siempre ciertas. Los axiomas permiten, junto con la herencia de conceptos, inferir conocimiento que no esté indicado explícitamente en la taxonomía de conceptos. En especial los axiomas son usados para verificar la consistencia de la Ontología. También son utilizados para inferir nuevo conocimiento.

- **Modelado de ontologías utilizando Lógica Descriptiva**

Las lógicas descriptivas (DL) son una familia de formalismos de representación del conocimiento. Describen los conceptos relevantes de un dominio.

Definen los conceptos relevantes de un dominio y los relacionan para especificar propiedades. Los bloques de construcción sintácticos básicos son los conceptos atómicos (predicados unarios), los roles atómicos (predicados binarios) e individuos (constantes). El poder expresivo de este lenguaje está restringido a un conjunto pequeño de constructores para construir conceptos complejos y roles. Además, el conocimiento implícito sobre conceptos e individuos puede inferirse automáticamente con la ayuda de procedimientos de inferencia. Podemos encontrar los siguientes elementos:

- **TBox**: contienen sentencias describiendo conceptos jerárquicos (e interrelaciones entre conceptos). Se podría decir que en este grupo estarían incluidas las clases.
- **ABox**: Contiene sentencias que indican a donde pertenecen los individuos en la jerarquía (relaciones entre individuos y conceptos). Se podría decir que en este grupo estarían incluidas las instancias.

- **Modelado de ontologías utilizando UML**

UML es uno de los lenguajes de modelado más utilizados para el diseño de sistemas software. Tiene el soporte del OMG (Object Management Group), que mantienen y actualizan las especificaciones de de este lenguaje de modelado. Centrándonos en el diseño de diagramas de clases con UML, podemos ver que se pueden representar clases, atributos de clases, jerarquías de clases, relaciones, y asumiendo otras reglas en función del diseño también instancias.

OCL (Object Constraint Language) siendo un lenguaje de descripción formal de expresiones en los modelos UML, podría utilizarse junto a éste último para representar ciertas especificaciones de las ontologías.

Además, OMG trabaja en algunos proyectos como puede ser ODM (Ontology Definition Metamodel), donde este grupo actualiza las especificaciones a nivel de representación de ontologías (OWL, RDF).

- **Modelado de ontologías utilizando diagramas E/R**

Los diagramas o modelos Entidad-Relación es una herramienta para el modelado de datos que permite representar entidades relevantes así como interrelaciones y propiedades. La entidad es un objeto del mundo real distinguible de otros objetos, las unidades se describen utilizando un conjunto de atributos. Estos diagramas incluyen algunas restricciones pero no incluyen operaciones. Según hace referencia el autor Iván J. Flores [40], al trabajo descrito por Gomez, Fernandez y Corcho, (2004), las clases pueden ser representadas mediante Entidades-ER y estás pueden ser organizadas en taxonomías utilizando relaciones de generalización entre los elementos. Los atributos de las clases podrían ser representados mediante Atributos-ER, y utilizando las relaciones-ER y utilizando las entidades-ER se representarían las relaciones correspondientes.

2.3.4 Lenguajes utilizados para la representación de ontologías

El W3C así como varios autores de publicaciones [44], [45] (aquí solo se menciona alguna publicación, pero hay muchas más al respecto), coinciden en el apoyo a la representación de ontologías mediante lenguajes como RDF, RDFS y OWL (basado en los anteriores). Mediante la utilización de OWL se ha conseguido un nivel bastante alto en cuanto a la precisión de la definción de los elementos de una ontología, así como recursos como reglas lógicas que permiten inferir conocimiento, esto junto con el apoyo y las especificaciones de W3C hacen que sea uno de los lenguajes más utilizados actualmente.

Los lenguajes más utilizados para la definición de ontologías serían los siguientes:

- RDF
- RDFS
- DAML+OIL
- OWL

En otro apartado de este documento se explicarán en detalle los lenguajes que son tratados en este proyecto (RDF, RDFS, OWL).

2.3.5 Razonamiento en ontologías

Lo que se hace en el campo de la Inteligencia Artificial, emulando las capacidades o técnicas utilizadas por los seres humanos, es la capacidad de razonar, es decir, capacidad de inferir nuevo conocimiento a partir del disponible. En IA, esto se realiza generalmente mediante el conocimiento de que se dispone (hechos) y una serie de reglas o lógica que se utilizan para obtener nuevo conocimiento.

En las ontologías se utilizan distintas técnicas para realizar la inferencia, dependiendo de los elementos disponibles en cada ontología, las operaciones que puedan realizar y los objetivos que se persigan. Podemos encontrar tres técnicas (las más utilizadas) [40]:

- Razonamiento con lógica de primer orden
- Razonamiento con lógica descriptiva
- Razonamiento con reglas

2.4 Lenguajes para la representación de ontologías Web

2.4.1 Introducción

Debido al gran crecimiento de la Web, la cantidad de información ha crecido enormemente, y esto junto con las nuevas necesidades de los usuarios, como son, las búsquedas (cada vez más exigentes), utilización de páginas de todo tipo que tienden a ser cada vez más personalizadas, en función de los perfiles y características de cada usuario (reservas de viajes, consultas de médicos, compras personalizadas, redes sociales, etc..), surgió la necesidad de hacer un lenguaje para describir las ontologías que se utilizarían en la Web Semántica. Uno de los objetivos principales es hacer que la información disponible en la Web sea más legible por las máquinas, y por tanto más procesable para realizar tareas de cualquier tipo.

El desarrollo de OWL ha sido influido por lenguajes de objetos basados en marcos, lógica descriptiva y los lenguajes web existentes [46]. OWL fue desarrollado para que las máquinas pudieran procesar mejor la información disponible en la Web.

El desarrollo de OWL ha sido basado en las recomendaciones y estándares de W3C, ofreciendo recursos y elementos para que los desarrolladores creen sus propias ontologías y puedan extender otras. OWL está construido en XML, lo que permite a los usuarios incluir anotaciones que pueden ser procesadas por las máquinas.

2.4.2 Capas de lenguajes en la Web Semántica

Mientras que la Web actual está más enfocada hacia la utilización y lectura por parte de las personas, los desarrolladores han tenido como objetivo crear unos lenguajes estructurados en una arquitectura de capas, teniendo cada capa más recursos y posibilidades que la capa anterior. Hay que tener en cuenta que cada una de las capas depende también de su capa anterior [46].

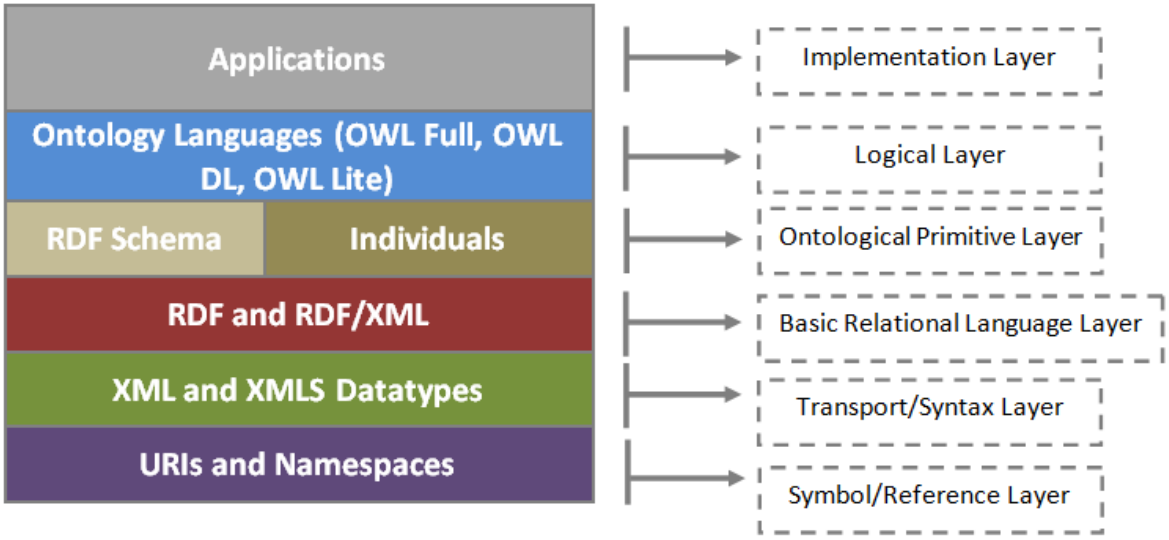


Ilustración 9. Arquitectura de capas en la Web Semántica [46]

Describimos brevemente la tecnología utilizada en cada una de las capas [46]:

- **URIs and Namespaces**: Provee símbolos e identificadores para los objetos que son descritos en las ontologías, mediante URIs y namespaces.

- **XML and XMLS Datatypes:** XML es utilizado para serializar la sintaxis OWL. XML es un metalenguaje para especificar formatos (sintaxis pero no semántica). XML Schema (XMLS), es utilizado para para definir tipos de datos estándar.

- **RDF** es utilizado para ofrecer el lenguaje relacional básico de la Web Semántica. Puede ser utilizado para describir sentencias con pares atributos/valor que describen objetos.

- **RDFS** proporciona la capa inicial de ontologías. Es usado para utilizar un vocabulario estándar para los elementos del modelo. RDFS también proporciona semántica sobre clases, las subclases y propiedades. Las

instancias son definidas mediante RDF, además de los valores o propiedades especificados en las instancias.

- **OWL** (todos sus tipos), son utilizados para especificar ontologías (clases, propiedades, restricciones). OWL es parte de una serie de capas relacionadas definidas por W3C.

2.4.3 RDF (Resource Description Framework)

RDF es un modelo estándar para el intercambio de datos en la Web. RDF facilita la integración de datos incluso con aplicaciones heterogéneas. RDF permite que los esquemas puedan cambiar durante el tiempo sin tener que cambiar toda la información introducida por los usuarios[47].

RDF extiende las propiedades de enlace en la Web, mediante la utilización de URIs para las asociaciones entre los extremos de dos elementos (generalmente relaciones llamadas triples) , donde está formado por relaciones sujeto-predicado-objeto. Por ello, RDF permite compartir y extender la información utilizada en la Web [47].

RDF es utilizado para especificar instancias en OWL, y como dice el propio nombre es un marco para descripción de recursos. Un recurso es cualquier cosa que pueda ser identificada mediante una URIref [46].

2.4.3.1Modelo RDF

El modelo de datos de RDF se define con tres elementos fundamentales que son recursos, literales y declaraciones. Define las propiedades como un tipo especial de recurso y los literales tipados como un tipo de literal. Las declaraciones son pares (atributo/valor) formados por propiedades clasificadas por un nombre y unos valores asociados.

Recursos

Los recursos son los elementos fundamentales de RDF. Los recursos están generalmente identificados con nombres. Un recurso en RDF es cualquier cosa que pueda ser identificada mediante una URIref. Los recursos son siempre nombrados utilizando URIref. Como ejemplos de elementos físicos que pueden ser identificados mediante URIs tenemos libros, menú, bebida… Y como elementos “virtuales” tenemos páginas web, imágenes, etc…

Propiedades

Las propiedades son un tipo específico de recurso que se utiliza como predicado en las declaraciones para describir otro recurso. Las propiedades describen atributos de recursos y relaciones entre recursos. Las propiedades describen una relación binaria entre un sujeto y un valor.

Literales

Los literales son cadenas de texto que opcionalmente pueden contener identificadores e identificadores de tipos de datos.

Declaraciones

RDF es utilizado para realizar simples declaraciones utilizando recursos. Las declaraciones RDF ofrecen un sujeto (recurso), un predicado (una propiedad con nombre), objeto (puede ser un literal o un recurso con el valor de la propiedad destinado al sujeto).

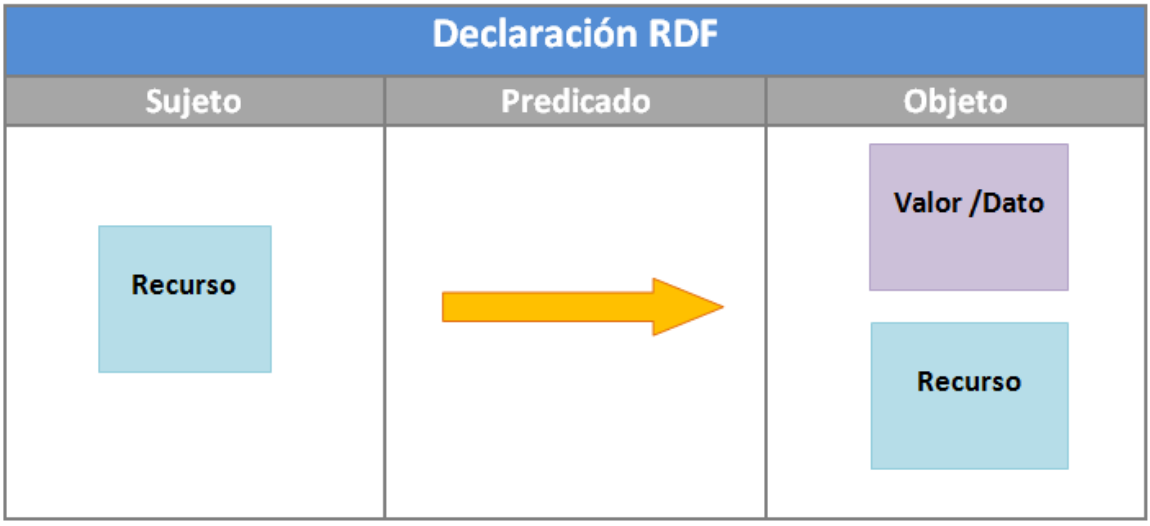


Ilustración 10. Elementos básicos en sentencias RDF

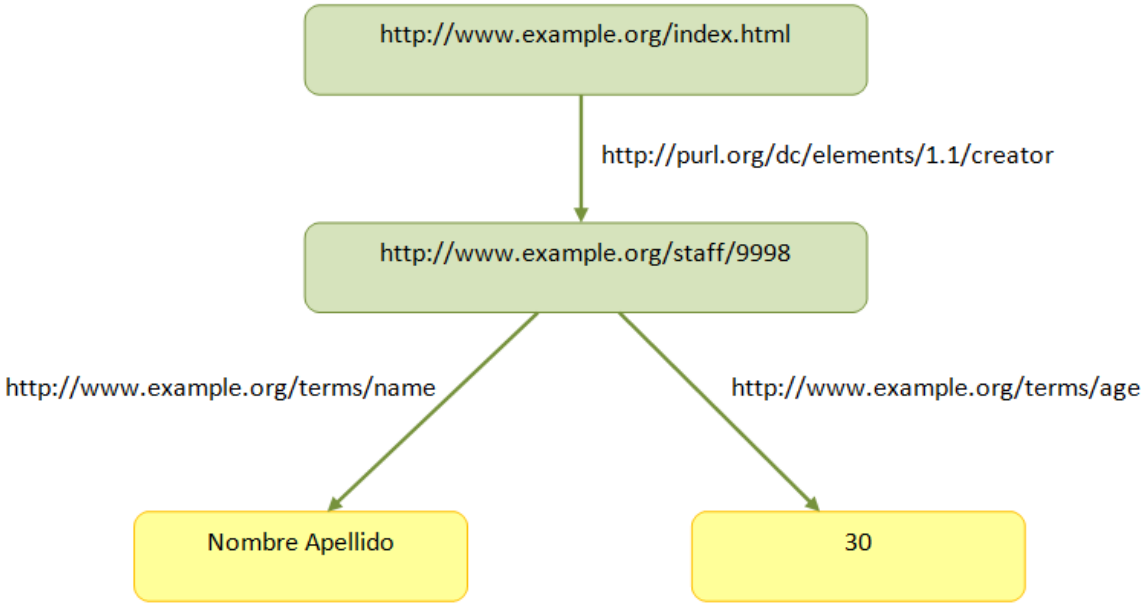


Ilustración 11. Esquema de utilización de recursos en RDF

Como se ha visto hasta ahora se pueden realizar con RDF declaraciones sencillas, pero si queremos representar una semántica más compleja tendrá que realizarse como una ontología.

Se muestran a continuación sólo los constructores que serán utilizados por OWL.

Categoría	Constructor	Finalidad
Identificación de recursos	rdf:ID attribute	Establece el identificador del recurso
	rdf:about attribute	Referencia al identificador del objeto
	rdf:resource attribute	Identifica el valor primario en un valor compuesto
	rdf:value attribute	Especifica la pertenencia de la clase
	rdf:type property	Indexador de acceso

Tabla 1. Recursos en FDF

2.4.4 RDFS (Resource Description Framework Schema)

RDFS es parte de la capa más inicial en cuanto a descripción de ontologías. Además, RDFS añade un vocabulario estandarizado para describir conceptos. Podemos encontrar estructuras básicas como clases y propiedades que son formalmente descritas. RDFS está diseñado a partir de RDF, a fin de expandirlo y ofrecer más semántica y vocabulario.

Se considera que RDFS es un lenguaje descriptivo, ofreciendo elementos para representar dominios añadiendo constructores a los ya existentes en RDF. El vocabulario que podemos encontrar en RDFS es una colección de clases y propiedades de descripción.

Categoría	Constructor	Descripción
Propiedades principales	rdf:type	Es utilizado para asociar una instancia con una clase.
	rdf:subClassOf	Es utilizada para representar especialización entre clases para formar jerarquías.
	rdfs:subPropertyOf	Igual que ocurre con las clases, las propiedades pueden presentar especialización entre ellas.
Propiedades de restricción	rdfs:range	Se puede restringir la relación hacia una clase o un dato (datatypes).
	rdfs.domain	Esta propiedad limita el sujeto en una propiedad hacia una determinada clase.
Propiedades de información	rdfs:seeAlso	Esta propiedad aporta información adicional sobre el sujeto.
Propiedades de	rdfs:label	Esta propiedad se utiliza para añadir texto que pueda

documentación		ser leído por personas y que pueda ser útil en una interfaz de usuario. Puede presentarse en una interfaz (dopb
	rdfs:comment	Esta propiedad se puede utilizar para añade información textual que puede ser leída por personas
Clases RDFS predefinidas	Rdfs:resource	Clase raíz de todos los recursos
	rdfsProperty	Superclase de todas las propiedades
	Rdfs:class	Superclase de todas las clases
	Rdfs:Literal	Superclase de todos los valores literales
	Rdfs:Datatype	Identifica datos o valores (datatypes)

Tabla 2. Algunos constructores en RDF

2.4.5 OWL, (Web Ontology Language)

OWL fue diseñado para ser utilizado por aplicaciones que necesiten procesar el contenido de la información y no para realizar otras tareas como pueden ser la visualización del contenido dirigida hacia las personas. OWL facilita y mejora la utilización de la información que existe en la Web por parte de las máquinas, mejorando el procesamiento y por tanto las tareas que puedan realizarse de forma automática. OWL mejora las capacidades ofrecidas por otras tecnologías utilizadas en este ámbito como son RDF, RDFS y XML, aportando más vocabulario y recursos para describir formalmente la semántica a tratar.

OWL dispone de tres sublenguajes que incrementan progresivamente la expresividad de las ontologías definidas:

- **OWL Lite**: Este nivel del lenguaje dentro de OWL nos permite representar aquellos elementos básicos que se necesitan para diseñar una ontología, como pueden ser elementos de clasificación jerárquica y restricciones

simples. Aunque OWL Lite sí permite el uso de cardinalidad, sólo pueden usarse cardinalidades de 0 ó 1. En la mayoría de casos OWL Lite sería apropiado para el diseño de ontologías que no tengan relaciones semánticas muy complejas, pudiendo también reutilizar de forma sencilla los elementos diseñados en XML y RDF/S.

- **OWL DL:** Este nivel del lenguaje OWL estaría indicado para los usuarios que quieren una gran expresividad y a la vez mantener unas buenas prestaciones computacionales con unos tiempos de ejecución finitos. OWL DL incluye todos los constructores disponibles en OWL, pero sólo pueden ser usados con restricciones.

- **OWL Full:** Este nivel de lenguaje dentro de OWL está indicado para los usuarios que quieren un nivel de expresividad máximo pero sin garantías computacionales (tiempo necesario para ejecutarlo puede ser alto). OWL Full no tiene restricciones sobre RDF e implica el lenguaje OWL completo y sin restricciones.

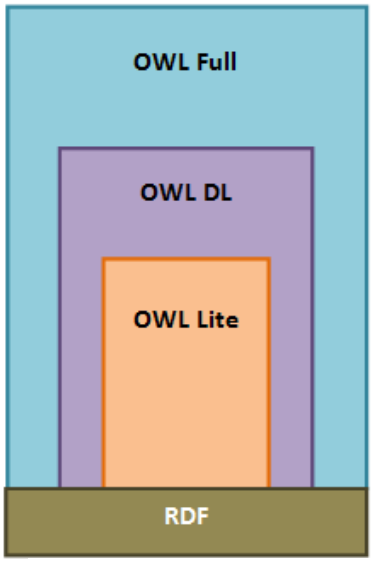


Ilustración 12. Lenguajes OWL

Podemos ver en la figura anterior como cada nivel de lenguaje incluye al anterior.

2.4.5.1Construcción de ontologías OWL

Las ontologías OWL son diseñadas en documentos que pueden ser referenciados utilizando URIs. En estos documentos se construyen las ontologías utilizando la sintaxis y constructores OWL (además de otros requisitos y elementos necesarios que debe contener el documento, se explicará a continuación). Estos archivos pueden estar separados en conjuntos de ontologías, instancias y definición de tipos de datos, pero podemos encontrarlos en un mismo documento también.

Las ontologías utilizadas pueden ser distribuidas en la Web utilizando distintas localizaciones para los archivos, como pueden ser servidores de una organización concreta, o cualquier otro servidor donde también se podría extender (reutilizar información que contiene otra ontología).

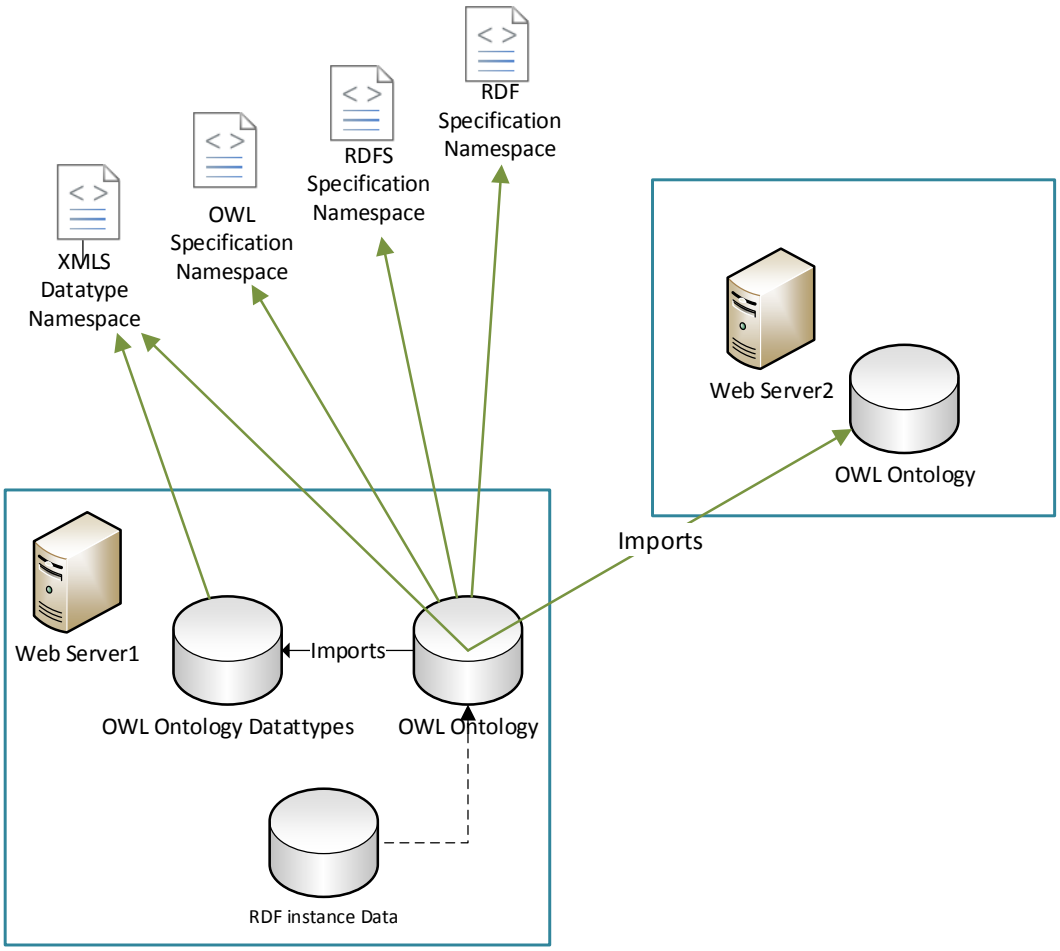


Ilustración 13. Esquema sobre espacios de nombres e importación de ontologías [46]

Como se ha comentado anteriormente, se recomienda separar los archivos que contienen los constructores y las instancias, siendo referenciados por URIs distintos. La estructura general para estos documentos está separada en “header” (cabecera), “body” (cuerpo) y footer (final o pie).

Sobre esos tres apartados (en el archivo de la ontología no se llaman así, sino que es sólo una forma de clasificar las zonas en el documento) se construyen los diferentes elementos necesarios en la ontología OWL.

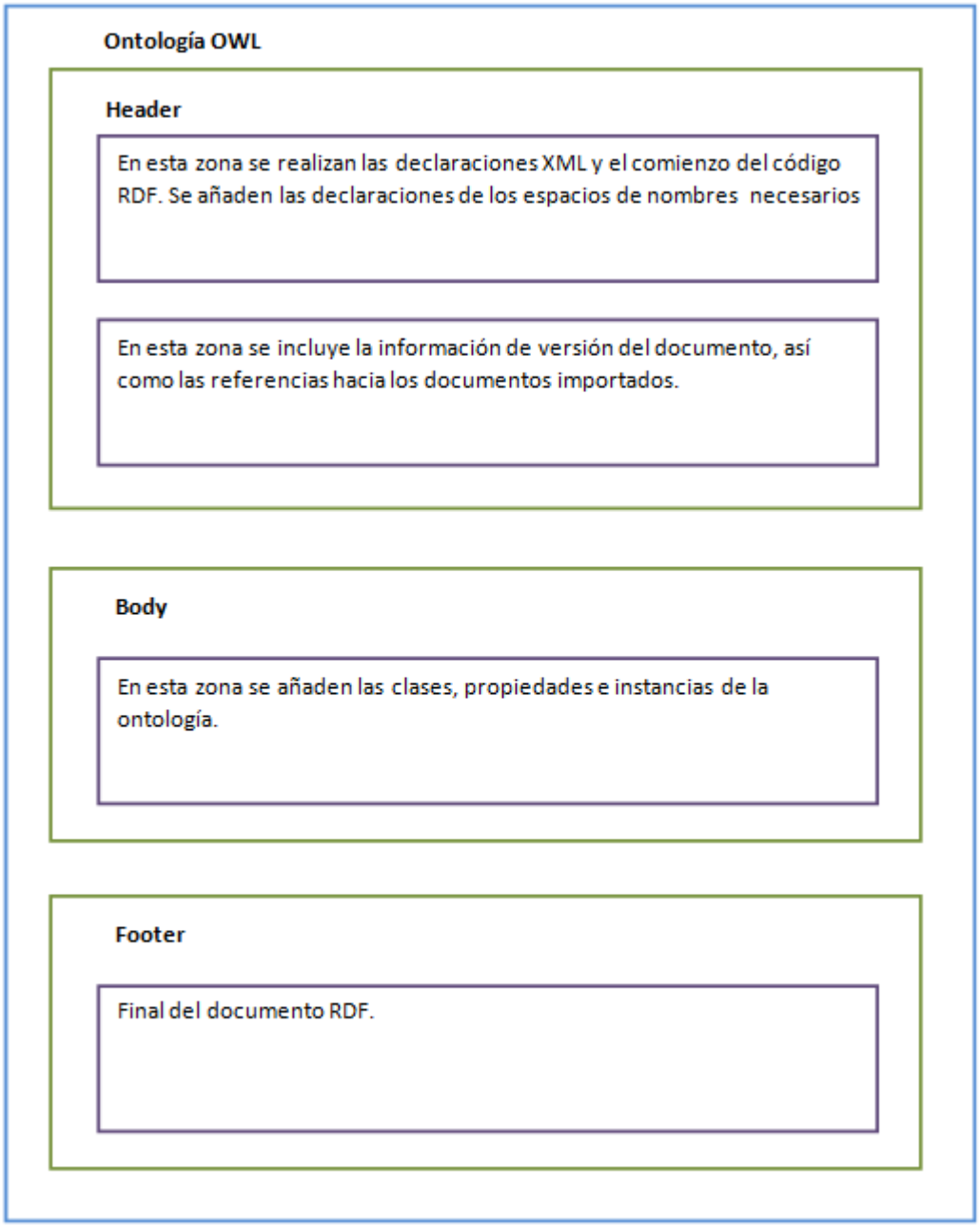


Ilustración 14. Estructura básica de ontologías OWL

2.4.5.2 Elementos de la cabecera en ontologías OWL

Este apartado sirve en la ontología para iniciar la etiqueta RDF, así como declarar el espacio de nombres y los elementos de la ontología (versiones, importación, etc…)

Se explican a continuación en la siguiente tabla:

Cabecera ontología OWL (Namespaces)	
Elemento	**Descripción**
xmlns:xsd = "http://www.w3.org/2001/XMLSchema#"	Este atributo RDF especifica los espacios de nombres para XML Schema
xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#	Este atributo RDF identifica los espacios de nombres para RDFS.
xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#	Este atributo RDF identifica los espacios de nombres para RDF.
xmlns:owl = "http://www.w3.org/2002/07/owl#	Este atributo RDF identifica los espacios de nombres para OWL.
xmlns:food = "http://www.w3.org/TR/2004/REC-owl-guide-20040210/food# **(Ejemplo)**	Este atributo RDF identifica los espacios de nombres para la ontología importada.
xmlns = "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#	Este atributo RDF identifica los espacios de nombres para la ontología actual.

Tabla 3. Espacio de nombres en cabeceras OWL

Cabecera ontología OWL (Elementos de información)	
Elemento	**Descripción**
owl:Ontology	Este elemento se utiliza para instanciar ontologías OWL. Su utilización está motivada en incluir y especificar metadatos como pueden ser información sobre la versión etc… para que las ontologías que necesiten importarla obtengan esta información.
owl:versionInfo	Ofrece una cadena de texto con información sobre la versión de la ontología.
owl:priorVersion	Especifica mediante una URI la versión anterior de la ontología.
owl:backwardCompatibleWith	Este elemento indica que versiones de la ontología son compatibles, esto quiere decir que todos los identificadores que sean iguales tendrán la misma interpretación. Por defecto se asume que una ontología no es compatible con otra.
owl:incompatibleWith	Este elemento se suele utilizar para informar sobre la no compatibilidad con versiones generalmente anteriores a la de la ontología actual.
owl:DeprecatedClass	Este elemento se utiliza para señalar que una determinada clase se dejará de utilizar en la ontología “pronto”, pero sólo funciona de momento como un comentario que se puede considerar.
owl:deprecatedProperty	Este elemento se utiliza para señalar que una determinada propiedad se dejará de utilizar en la ontología en el futuro. Por tanto, indica que esta propiedad no debería ser utilizada en nuevos documentos.
owl:imports	Este elemento permite referenciar ontologías externas para su reutilización. Este mecanismo es muy similar al utilizado en algunos lenguajes de programación como “include” o “using”.

Tabla 4. Elementos de información en ontologías OWL

2.4.5.3Elementos OWL Lite RDF Schema

En OWL Lite podemos utilizar algunas de las características y recursos que ofrece OWL, teniendo algunas restricciones sobre los elementos utilizados.

- **owl:Class**: Las clases definen a un grupo de instancias que tienen propiedades o atributos en común. Las clases pueden ser organizadas utilizando una jerarquía (especialización) utilizando el elemento **rdfs:subClassOf**. Hay que tener en cuenta que existe una clase general llamada Thing, la cual es superclase de todas las instancias y clases de la ontología.

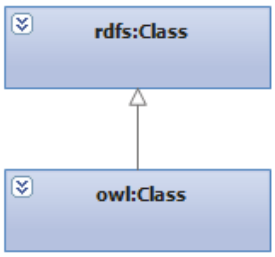


Ilustración 15. Jerarquía en UML

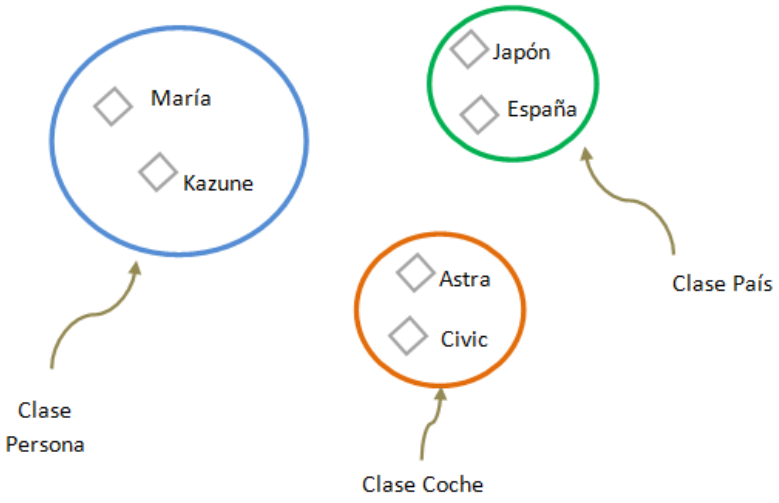


Ilustración 16. Representación conceptual de clases en OWL

- **rdfs:subClassOf**: Las jerarquías de clases pueden ser formadas con declaraciones, especificando que una clase es subclase de otra. Podríamos tener por ejemplo la clase “Animal” y su subclase “gato”, y por tanto una aplicación que aplicase inferencia podría saber que un gato es también un animal.

- **rdf:Property**: este elemento sirve para representar la relación entre instancias o entre instancias y valores o datos. Dentro de esta propiedad nos encontramos otras dos, que son owl:ObjectProperty y owl:DatatypeProperty. La propiedad owl:ObjectProperty se ocupa de las relaciones entre instancias, por ejemplo entre “Persona” , la relación “tieneHermano” y otra instancia que identifique al hermano. Por otra parte, la propiedad owl:DatatypeProperty sirve para relacionar por ejemplo, una instancia “Persona”, la relación “tieneEdad” y un valor numérico, por ejemplo 30

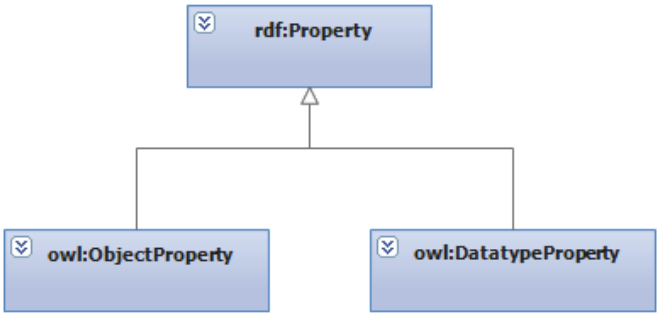


Ilustración 17. Jerarquía de propiedades en OWL

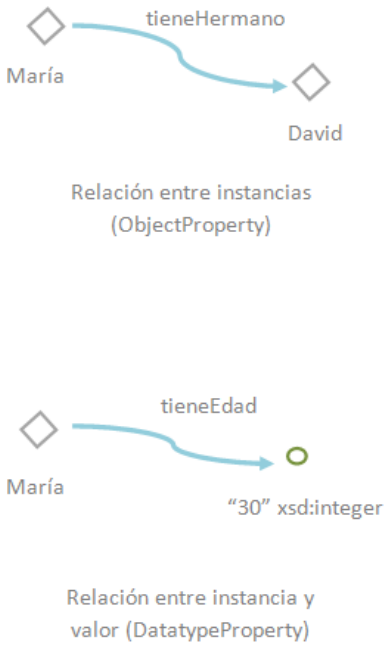


Ilustración 18. Esquema conceptual de propiedades en OWL

- **rdfs:SubPropertyOf**: Al igual que ocurre con las clases pueden formarse jerarquías entre las propiedades definidas. Por ejemplo, si se tiene la propiedad "tieneHijo" que es "subProperty" de otra que es "tieneFamiliar", mediante inferencia se podría deducir que en una relación "tieneHijo" además tiene una relación "tieneFamiliar".

- **Rdfs:domain**: Esta propiedad limita las instancias para las que puede ser aplicada una relación. Por ejemplo, si se tiene como dominio la clase Persona, y como relación "TieneHijo", entonces la instancia de la que parte la relación debe pertenecer a la clase dominio. Hay que tener en cuenta que en OWL las relaciones (ObjectProperty, DatatypeProperty), se consideran con una navegabilidad de dominio hacia rango.

- **rdfs:range**: Esta propiedad limita las instancias que puede tener una propiedad como valor. Si tenemos una propiedad que relaciona dos instancias, y esta propiedad identifica a una clase como su rango, entonces las instancias utilizadas deben pertenecer a esta clase. Como en el caso anterior, hay que tener en cuenta que en OWL las relaciones (ObjectProperty, DatatypeProperty), se consideran con una navegabilidad de dominio hacia rango.

- **Individual**: Son instancias de clases, o propiedades que pueden ser utilizadas para relacionar una instancia con otra.

2.4.5.4Elementos de igualdad y desigualdad en OWL Lite

- **owl:equivalentClass**: Se pueden definir clases equivalentes de tal manera que signifique que las instancias de una clase son también de la otra, por ejemplo entre coche y automóvil. Esta propiedad que dos clases son equivalentes, y que tienen ls mismas instancias, pero no hay que confundirlo con la propiedad owl:sameAs, que identifica dos instancias como exactamente iguales

- **owl:equivalentProperty:** Las propiedades también pueden ser definidas como equivalentes, teniendo entonces que estas propiedades relacionan una instancia con el mismo grupo de instancias para el que es equivalente la propiedad.

- **owl:sameAs:** Se utiliza para indicar que dos instancias son exactamente iguales.

- **differentFrom:** Indica que una instancia es distinta a otras. El uso de esta propiedad ayuda a no llegar a contradicciones cuando se realiza inferencia, teniendo en cuenta que lenguajes como OWL y RDF no asumen que una instancia pueda tener un solo nombre. Si tenemos dos instancias, (dos nombres de la clase “Persona” por ejemplo), mediante inferencia no se podrá decidir que son instancias distintas, mientras que la propiedad “differentFrom” sí lo especifica.

- **Owl:allDifferent:** Esta propiedad nos permite establecer como distintas entre ellos una lista de instancias proporcionada.

2.4.5.5Características de las propiedades

Podemos encontrar en OWL Lite identificadores que ofrecen más información sobre las propiedades utilizadas (ObjectProperty, DatatypeProperty).

- **owl:inverseOf:** Esta propiedad puede ser utilizada para especificar que una propiedad es inversa respecto a otra. Por ejemplo si se tienen dos propiedades P1 y P2, y la propiedad P1 relaciona X con Y, mediante esta propiedad (en P2) tenemos que Y está relacionado con X (recordemos que

en OWL las relaciones tienen navegabilidad desde el dominio hacia el rango).

- **Owl:TransitiveProperty:** Las propiedades pueden establecerse como transitivas, es decir, si tenemos una propiedad transitiva, y el par de la relación es (x,y) es instancia de la propiedad transitiva P, y tenemos otro par con (y,z) siendo instancia de P, entonces el par (x,z) es también instancia de P.

- **owl:SymmetricProperty:** Las propiedades pueden definirse como simétricas. Si por ejemplo una propiedad es simétrica, y tenemos una instancia de de esta propiedad con (x,y), entonces el par (y,x) también es instancia de P.

- **owl:FunctionalProperty:** En algunas ocasiones podremos tener un solo valor (o instancia) asociado a una propiedad. Esta propiedad indica que sólo puede haber un valor para un objeto (dominio). Por tanto en este caso se especifica indirectamente la cardinalidad máxima de la relación en 1.

- **Owl:InverseFunctionalProperty:** Este elemento nos permite indicar que una relación tendrá la característica opuesta que en owl:FunctionalProperty. Esto se traduce en que el dominio de la relación será funcional, es decir, que puede haber como máximo un valor en el dominio para la relación.

2.4.5.6 Restricción en propiedades

OWL Lite permite establecer restricciones sobre las instancias utilizadas en una relación. Estas restricciones sólo pueden usarse dentro del nodo "**owl:Restriction**".

- **owl:allValuesFrom**: Esta restricción es utilizada en las propiedades (relaciones) con respecto a una clase. Esto significa que esta clase tiene una restricción sobre el rango en esta relación. Además, tenemos que si una instancia de esta clase está relacionada por esta propiedad con otra instancia, se puede inferir que el rango de esta relación tiene la misma restricción.

- **owl:someValuesFrom**: Esta propiedad se utiliza en la restricción de una clase para enlazar con otra clase o valor como rango. Esta restricción

describe la clase para la que todas sus instancias tienen al menos un valor para la relación y el rango especificados.

2.4.5.7Restricciones de cardinalidad

OWL Lite permite restricciones sobre la cardinaliad en las propiedades (relaciones) entre clases o instancias de esas clases. En el caso de OWL Lite, debido a las limitaciones de este nivel en OWL existen restricciones en cuanto a la cardinalidad, pudiendo ser sólo 1 ó 0.

- **owl:minCardinality**: Se puede establecer la cardinalidad mínima sobre una determianda clase en una relación. Si por ejemplo la cardinalidad mínima en una propiedad con respecto a una clase es 1, entonces siempre al menos habrá una instancia de esa clase en la propiedad descrita. Esto es una forma indirecta de requerir que todas las instancias de la clase tengan al menos un valor. Esta propiedad describe una clase para la que todas sus instancias tienen al menos N valores (donde N es la cardinalidad mínima) semánticamente distintos en la propiedad relacionada [27]. Este valor se especifica mediante XML Schema datatype con valores no negativos.

- **Owl:maxCardinality**: Se puede establecer la cardinalidad máxima sobre una determianda clase en una relación. Si por ejemplo la cardinalidad mínima en una propiedad con respecto a una clase es 1, entonces siempre al menos habrá una instancia de esa clase en la propiedad descrita. Esto es una forma indirecta de requerir que todas las instancias de la clase tengan al menos un valor. Esta propiedad describe una clase para la que todas sus instancias tienen al menos N valores (donde N es la cardinalidad mínima) semánticamente distintos en la propiedad relacionada [27]. Este valor se especifica mediante XML Schema datatype con valores no negativos.

- **Owl:Cardinality**: Esta restricción es útil en los casos en que se quiere establecer una cardinalidad máxima y mínima que sea igual el valor. Por ejemplo, si especificamos para la clase “Persona” la relación “TieneFechaCumpleaños”, con una cardinalidad máxima de 1, esto hará que al ser procesado este elemento se tenga que sólo existe una fecha de nacimiento.

2.4.5.8 Intersección de clases

- **Owl:intersectionOf:** Es posible describir clases mediante intersección, utilizando clases y otras restricciones especificadas en una lista. Esta propiedad define que todas las instancias de la clase descrita son miembros o cumplen las descripciones incluidas en una lista.

2.4.5.9Otros elementos en OWL DL y OWL Full

- **Owl:oneOf**: Las clases pueden ser descritas como una lista de instancias. Por tanto, la clase tendrá exactamente esas instancias (no puede tener más o menos). Hay que tener en cuenta que en la definición de este elemento podemos encontrar que las instancias incluidas pueden ser definidas de las formas posibles según la sintaxix OWL, utilizando “owl:Thing”, o “ClassName rdf:ID=”InstanceName””

- **Owl:hasValue**: Es posible describir propiedades que tengan una instancia como valor.

- **Owl:disjointWith**: Se puede especificar que algunas clases sean disjuntas entre sí, es decir que estas clases no tienen instancias en común.

- **unionOf, complementOf, intersectionOf :** OWL DL y OWL Full permiten que haya arbitrariedad de valores booleanos en estas descripciones

- **minCardinality, maxCardinality, cardinality:** En OWL DL y OWL Full son permitidas cardinalidades de cualquier valor siempre y cuando sean enteros positivos.

2.5 Herramientas para representación y transformación de ontologías en la actualidad

Se realizará en este apartado un análisis de las herramientas y aplicaciones que existen actualmente para la representación de ontologías OWL, así como también las que son utilizadas para realizar transformaciones desde OWL a UML y viceversa.

El objetivo de este análisis es poder conocer el mercado en éste tipo de aplicaciones, a fin de poder comprender mejor las necesidades, objetivos, fortalezas y debilidades de estos productos.

Hay que señalar también, que las aplicaciones analizadas pueden no tener el mismo objetivo si nos centramos en los detalles y resultados obtenidos de éstas. Aunque en la mayoría de casos comparten entre sí la transformación OWLtoUML / UMLtoOWL, son generalmente desarrolladas para realizar representaciones o cumplir con ciertas necesidades particulares de cada desarrollador o institución que las ha realizado.

En el caso de la aplicación que se desarrolla en este proyecto, aunque la tarea más destacada es la transformación OWL2UML, las decisiones de diseño y transformación serán influidas por el objetivo final de recuperar la información de la ontología OWL en RSHP, por lo que se intentará ajustar esta transformación a elementos recuperables por éste último lenguaje (RSHP).

A continuación se mostrará un gráfico en el que se recogen algunas de las aplicaciones que se han encontrado en el análisis de algunas publicaciones [48], e investigación de otras organizaciones o personas que han desarrollado herramientas similares [49] [50] [51] [52] para la transformación o representación visual de ontologías OWL.

Se ha decidido establecer una separación de las aplicaciones analizadas de acuerdo a las siguientes características:

- **OWL2UML**: las aplicaciones pertenecientes a este grupo se caracterizan por tener como principal funcionalidad la transformación de ontologías OWL (entrada fichero .OWL) a UML (salida fichero XMI). En algunos casos, se corresponden con módulos que se han desarrollado para incluirse en aplicaciones más grandes o que tienen otras funcionalidades a parte de esta.

- **UML2OWL**: se han incluido en este grupo las aplicaciones que tienen como objetivo principal la transformación de modelos u ontologías representadas en UML a ontologías en lenguaje OWL.

- **Hybrid**: Podemos encontrar algunos casos en que las aplicaciones encontradas en este campo, ofrecen una interfaz en la que el usuario pode diseñar una ontología de forma visual (“estilo UML”, como lo suelen denominar) y exportarlo a formatos como OWL o UML. Además en algunos casos podemos encontrar que algunas aplicaciones permiten importan ontologías OWL haciendo una representación que se ha desarrollado de

forma propia o específica para estos casos, siendo similares a modelos UML pero sin llegar a cumplir en su totalidad las recomendaciones para representación en este lenguaje.

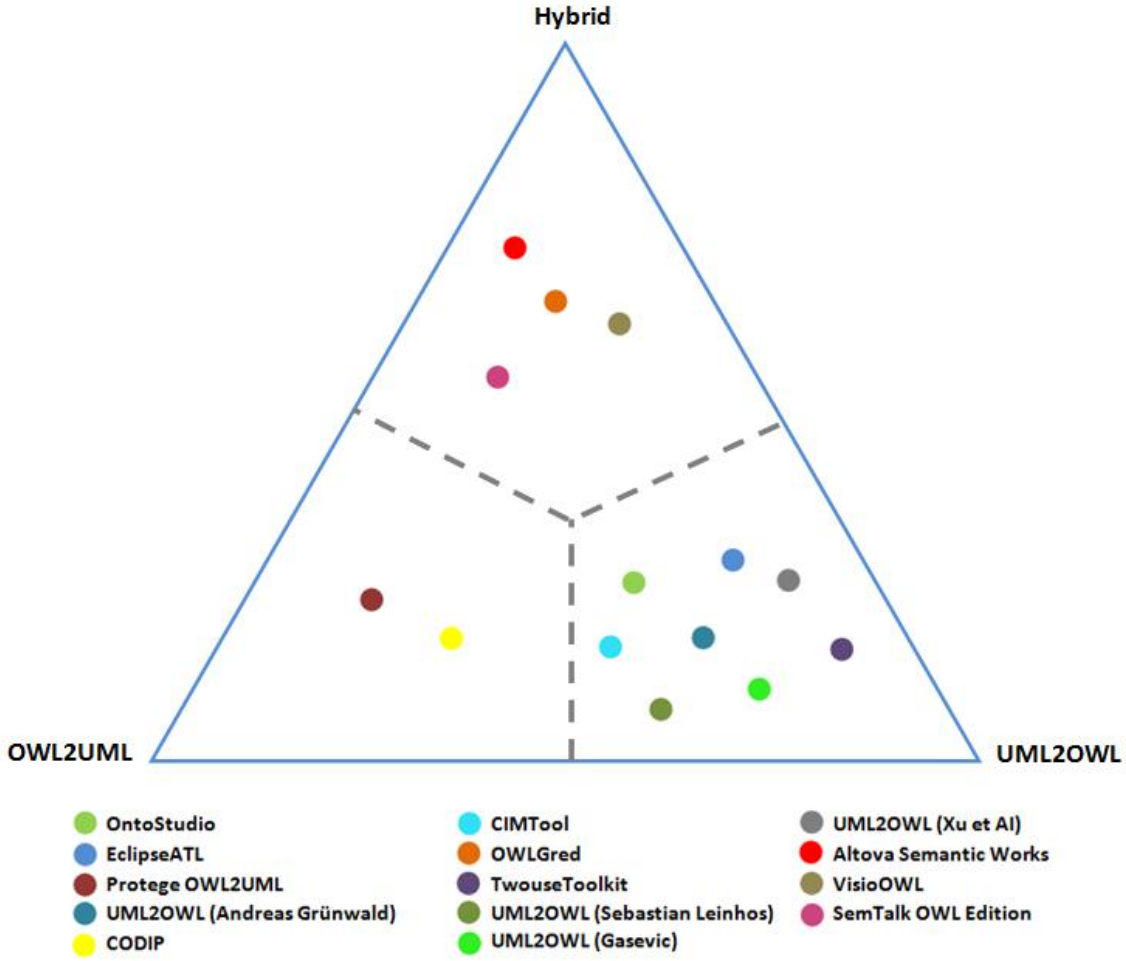


Ilustración 19. Herramientas de transformación OWL similares

2.5.1 Características a considerar sobre las aplicaciones del mercado

Con el objetivo de tener una comparación lo más objetiva y uniforme sobre las aplicaciones que se han encontrado y que son objeto de análisis en este documento, se establecen los siguientes elementos:

- **Tecnología**: se especifica en este caso la tecnología utilizada para desarrollar la aplicación.
- **Plataforma**: se incluye información sobre la plataforma en la que se puede utilizar la aplicación (SO, Web, etc…)

- **Licencia**: se aporta información sobre las licencias necesarias para el uso de la aplicación.
- **Última actualización**: información sobre la última actualización de la aplicación.
- **Soporte**: información sobre el soporte a los usuarios en la utilización de la aplicación.
- **Documentación**: información sobre documentación disponible.
- **Transformación**: información sobre la transformación que ofrece la aplicación.

Se han incluido datos en la siguiente tabla considerando las aplicaciones más relevantes y utilizadas, obteniendo información de algunos autores que han realizado estudios anteriormente [48], así como por la investigación realizada en este mismo proyecto.

	Protege OWL2UML	CODIP	CIMTool	OWLGrEd	Eclipse ATL	UML2OWL (Gasevic)	UML2OWL (Xu et al)
Tecnología	JAVA	JAVA	Eclipse	JAVA/JSON	Eclipse IDE	Xalan-Java /XSLT	Java
Plataforma	Windows Linux	-	Windows	Web / Windows	-	Protege/ XSLT	-
Licencia	Mozilla Public License	BSD[5]	GPL2.1[6]	Académico gratis. Profesional con licencia	EPL/CPL[7]	Libre	-
Última actualización	2009	2004	Versión 1.9.7	Version 1.6.1, 2014	2015	2006	-
Soporte	Protege team	Foro sin actividad	Foros CIMTool	Universidad de Latvia	Foros Eclipse	Dragan Gasevic	-
Documentación	Wiki Protege	Documento	Página web	Página web	Sí, ATL Eclipse	Publicaciones página autor	Publicación [49]

[5] Berkeley Software Distribution
[6] GNU lesser General Public Licence
[7] Eclipse Public Licence: https://eclipse.org/legal/eplfaq.php#CPLEPL

Transformación	OWL2UML	OWL2UML, UML2OWL	UML2OWL	OWL2UML (estilo similar UML)[8]	UML2OWL	UML2OWL	UML2OWL

Tabla 5. Comparativa de herramientas de transformación OWL

2.5.2 Descripción de herramientas similares

- **Protege OWL2UML**: este complemento de la aplicación Protege , realiza la transformación de ontologías OWL a modelos UML, según afirman ellos respetando las recomendaciones y normas ofrecidas por OMG. La aplicación está desarrollada en Java y puede utilizarse tanto en plataformas Windows como Linux. Como aspectos positivos a destacar, es la vinculación al equipo que mantiene trabaja con la aplicación Protege, una de las más famosas y utilizadas para trabajar con ontologías. Este proyecto ha estado vinculado a OWLGrEd, de la Universidad de Latvia, con los que el equipo de desarrolladores de Protege ha trabajado conjuntamente en algunas de las actividades como hacen referencia en su página[9].

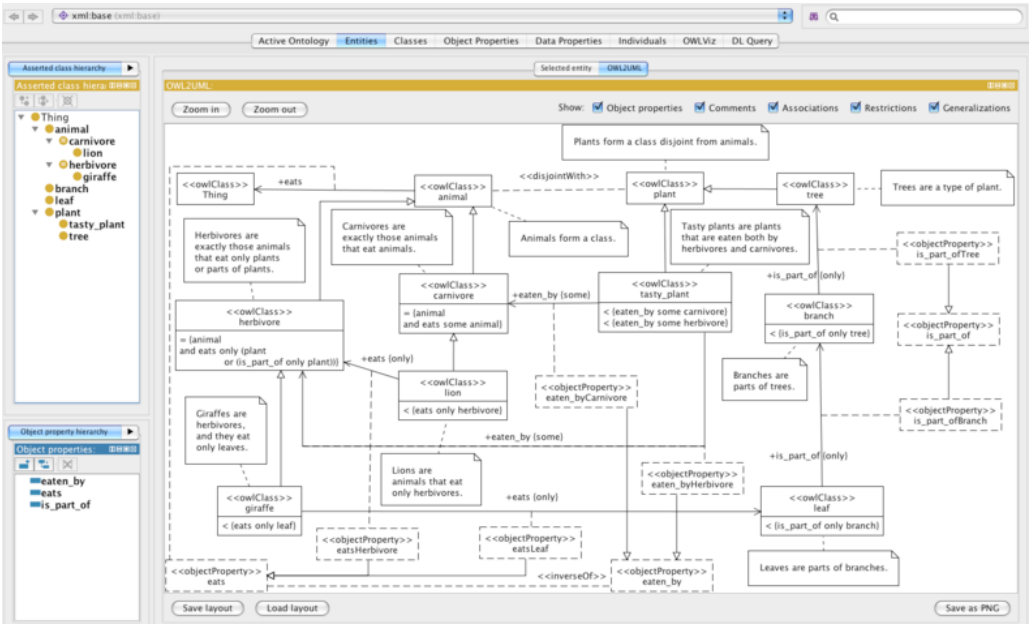


Ilustración 20. Captura de pantalla OWL2UML en Protege

[8] Los creadores especifican que el diseño es estilo UML, pero sin llegar a cumplir con las recomendaciones OMG

[9] OWL2UML, Protege: http://protegewiki.stanford.edu/wiki/OWL2UML

- **CODIP**: esta herramienta (o conjunto de herramientas), pertenecen a SemWebCentral, un sitio web que ofrece productos de software libre para satisfacer necesidades en el mundo de la Web Semántica. CODIP, la herramienta estudiada en este caso, ofrece módulos desarrollados en Java a para procesar archivos OWL y UML, y realizar transformaciones entre estos lenguajes. Según explican en su página web, las transformaciones que involucran modelos UML, siguen las recomendaciones de OMG.

- **CIMTOOL (Common Information Model Tool)**: esta herramienta, fue diseñada en un principio para la representación de conocimiento en el ámbito de la industria eléctrica, pero se ha incluido también en otros usos y dominios. Es distribuida como software libre y tal como describen en su página web, se pueden leer archivos XMI (UML), verificar modelos UML y convertirlos a OWL. Se puede encontrar en su página abundante información sobre la herramienta y comparación de algunos elementos OWL y UML. A pesar de esto, no existen transformaciones que impliquen cierta complejidad, siendo sólo disponibles elementos más o menos sencillos de la transformación UML2OWL. Algún autor ha notificado fallos en el funcionamiento de la herramienta al importar archivos XMI 2.1.

- **OWLGrEd**: Esta herramienta resulta ser una de las más recientes en el ámbito de la representación y transformación de ontologías OWL/UML. Siendo responsable de su desarrollo y soporte la Universidad de Latvia, (Institute of Mathematics and Computer Science), también aparecen como colaboradores el equipo de desarrollo de Protege, de la Universidad de Standford. Tal como especifican los responsables de OWLGrEd, los modelos conseguidos al procesar una ontología OWL, resultan ser de “estilo UML”, pero sin llegar a cumplir las recomendaciones de OMG. Como se puede apreciar en las representaciones visuales obtenidas en la herramienta, algunos elementos difieren del formato estricto definido en UML.

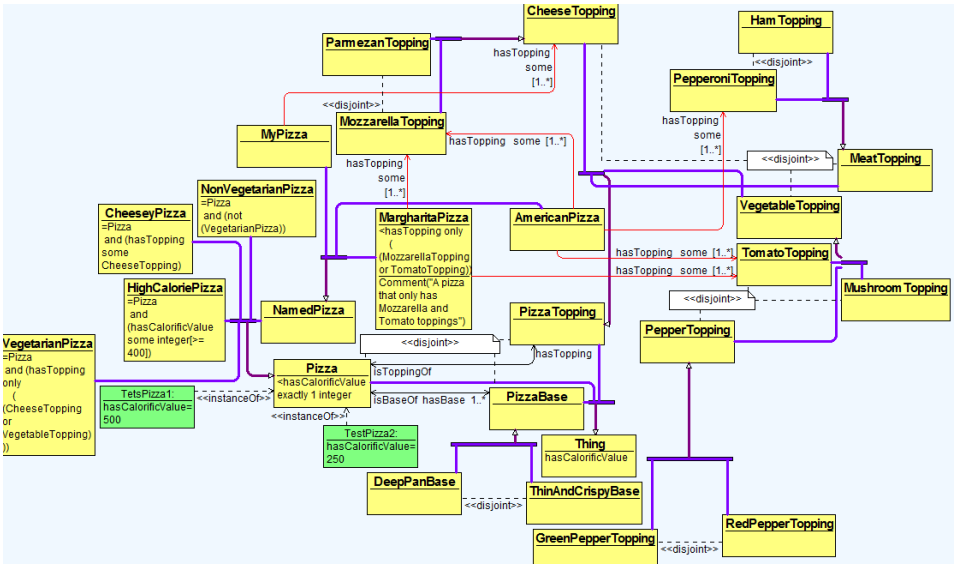


Ilustración 21. Captura de pantalla de visualización de ontologías en OWLGrEd

- **Eclipse ATL**: Según refieren en su página web, esta herramienta proporciona la transformación de modelos UML definidos de acuerdo al ODM de OMG a OWL, siendo compatible con UML 2.0. La herramienta provee una transformación apoyada en MOF (Meta Object Facility) y UML. Es posible recuperar como salida en la aplicación ficheros que cumplen el formato y estructura definidos para OWL/XML. No es posible la transformación inversa a la expuesta.

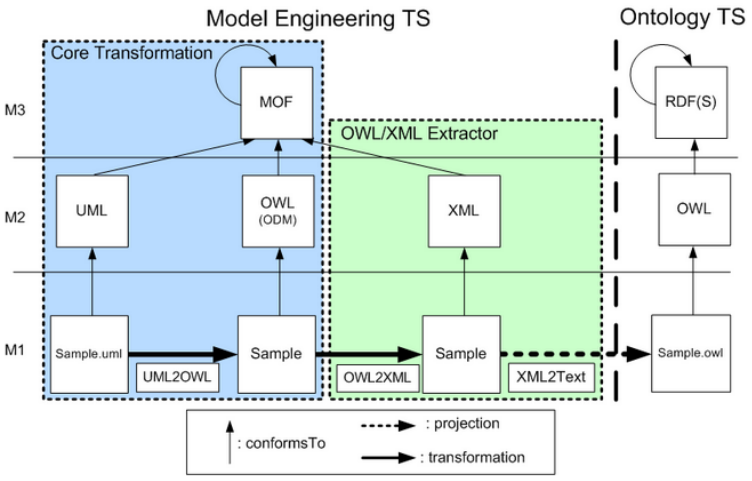


Ilustración 22. Esquema del sistema ATL Eclipse

- **UML2OWL (Gasevic)**: Esta herramienta desarrollada por Dragan Gasevic [53], proporciona la transformación de modelos UML (OUP, Ontology UML Profile) que están en formato XMI a ontologías OWL, que como ha sido explicado en este documento se corresponden con formato XML. La aplicación está basada en XSLT (eXtensible Stylesheet Language Transformation), que realiza la transformación del modelo UML (OUP) a un archivo en formato XML que contiene la ontología transformada. Como aspecto menos favorable, es la necesidad de que el modelo UML cumpla con las condiciones de UML (OUP) para poder ser procesado.

- **UML2OWL (Xu et al)**: El autor de este trabajo, Xu et al, nos describe la herramienta desarrollada [49] para la transformación de ontologías UML a OWL. La aplicación ha sido desarrollada utilizando Java, utilizando la DOM (Document Object Model) API para Java, para el procesamiento previo de los nodos del archivo XMI, antes de la utilización de las reglas correspondientes para la transformación del lenguaje. El autor hace crítica a las aplicaciones de transformación a las herramientas con el mismo fin que utilizan XSLT, por no considerarlo adecuado para ontologías que contienen elementos complejos).

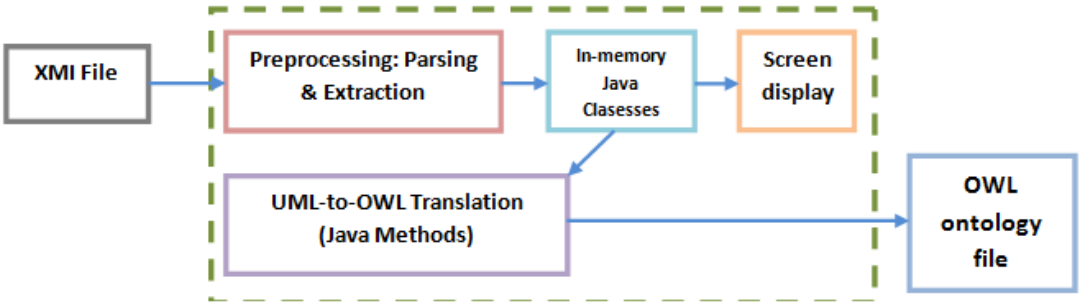


Ilustración 23. Esquema de funcionamiento de la aplicación de Xu et al

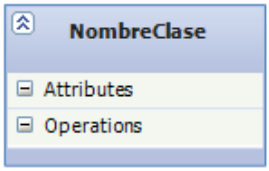
2.6 Unified Modeling Language, UML

El lenguaje unificado de modelado, conocido como UML, es utilizado para especificar, visualizar y documentar modelos de sistemas software, incluyendo su estructura y diseño. Existen varias herramientas en el mercado que son utilizadas para el modelado utilizando UML. Es posible utilizando estas herramientas analizar los requisitos de las aplicaciones así como diseñar la solución cumpliendo dichos requisitos.

Es utilizado para la programación orientada a objetos, pudiéndose utilizar lenguajes como C++, Java o C# además de otras.

A la hora de describir los elementos fundamentales de UML, tendremos en cuenta en este proyecto los diagramas de clases, que serán los utilizados para la transformación de OWL a RSHP.

Se muestran a continuación los elementos UML más importantes de los diagramas de clases:

<table>
<tr><th colspan="2">Clase</th></tr>
<tr><th>Representación</th><th>Descripción</th></tr>
<tr><td> 
</td><td>- Las clases son unidades básicas que encapsulan toda la información de un objeto (instancia). Utilizando clases podemos modelar el problema a tratar, considerando también las clases como conceptos utilizados en el dominio que se está estudiando.
- La descripción visual está dividida en tres zonas o partes, en las que se tiene: Nombre de la clase, atributos y operaciones.
- Los atributos son características que tiene una clase.
- Los métodos son la forma en que la clase interactúa con el entorno

Los atributos y operaciones pueden clasificarse en los siguientes grupos:

➢ public (+): El método o atributo será visible tanto desde dentro como desde fuera de la clase.

➢ private (-): El método sólo será accesible desde dentro de la clase (sólo los métodos de la propia clase pueden acceder).

➢ protected (#): Indica en este caso que los métodos o atributos sólo podrán utilizarse desde la propia clase, y las clases que sean subclases de ella.</td></tr>
</table>

Tabla 6. Elemento clase en UML

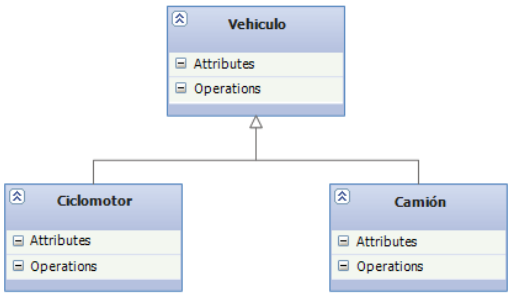
Herencia: Generalización y especialización	
Representación	**Descripción**
	- Mediante la herencia, las subclases contienen los mismos atributos y métodos que la superclase o clase base (siempre que sean "public" o "protected"), además de los atributos y métodos propios.

Tabla 7. Elemento generalización en UML

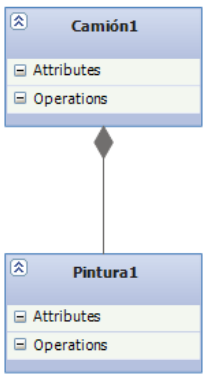
Composición	
Representación	**Descripción**
	- La composición representa una relación estática o fuerte donde el tiempo de vida del objeto incluido está definido por el tiempo de vida del objeto que lo incluye, es decir, que el objeto incluido no puede existir sin el objeto que le incluye.

Tabla 8. Composición en UML

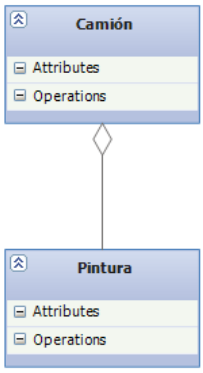
Agregación	
Representación	**Descripción**
	- La agregación representa una relación dinámica o débil, donde el tiempo de vida del objeto incluido no depende del tiempo de vida del objeto que lo incluye, es decir, que el objeto incluido puede existir sin el objeto que lo incluye.

Tabla 9. Agregación en UML

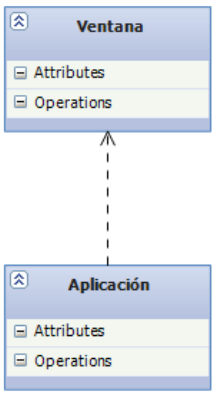
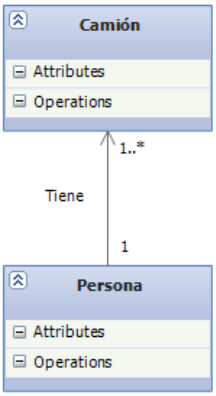
Dependencia o instanciación	
Representación	**Descripción**
	- Es un tipo de relación en el que una clase es instanciada (la instanciación depende de otro objeto). Por ejemplo en este caso, la instanciación del objeto ventana depende de la instanciación desde el objeto Aplicación.

Tabla 10. Dependencia en UML

<table>
<tr><th colspan="2">Asociación</th></tr>
<tr><th>Representación</th><th>Descripción</th></tr>
<tr><td> 
</td><td>

Este tipo de relación permite asociar objetos que colaboran entre sí. No es una relación fuerte, es decir, que la existencia de un objeto no depende del otro.

Podemos encontrar en las asociaciones los siguientes elementos:

Rol: Identifica con nombres los extremos que aparecen en la línea de relación, siendo este nombre la semántica que tiene la relación en el sentido que sea indicado.
Multiplicidad: Este elemento indica cuantos objetos de cada tipo intervienen en la relación en cada uno de los extremos. Podemos encontrar las siguientes multiplicidades: (1), (0..1), (0..*) de cero a muchos/varios, (1..*) de uno a muchos/varios. Mediante la utilización de la multiplicidad podemos considerar que:

Cada asociación tiene dos multiplicidades.
Es necesario especificar la multiplicidad mínima y máxima.
Si la multiplicidad mínima es 0, la relación sería opcional.
Si la multiplicidad mínima es 1 la relación es obligatoria.

</td></tr>
</table>

Tabla 11. Asociación en UML

2.7 RSHP (Relationship Language)

El modelo relacional de representación de información, RSHP, tiene como objeto la representación de un dominio, basándose esencialmente en la relación entre hechos.

Es necesario tener en cuenta a la hora de trabajar con RSHP, la siguiente definición [66] : “El primer elemento descriptivo que debe ser encontrado dentro de un artefacto debe ser la relación. Esta relación es la encargada de enlazar elementos de información”.

Podemos representar mediante RSHP otros lenguajes o herramientas de modelado:

- El modelado relacional está basado en la relación entre entidades, es por tanto aplicable la utilización de RSHP a la hora de representar la información basándonos en relaciones.
- Si tenemos en cuenta el lenguaje UML (Unified Model Language), sabemos que éste lenguaje de modelado, nos permite la relación de los elementos del modelo.
- En la orientación a objetos encontramos también relaciones de herencia, asociación, generalización (también muy estrechamente ligado con UML).
- Si tenemos en cuenta el texto libre, podríamos realizar relaciones entre términos.

Esta característica común que observamos en los casos anteriores, nos permite identificar RSHP como un buen recurso para tratar la información que podamos encontrar en ellos, y en definitiva reutilizarla.

Un elemento de información (IE), es un componente atómico de información que está enlazado por una o más relaciones con otros elementos de información, para formar información. En RSHP los elementos de información pueden representar cualquier artefacto.

A la hora de representar un artefacto (contenedor de información), podemos seguir la siguiente estructura:

Artefacto

Los artefactos son considerados contenedores de información. Los artefactos pueden contener relaciones, metapropiedades y metadatos. En cuanto a las restricciones a la hora de definir un *artefacto* se debe tener en cuenta que una propiedad debe ser metapropiedad o metadatos (mediante LXor), aunque pueden existir ambas propiedades dentro de un artefacto.

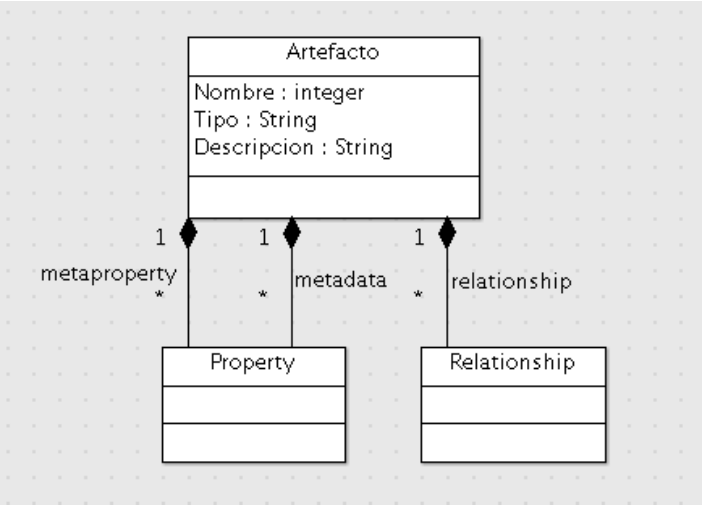


Ilustración 24. Artefacto RSHP

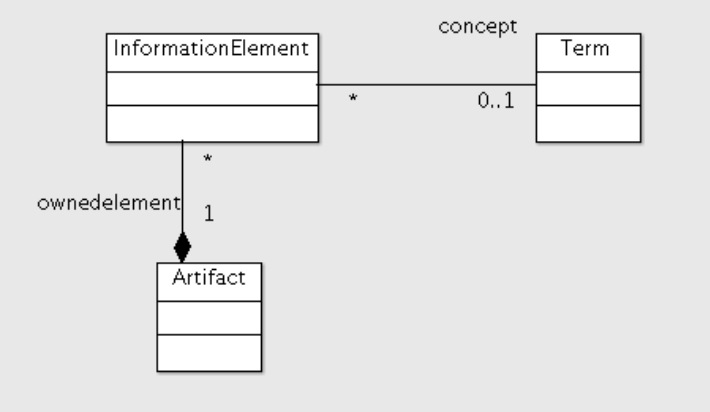


Ilustración 25. Representación por términos

Términos

Los términos representan el vocabulario del dominio. Un término se debe corresponder con una representación única del lenguaje humano de un concepto semántico. Se entiende también, que un término es una representación normalizada de un concepto dado, es decir que sólo debe usarse una forma

léxico-sintáctica. Los términos estáticos (nombres) y dinámicos (verbos) deben ser almacenados y modelados, extrayendo la relación con el término verbo.

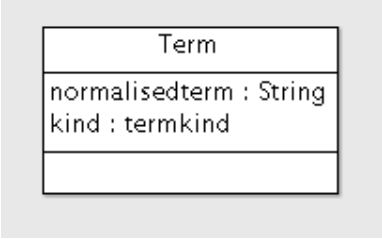


Ilustración 26. Termino en RSHP

Relaciones

Las relaciones son el elemento esencial en el modelo RSHP. Las relaciones son usadas como descriptores de la información de todos los tipos de artefactos, estando compuestas por las distintas ocurrencias de los Elementos de Información. Dichas ocurrencias pueden tener orden, sobre todo cuando las relaciones no son simétricas. El modelo RSHP puede tener elementos de información (IE) que den nombre a la acción dinámica de las relaciones.

Cada una de las relaciones debe tener asignado también un tipo, ya que los algoritmos de indexación o recuperación de la información son independientes de las relaciones que encuentren entre los artefactos.

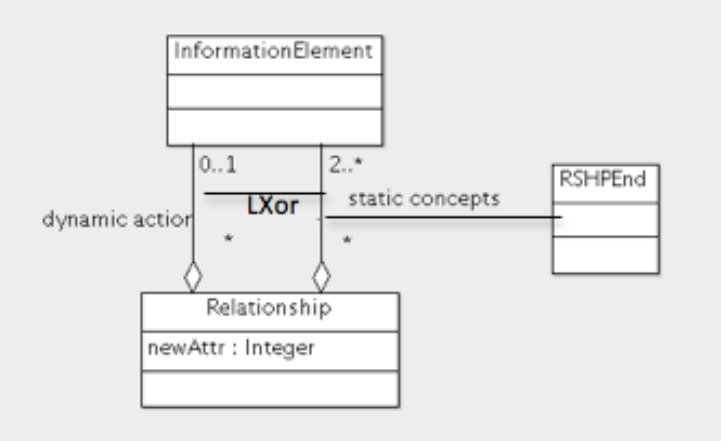


Ilustración 27. estructura de relaciones en RSHP

A continuación se describen los distintos tipos de relaciones que podemos **encontrar** en RSHP:

- **RSHP 1**: Esta relación se denomina usa, ya que un cambio en alguna de las especificaciones de los elementos puede afectar a otros que también lo usan. Están descritas en UML baja la especificación de:
    - Relación de independencia de instanciación (instancia de)
    - Relación de independencia de traza (principio de trazabilidad)
    - Relación de dependencia de uso (Usa a).
- **RSHP 2:** Gen – Spec: relaciones de generalización especificación. Descritas en UML bajo la especificación de:
    - **Gramatical Gen-Espe**
    - **Semantical Gen-Espe**
- **RSHP 3:** Asociación. Es una relación estructural que especifica qué instancias de un artefacto están conectadas a instancias, es decir objetos, de otro artefacto. Se recogen en las siguientes categorías:
    - Asociación semántica-estructurales: cualquier asociación semántica
    - Agregación: Todo – parte.
    - Composición: compuesto por.
- **RSHP 4: Realización**: entre un artefacto que proporciona un contrato y otro artefacto que lo proporciona/garantiza.
- **RSHP 5:** Cualificación. Basado en el paradigma atributo – valor, cubre los atributos que caracterizan la caracterización de un artefacto. No está contemplado en este caso en UML.
- **RSHP 6:** Equivalencia. Relación de sinonimia, equivalencia entre dos conceptos. No contemplada en UML.
- **RSHP 7:** Idiomática. traducción de un descriptor representando un descriptor en otros idiomas.
- **RSHP 8:** Lingüística. Relaciones que surgen de las acciones verbales en los documentos de texto y que no pertenecen a ninguno de los tipos anteriores.

Elemento de información

El elemento de información se considera la unidad mínima de información en RSHP. El elemento de información (IE) representa la ocurrencia de un término dentro de un artefacto, como también un artefacto (contenedor de información). Un elemento de información puede formar parte de una relación o de una propiedad, y opcionalmente puede tener una posición dentro de un artefacto.

Propiedad

Las propiedades modelan la representación de metadatos acerca del artefacto. Cada propiedad debe tener un IE que dé nombre a la etiqueta de la propiedad, siendo posible tener múltiples valores de etiquetas.

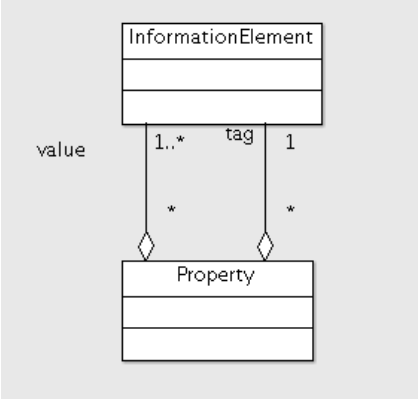


Ilustración 28. Estructura de propiedades en RSHP

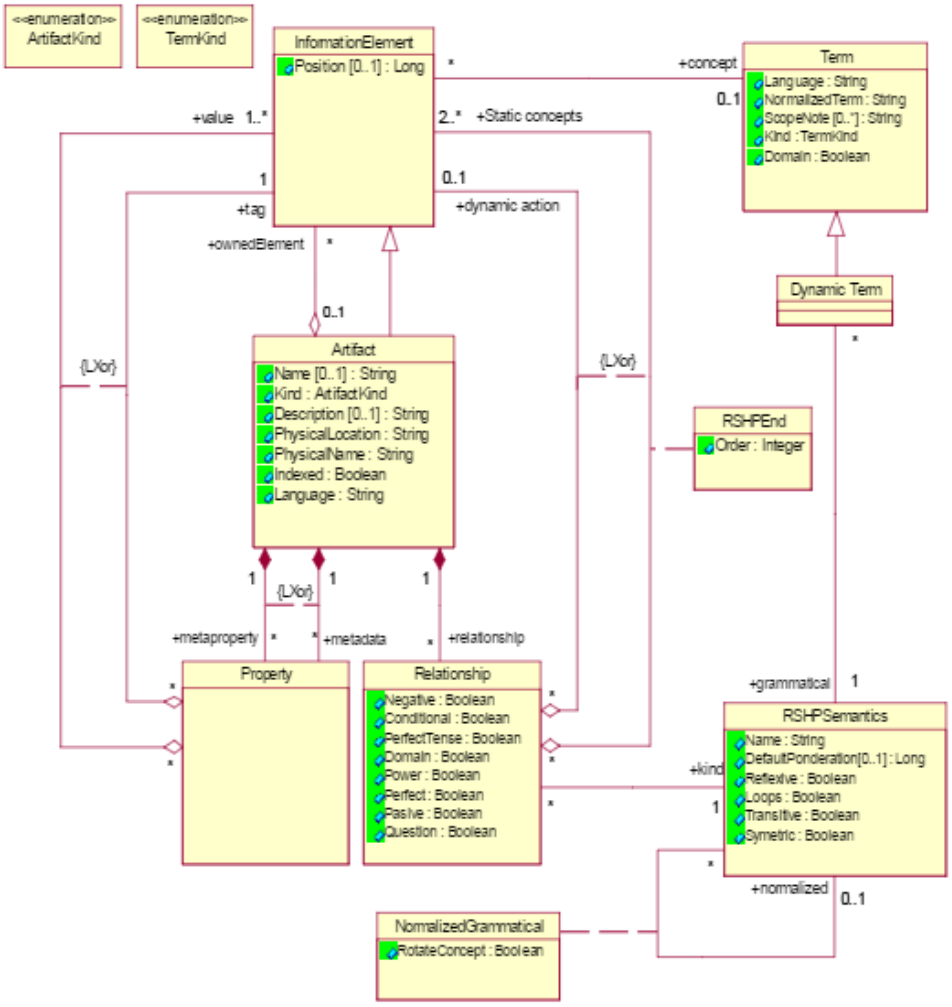


Ilustración 29. Modelo de representación RSHP [66]

2.8 UMLModels

El proyecto actual representa un pequeño módulo de un proyecto más grande y complejo, que es UMLModels.

Esta herramienta, fue concebida para poder gestionar la documentación relacionada con ingeniería del software desde un mismo lugar, habiéndose desarrollado un sitio web para su utilización.

UMLModels se utiliza mediante un buscador, desde el cual los miembros o interesados en el campo de la ingeniería del software pueden obtener desde este buscador resultados que aporten una mayor semántica (y por tanto más significado) así como recursos que pueden ser útiles para la realización de proyectos software.

Esta herramienta, utilizando la información recogida por “crawlers”, nos permite recuperar información de forma visual y normalizada en modelos UML que posibilita la reutilización de dicha información de una forma más sencilla, al representar la información en un modelo que es el más utilizado para el modelado en ingeniería del. Partiendo de la información a recuperar, este sistema mediante una serie de reglas y aplicaciones integradas en él es capaz de transformar la información de origen en modelos UML, siendo generados en ficheros XMI. Aprovechando estas capacidades y funcionalidades que ofrece UMLModels, podemos considerar que, esta herramienta nos permite recuperar información de distintas fuentes, modelos o lenguajes, reuniéndolos en un formato estandarizado y ampliamente conocido en el campo de la ingeniería del software, como es UML. Esta homogenización que se consigue con UMLModels, facilita la posterior reutilización de la información por otros sistemas, a fin de crear nuevos modelos, generar código, extraer conocimiento, o realizar cualquier otro tipo de tarea donde pueda reutilizarse código XMI.

Además de esto, UMLModels permite la representación gráfica de los modelos en formato XMI, permitiendo una rápida comprensión y visualización de dichos modelos.

El usuario puede configurar el origen de datos que quiere utilizar, pudiendo ser ésta, información disponible en Access, imágenes, XML u ontologías OWL como es el objetivo de este proyecto. Por tanto, el proyecto actual, representa una funcionalidad más dentro de UMLModels, desde la que los usuarios podrán recuperar ontologías OWL disponibles en la Web, obteniendo modelos UML.

Podemos identificar los siguientes elementos principales en UMLModels:

- **Crawler**: esta aplicación realiza búsquedas en la Web, extrayendo los elementos que podrán ser tratados por la aplicación UMLModels. Se recogerá información disponible en la Web en formato imagen, BBDD Access, ficheros XML así como ontologías OWL.

- **Indexer**: esta aplicación realiza el almacenamiento de la información recuperada en archivos XMI o RSHP. Interacciona con el módulo a desarrollar en este proyecto aportando las reglas necesarias para la recuperación de ontologías.

- **BBDD Cake**: En esta base de datos se dispone de la información necesaria para la recuperación de la información en RSHP.

3 Estudio de viabilidad del sistema

En este capítulo se realizará el análisis de un conjunto concreto de necesidades para proponer una solución a corto plazo, que tenga en cuenta restricciones económicas, técnicas, legales y operativas. Además se incluirán las decisiones derivadas de la investigación sobre la transformación de elementos OWL en modelos UML.

3.1 Identificación del alcance del sistema

Partiendo de la situación inicial, situación actual, y los requisitos recogidos, se expondrán una serie de alternativas que sean adecuadas a la solución que se necesita.

Para ello, se estudiarán también posibles desarrollos a medida, así como otras posibles opciones que se puedan encontrar ya desarrolladas y que se puedan incorporar al proyecto.

Sobre las alternativas propuestas, se tendrán en cuenta varios aspectos como pueden ser el coste, los riesgos asociados a la alternativa, así como otros indicadores que puedan considerarse útiles para la elección de la alternativa más adecuada para el desarrollo del proyecto.

3.2 Estudio de alternativas de solución

Una vez analizadas las tecnologías que son utilizadas en este proyecto, se recogieron los requisitos de usuario, y se llevaron a cabo las tareas de análisis correspondientes para valorar el alcance de la aplicación que se quiere desarrollar.

Teniendo en cuenta los datos recogidos, y el análisis realizado sobre los mismos se incluye a continuación una valoración de alternativas sobre el entorno tecnológico y el software necesario para desarrollar la aplicación.

Debemos tener en cuenta también, que esta aplicación formará parte del proyecto UMLModels, el cual tiene como objetivo la recuperación de información desde distintos lenguajes en el lenguaje de modelado unificado (UML). Por ello, se valorarán las alternativas desde el punto de vista de compatibilidad y similitud de arquitectura del sistema con el resto de librerías que componen UMLModels.

A fin de extraer una decisión lo más objetiva posible sobre el entorno a seleccionar, así como lo más apropiado a las necesidades del cliente, se han definido los siguientes indicadores para valorar la solución a seleccionar:

	Malo	Regular	Bueno	Muy bueno
Experiencia	0	1	2	3
Compatibilidad				
Reutilización				
Coste				
Potencial para el producto				

Tabla 12. Escala de valores a utilizar en evaluación

A cada uno de los indicadores a utilizar se les ha puesto un valor entre cero y tres a fin de valorar las alternativas de acuerdo a su puntuación.

	Experiencia	Reutilización	Coste	Potencial para el producto	Comentarios	Puntuación
Propuesta XX:						
Sistema Operativo	Experiencia en el SO	NA	Coste de licencias	Afinidad con la solución	Comentarios más destacados	Puntuación total obtenida por la solución propuesta
Tecnología de programación	Experiencia en el lenguaje	Utilización de funciones o código en futuros proyectos				
Entorno	Experiencia en el entorno de programacion	NA				
Sistema de BBDD	Experiencia en el SGBD	Reutilización de código BBDD en futuros proyectos				

Tabla 13. Criterios de evaluación

	Experiencia	Reutilización	Coste	Potencial para el producto	Comentarios	Puntuación
Propuesta 1:						
MS Windows 7	2	NA	3	3	Muy buena compatibilidad y reutilización con UMLModels	38
.NET	2	3	3	3		
Visual studio	2	NA	3	3		
SQL Server	2	3	3	3		
Propuesta 2:						
MS Windows 7	2	NA	3	3	Poca experiencia en Java, y poca reutilización en UMLModels	33
Java	1	1	3	3		
Eclipse	1	NA	3	3		
MySQL	2	2	3	3		
Propuesta 3:						
Linux	2	NA	3	2	Lenguaje no apropiado para las necesidades del cliente	31
C	2	1	3	1		
Eclipse	1	NA	3	3		
MySQL	2	2	3	3		

Tabla 14. Resultados de la evaluación

De acuerdo a las propuestas analizadas según los parámetros que consideramos más importantes para el éxito de este proyecto, la más recomendable y por tanto elegida será la **propuesta 1**.

A continuación se realizan comentarios de carácter general para cada una de las opciones, basándonos en aspectos como el coste y riesgos asociados a las alternativas:

- **Propuesta 1**: Es la propuesta finalmente elegida para el desarrollo de la aplicación de este proyecto. Contiene una serie de ventajas que no llegan a ofrecer el resto de propuestas ya que además de contar el programador con más experiencia en ésta tecnología, hay que tener en cuenta también que el resto de la aplicación UMLModels está desarrollada utilizando tecnología .NET. Por otra parte, parte del código desarrollado podría servir para futuras modificaciones de la aplicación UMLModels, mejorando la reutilización del mismo, y facilitando el mantenimiento del la aplicación globalmente (no serían necesarios expertos en varios lenguajes). Por tratarse de una tarea del ámbito académico, la Universidad Carlos III dispone de licencias gratuitas para el desarrollo de la aplicación.

- **Propuesta 2**: Como elemento a favor, estaría la semejanza en cierto modo del lenguaje JAVA y C#, siendo los dos orientados a objetos y con una sintaxis parecida. Sin embargo, el desarrollo de la aplicación en JAVA

implicaría el no poder reutilizar parte del código o funciones que pudiesen ser útiles, sabiendo que la aplicación UMLModels ha sido desarrollada completamente con tecnología .NET. El programador tiene menos experiencia en este lenguaje que en C#. El entorno Eclipse no tendría coste asociado.

- **Propuesta 3**: Considerando un SO libre (sin licencia), se ofrece la alternativa LINUX, pero habría que tener en cuenta la menor experiencia en este SO, así como la utilización del lenguaje C, que no sería apropiado para las necesidades y características del software que necesita el cliente. Con respecto a la licencia y coste de la misa para el resto de opciones, finalmente no sería comparativamente mejor en este caso (siendo libre) considerando que por estar realizando una actividad académica disponemos de la licencia para utilización del entorno MS Visual Studio en otras opciones descritas anteriormente.

Costes asociados a la utilización de las tecnologías y entornos utilizados:

Descripción	Coste (€)
Java	0 €
Eclipse	0 €
MySQL	0 €
MS Windows 7	0 €
MS Visual Studio	0 €
SQL Server	0 €
.NET	0 €

Tabla 15. Coste de herramientas a utilizar

3.3 Análisis de la transformación de elementos OWL a UML

Como objetivo final, la aplicación transformará la información inicial (OWL) en formato RSHP (THE), pero es un paso imprescindible analizar la correspondencia entre OWL y UML, ya que utilizaremos como paso intermedio librerías para la

transformación de la ontología utilizando las reglas correspondientes en cada caso.

En primer lugar, habría que analizar por qué establecer una relación entre OWL y UML. Ambos leguajes han sido utilizados para representación de conocimiento en sus respectivos campos. UML es un lenguaje de modelado ampliamente utilizado en Ingeniería del Software para visualizar, especificar la estructura, comportamiento y arquitectura de un sistema software [23]. Concretamente, nos vamos a centrar en este proyecto en la representación estática de los diagramas de clases. Por otra parte OWL, un lenguaje que depende del W3C, es utilizado en el área de Inteligencia Artificial para la representación de ontologías[10].

Podemos encontrar representaciones visuales de algunos lenguajes de modelado como puede ser UML, y ER (Entity-Rrelationship model). Ambos son utilizados tradicionalmente para descripción conceptual de un sistema software. Otros lenguajes de representación de conocimiento como puede ser Topic Maps, que utiliza una notación bien definida para la representación del conocimiento utilizando grafos.

A la vez que se han ido desarrollando los lenguajes de modelado de manera que sean más legibles para las máquinas [24] [25], y por tanto que puedan ser mejor procesados para cualquier tipo de tarea, también ha sido de gran importancia la representación visual de los mismos, ya que tiene gran interés por parte de los usuarios que los utilizan. A pesar de esto, un lenguaje para definición de ontologías como es OWL, fue utilizado en un principio como un lenguaje expresado de forma textual y basándose en lógica descriptiva, pero sin una representación visual que fuese propia. En 2003[11], la entidad OMG desarrolló ODM (Ontology Definition Metamodel) como una especificación para arquitecturas dirigidas por modelos (Model Driven Arquitecture), con el fin de poder representar ontologías con lenguajes más familiares como puede ser UML y elementos de ER.

Sabemos que en las ontologías definidas mediante OWL disponemos de clases, subclases, relaciones entre clases, y axiomas. Estos elementos son normalmente utilizados en los modelos UML, como son los diagramas de clases, así como en OCL (Object Constraint Language). Varios autores han apoyado la representación de ontologías utilizando UML [24], [26], [27], [28], mostrando interés en establecer similitudes en la representación de ambos lenguajes, así como propuestas para la representación de ontologías OWL en UML.

Podemos encontrar también, como varios autores [24] han dado algunas razones para apoyar la representación de ontologías en UML, como por ejemplo:

- UML es una notación gráfica que tiene bastantes años de recorrido, y es ampliamente utilizada por empresas y organizaciones en todo el mundo.

[10] Se explicarán en otro apartado detalladamente estos dos lenguajes, OWL y UML.
[11] http://www.omg.org/spec/ODM/1.1/

- Es un estándar abierto mantenido por OMG.
- UML consta de extensiones para representación de ontologías.
- UML es utilizado por empresas y universidades, y además las técnicas basadas en representación de conocimiento están basadas en KIF (Knowledge Interchange Format).
- UML puede ser utilizado mediante herramientas tipo CASE, lo amplía el espectro de usuarios que generalmente utiliza estas aplicaciones. Por el contrario podemos encontrar aplicaciones específicas para diseño de ontologías como Protege y Ontolingua.
- Existen agentes que están construidos para procesar información basada en modelos UML.

Además de las razones expuestas anteriormente, podemos en encontrar trabajos previos para la representación de RDF en UML [29], por lo que podemos aprovechar toda la información existente al respecto para diseñar las reglas de transformación necesarias en el trajo en que se centra este proyecto (OWL-UML-RSHP).

Como se ha explicado anteriormente, el objetivo en este trabajo será la recuperación de ontologías OWL en RSHP, pero para ello realizaremos un paso intermedio que será la transformación de la ontología OWL en UML. Aprovecharemos las librerías existentes que nos permiten la transformación UML en RSHP, y nos centraremos en definir la transformación OWL en UML, ya que el segundo paso está actualmente desarrollado dentro del proyecto reúne este mismo.

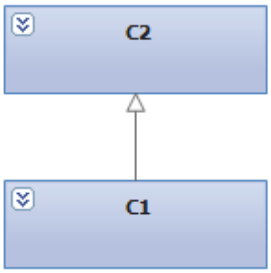
A fin de estructurar cada uno de los elementos a estudiar o transformar, se utilizará la siguiente estructura:

<table>
<tr><th colspan="2">Transformación TXX</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">Elemento OWL</td><td>Representación del elemento OWL en UML</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>Representación mediante lógica de primer orden</td></tr>
<tr><th colspan="2">Descripción</th></tr>
<tr><td colspan="2">Descripción de la transformación realizada en UML.</td></tr>
<tr><th colspan="2">Recuperación de información</th></tr>
<tr><td colspan="2">Transformado en UML : Sí □ No □</td></tr>
</table>

Tabla 16. Ejemplo tabla transformación

<table>
<tr><th colspan="2">Transformación T01</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:Class</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>C</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Tenemos en este caso una transformación directa entre OWL y UML. Este elemento en OWL, no tiene mucho significado en sí mismo si no está relacionado con otros elementos (rdfs:subClassOf, owl:equivalentClass, owl:disjointWith). Podemos ver en el caso de UML, que representa una clase sin ningún otro elemento.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML : Sí ☑ No ☐</td></tr>
</table>

Tabla 17. Transformación owl:Class

<table>
<tr><th colspan="2">Transformación T02</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">rdfs:subClassOf</td><td> 
</td></tr>
<tr><th>Representación lógica</th></tr>
<tr><td>C1 ⊆ C2

(∀x (C1(x) → C2(x))</td></tr>
<tr><th colspan="2">Descripción</th></tr>
<tr><td colspan="2">Teniendo en cuenta la similitud de los elementos, es posible la transformación a UML. En ambos lenguajes, se tiene que al tener una clase C1 que es subclase de la clase C2, las instancias (también llamados generalmente individuals en OWL) de C1 son también instancias de C2. Además, tenemos también en cuenta con este elemento la herencia de la cardinalidad, así como de atributos de la superclase en la representación del modelo UML.</td></tr>
<tr><th colspan="2">Recuperación de información</th></tr>
<tr><td colspan="2">Transformado en UML : Sí ☑ No ☐</td></tr>
</table>

Tabla 18. Transformación rdfs:subClassOf

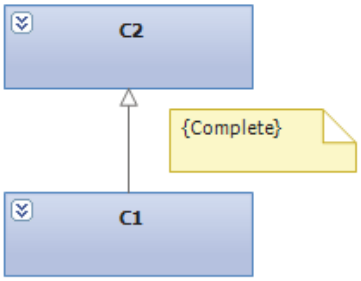
<table>
<tr><th colspan="2">Transformación T03</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">Owl:equivalentClass</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>C1 = C2
∀x (C1(x) → C2(x) ∧ C2(x) → C1(x))</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Lo que se quiere expresar en OWL con esta propiedad, es que las clases definidas como equivalentes tienen el mismo conjunto de instancias. Aunque lo que se quiere expresar en OWL es cierta equivalencia entre clases, no hay que confundirlo como una equivalencia exacta (mismos conceptos [30]). Algún autor ha considerado [27] que esta transformación podría realizarse como una generalización entre C1,C2 , añadiendo un comentario de tipo {Complete} . Aún siendo posible esta solución, en este proyecto se ha decidido realizarlo mediante una doble generalización entre C1,C2, es decir, una generalización de C2 sobre C1 y viceversa. Esta posición ha sido también apoyada por OMG, y ha sido determinante para elegirlo que esta transformación sería recuperada en UML y RSHP, mientras la primera sólo lo sería en UML.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML : Sí ☑ No ☐</td></tr>
</table>

Tabla 19. Transformación owl:equivalentClass

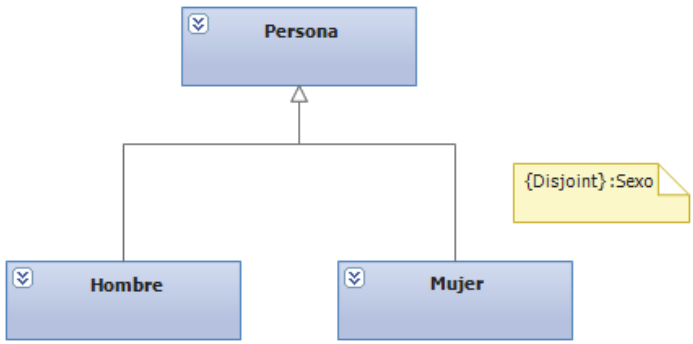
Transformación T04	
OWL	UML
owl:disjointWith	
	Representación lógica
	$C1 \cap C2 = \emptyset$ $\forall x\,(C1(x) \rightarrow \neg C2(x)) = \neg(\exists x\,(C1(x) \rightarrow C2(x)))$
Descripción	
Teniendo en cuenta que "owl:disjointWith" es una propiedad que indica que no existen instancias en la clase C1 que se puedan encontrar en la clase C2, y viceversa, se ha considerado que este caso que se puede representar en UML mediante una generalización, y un comentario sobre este elemento del tipo {Disjoint}. Por tanto, este elemento sí sería recuperable en UML, pero no sería posible en RSHP ya que los comentarios no son considerados en la transformación.	
Recuperación de información	
Transformado en UML: Sí ☑ No ☐	

Tabla 20. Transformación owl:disjointWith

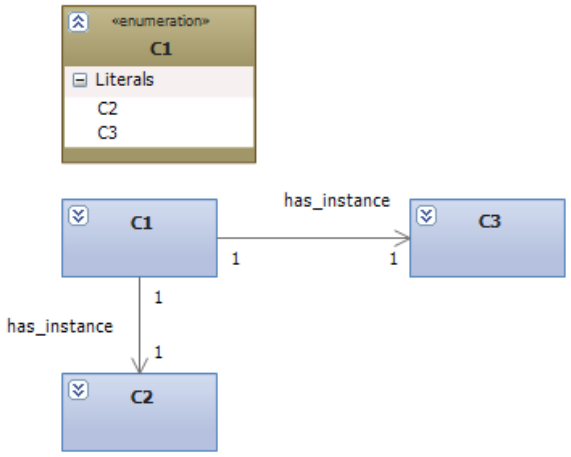
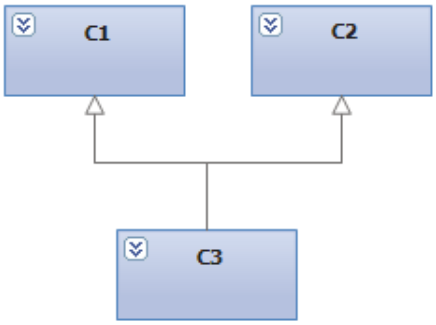
Transformación T05	
OWL	UML
owl:oneOf	
	Representación lógica
Descripción	
Utilizando esta propiedad, se realiza una enumeración de las instancias de la clase correspondiente. Hay que destacar, que debido a las propias restricciones que tenemos para la transformación del modelo UML a RSHP mediante CAKE, se ha considerado que las instancias serán representadas en el modelo como clases, y tendrán “una relación de instancia” con su superclase.	
Recuperación de información	
Transformado en UML : Sí ☑ No ☐	

Tabla 21. Transformación owl:oneOf

<table>
<tr><th colspan="2">Transformación T06</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:intersectionOf</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>C3 = C1 ∩ C2

∀x ((C3(x) → C1(x) ∧ C2(x)) ∧ (C1(x) ∧ C2(x) → C3(x)))</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">En esta propiedad enlaza una clase con una lista de descripción de clases. Esta clase definida, tendrá exactamente las instancias que son instancias también de la descripción de clases indicada. En UML, podemos representar este caso mediante herencia múltiple teniendo en cuenta que las instancias de la clase definida (C3), son instancias a la vez de C1,C2. Además, podemos encontrar distintos casos a la hora de definir una intersección y que trataríamos en este caso [30], [31]:

• Lista/Enumeración de instancias mediante owl:oneOf: En este caso se resuelve seleccionando sólo las instancias que coincidan en todas las enumeraciones incluidas en la definición.

• Combinación de subclases (rdfs:subClassOf), con elementos de relación tipo owl:objectProperty: La transformación en UML se realiza en este caso, incluyendo la generalización correspondiente y la relación a la instancia indicada.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☑ No ☐</td></tr>
</table>

Tabla 22. Transformación owl:intersectionOf

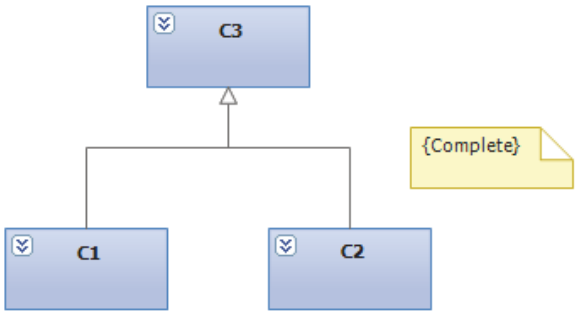
Transformación T07	
OWL	UML
owl:unionOf	
	Representación lógica
	$C3 = C1 \cup C2$ $\forall x ((C3(x) \rightarrow C1(x) \vee C2(x)) \wedge (C1(x) \vee C2(x) \rightarrow C3(x)))$
Descripción	
Teniendo en cuenta que la propiedad owl:unionOf enlaza la definición de una clase con una lista de descripciones. Además, hay que tener en cuenta que esta descripción indica que la clase definida tiene las instancias que al menos ocurren en una de las descripciones incluidas [31] [32]. Al realizar la transformación a UML, se tiene que podemos realizarlo mediante una generalización con la clase definida como superclase, y las clases incluidas en la descripción como subclases, además añadiendo un comentario de tipo {Complete} en el elemento de generalización [27] [29]. Aunque mediante esta solución, es posible la transformación a UML, no es posible recuperarlo en RSHP, debido a que no son procesados los comentarios.	
Recuperación de información	
Transformado en UML: Sí ☑ No ☐	

Tabla 23. Transformación owl:unionOf

<table>
<tr><th colspan="2">Transformación T08</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:complementOf</td><td></td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$C1 = \neg C2$
$IE(C1) = \Delta - IE(B)$</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad establece la relación entre dos clases, para las cuales, se describe los objetos complementarios u opuestos de la clase descrita.
No es posible representarlo en UML/RSHP.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☐ No ☑</td></tr>
</table>

Tabla 24. Transformación owl:complemntOf

Transformación T09	
OWL	UML
owl:hasValue	
	Representación lógica
	$\forall x\ (C1(x) \rightarrow \exists y(R(x,y) \wedge y = V))$
Descripción	
Esta propiedad enlaza una clase a un valor V (instancia o tipo de dato) [31]. Teniendo en cuenta que en el desarrollo de este proyecto se ha tenido en cuenta que las instancias serán representadas como clases y relación de instancia, podemos representarlo en un diagrama de clases como una relación con cardinalidad exacta de 1 [32], cuando esta restricción esté aplicada sobre un objectProperty.	
Recuperación de información	
Transformado en UML: Sí ☑ No ☐	

Tabla 25. Transofrmación owl:hasValue

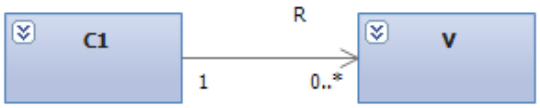
Transformación T10	
OWL	UML
owl:allValuesFrom	
	Representación lógica
Descripción	
Esta propiedad representa una restricción de una clase hacia una instancia o tipo de dato. Todas las instancias que estén incluidas en una clase que tenga esta restricción, tienen que estar relacionadas con alguna de las instancias o datos del rango representado en la descripción. Además según podemos ver en alguna publicación [27], podríamos asumir en este caso que en una relación R entre dos clases C1, C2, con una cardinalidad [0..*] estamos representando la misma semántica que en el caso de OWL.	
Recuperación de información	
Transformado en UML: Sí ☑ No ☐	

Tabla 26. Transformación owl:allValuesFrom

<table>
<tr><th colspan="2">Transformación T10</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:someValuesFrom</td><td></td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td></td></tr>
<tr><th colspan="2">Descripción</th></tr>
<tr><td colspan="2">Esta propiedad representa una restricción sobre una clase en una instancia o dato. Esta restricción determina que, todas las instancias de una determinada clase tendrán al menos un valor del dato o instancia indicado en el rango de la relación [31].</td></tr>
<tr><th colspan="2">Recuperación de información</th></tr>
<tr><td colspan="2">Recuperado en UML: Sí ☐ No ☑</td></tr>
</table>

Tabla 27. Transformación owl:someValuesFrom

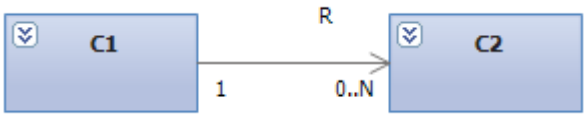
<table>
<tr><th colspan="2">Transformación T11</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:maxCardinality</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$\forall x\,(A(x) \rightarrow |\{y\,|(R(x,y)\}| \leq N)$</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Como se ha explicado anteriormente, esta propiedad representa una restricción de cardinalidad máxima y determina que para todas las instancias de una clase se tiene como máximo N valores semánticamente distintos (instancias o valores). Esta transformación se realizará añadiendo la cardinalidad máxima correspondiente del rango de la relación OWL a la asociación UML. Aplicando también los requisitos sobre herencia, tendremos la herencia de la cardinalidad en los casos en que exista en alguna superclase de la clase a tratar, siempre que no se defina cardinalidad explícitamente sobre la clase.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☑ No ☐</td></tr>
</table>

Tabla 28. Transformación owl:maxCardinality

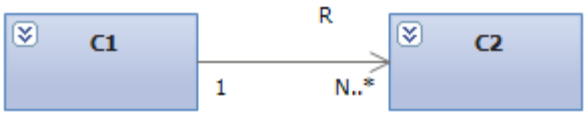
Transformación T12	
OWL	UML
owl:minCardinality	
	Representación lógica
	$\forall x\ (A(x) \rightarrow \lvert\{y \mid (R(x,y)\}\rvert \geq N)$
Descripción	
Como se ha explicado anteriormente, esta propiedad representa una restricción de cardinalidad mínima y determina que para todas las instancias de una clase se tiene como mínimo N valores semánticamente distintos (instancias o valores). Esta transformación se realizará añadiendo la cardinalidad mínima correspondiente del rango de la relación OWL a la asociación UML. Aplicando también los requisitos sobre herencia, tendremos la herencia de la cardinalidad en los casos en que exista en alguna superclase de la clase a tratar, siempre que no se defina cardinalidad explícitamente sobre la clase.	
Recuperación de información	
Transformado en UML: Sí ☑ No ☐	

Tabla 29. Transformación owl:minCardinality

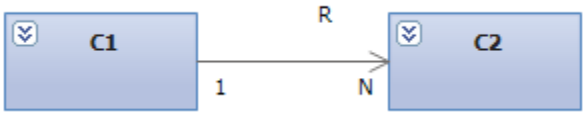
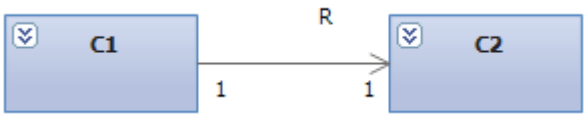
<table>
<tr><th colspan="2">Transformación T13</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:cardinality</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$\forall x\,(A(x) \rightarrow |\{y\,|(R(x,y)\}| = N)$</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Como se ha explicado anteriormente, esta propiedad representa una restricción de cardinalidad exacta y determina que para todas las instancias de una clase se tiene como mínimo N valores semánticamente distintos (instancias o valores). Esta transformación se realizará añadiendo la cardinalidad exacta correspondiente del rango de la relación OWL a la asociación UML. Aplicando también los requisitos sobre herencia, tendremos la herencia de la cardinalidad en los casos en que exista en alguna superclase de la clase a tratar, siempre que no se defina cardinalidad explícitamente sobre la clase.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☑ No ☐</td></tr>
</table>

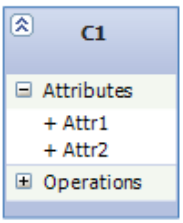
Tabla 30. Transformación owl:cardinality

<table>
<tr><th colspan="2">Transformación T14</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:ObjectProperty</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>R ⊆ OD x OD

(Restricción de relación entre instancias)</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">La propiedad owl:ObjectProperty puede ser definida como un axioma con la única intención de declarar cierta propiedad en una ontología, pero ese caso no se tratará ya que no se puede llevar a cabo. Nos centraremos en las propiedades ObjectProperty que son definidas con un dominio y un rango (relacionan instancias entre sí). A la hora de hacer la transformación a un modelo UML, esta propiedad será fundamental ya que podemos obtener a partir de ella información relevante sobre otras propiedades [12]que puedan afectar a la relación. Aunque la solución más adecuada con respecto a una transformación a UML, sería una relación unidireccional especificando dominio y rango, se adaptará esta solución a las restricciones de transformación que existen en las librerías nUML, que sólo admiten asociaciones (bidireccionales). Por ello, a fin de especificar siempre el rango se incluirá la cardinalidad (cuando exista, y cuando no, sería 1) en el rango de la relación.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☑ No ☐</td></tr>
</table>

Tabla 31. Transformación owl:ObjectProperty

[12] subPropertyOf, equivalentProperty, inverseOf, FunctionalProperty, InverseFunctionalProperty, TransitiveProperty, SymmetricProperty

<table>
<tr><th colspan="2">Transformación T15</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:DatatypeProperty</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>R ⊆ OD x DD

(Restricción de relación entre instancia y un dato o valor)</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">La propiedad owl: DatatypeProperty puede ser definida como un axioma con la única intención de declarar cierta propiedad en una ontología, pero ese caso no se tratará ya que no se puede llevar a cabo. Nos centraremos en las propiedades ObjectProperty que son definidas con un dominio y un rango (relacionan instancias entre sí). A la hora de hacer la transformación a un modelo UML, esta propiedad será fundamental ya que podemos obtener a partir de ella información relevante sobre otras propiedades [13]que puedan afectar a la relación</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☑ No ☐</td></tr>
</table>

Tabla 32. Transformación owl:DatatypeProperty

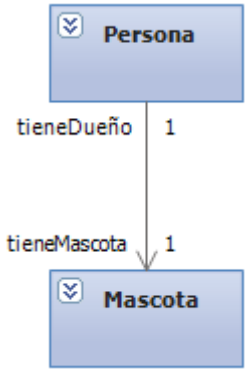
<table>
<tr><th colspan="2">Transformación T16</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:inverseOf</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$\forall x,y,a,b\ (R(x, y) \rightarrow S(y,x) \wedge S(a,b) \rightarrow R(b,a))$</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Como se ha visto anteriormente, las propiedades owl:ObjectProperty son repersentadas con una relación unidireccional en UML entre el dominio y el rango (en el caso de este proyecto, bidireccional con especificación de cardinalidad). En este caso la propiedad owl:inverseOf puede ser representada como una relación bidireccional añadiendo en cada extremo de la relación el nombre del rol de cada instancia o clase [27].</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☑ No ☐</td></tr>
</table>

Tabla 33. Transformación owl:inverseOf

<table>
<tr><th colspan="2">Transformación T17</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:subPropertyOf</td><td></td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$R \subseteq S$

$\forall x, y ((R(x,y) \rightarrow S(x,y))$

(Restricción de relación entre instancia y un dato o valor)</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad indica que la propiedad es una especialización de otra, es decir, que si existe una propiedad P1 definida como owl:subPropertyOf de otra propiedad P2, los pares de la propiedad P1 deben ser los mismos que P2 [30]. En el caso de este propyecto, no se realiará una generalización entre las relaciones ya que no es posible incluirlo en el modelo nUML. Teniendo en cuenta la casuística de esta propiedad en la mayoría de ontologías, lo que se hará es identificar la propiedad de la que depende y “heredar” los elementos necesarios para completar la relación.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☑ No ☐</td></tr>
</table>

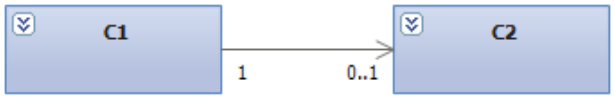
Tabla 34. Transformación owl:subPropertyOf

<table>
<tr><th colspan="2">Transformación T18</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:equivalentProperty</td><td></td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$R = S$

$\forall x,y\ (R(x, y) \rightarrow S(y,x) \wedge S(x,y) \rightarrow R(x,y))$

(Restricción de relación entre instancia y un dato o valor)</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad establece que dos propiedades tienen la misma extensión, es decir, que dos propiedades equivalentes pueden tener los mismos valores o instancias, pero hay que tener en cuenta que no significa que la propiedad sea exactamente equivalente [30] (mismo concepto). Para una equivalencia exacta es necesario expresarlo con la propiedad owl:sameAs. En cuanto a la transformación al modelo UML, no se podría expresar que la asociación de la propiedad es la misma de la de la clase de la que hereda.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML: Sí ☐ No ☑</td></tr>
</table>

Tabla 35. Transformación owl:equivalentProperty

<table>
<tr><th colspan="2">Transformación T19</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:FunctionalProperty</td><td> 
</td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$\forall x,a,b\ (P(x,a) \wedge P(x,b) \rightarrow b = a)$</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad establece que sólo puede haber un valor para las instancias incluidas en esta ella, es decir, en una propiedad P(x,y) de tipo owl:FunctionalProperty sólo puede haber un valor “y” para cada instancia “x” [30]. Es utilizada para establecer la cardinalidad global para los elementos de esta propiedad [27].

Teniendo en cuenta, que lo que hace esta propiedad es establecer una cardinalidad máxima de 1 sobre la relación, en la transformación al modelo UML será incluida esta cardinalidad máxima.</td></tr>
<tr><td colspan="2">Recuperación de información</td></tr>
<tr><td colspan="2">Transformado en UML : UML: Sí ☑ No ☐</td></tr>
</table>

Tabla 36. Transformación owl:FunctionalProperty

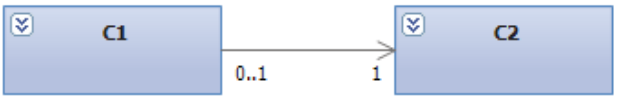
Transformación T20	
OWL	UML
owl:InverseFunctionalProperty	
	Representación lógica
	$\forall x,a,b\ (P(x, a) \wedge P(b,x) \rightarrow a = b)$
Descripción	
Es propiedad es utilizada para establecer la cardinalidad sobre el sujeto (dominio), ya que una propiedad P es de tipo owl:InverseFunctionalProperty, un valor “x” sólo puede tener un valor “y”, es decir no podría haber dos valores para y de tal manera que P(x1,y), P(x2, y) [30]. Una posible solución sería incluir la cardinalidad máxima en el extremo del dominio [27], pero a la hora de trasladar esta propiedad al modelo generado por nUML, vemos que no es posible realizarlo por las propias restricciones que tenemos en la aplicación, ya que al ser usadas todas las relaciones como asociaciones con cardinalidad siempre en el rango de la relación, si incluimos la cardinalidad en el dominio, éste sería considerado rango, y por tanto no sería consistente con nuestro modelo.	
Transformado en UML : UML: Sí ☑ No ☐	

Tabla 37. Transformación owl:inverseFunctionalProperty

<table>
<tr><th colspan="2">Transformación T21</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:TransitiveProperty</td><td></td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$\forall x,y,z\ (P(x, y) \wedge P(y,z) \rightarrow P(x,z))$</td></tr>
<tr><td colspan="2">Descripción</td></tr>
<tr><td colspan="2">Esta propiedad indica lógica necesaria sobre el elemento owl:ObjectProperty. Esplicado formalmente, Si tenemos una propiedad transitiva P(x,y), que es instancia de otra propiedad con pares P(y,z), entonces podemos inferir que existe el par P(x,z). En un diagrama UML no es posible representar estas características que implican inferencia.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Transformado en UML : UML: Sí ☐ No ☑</td></tr>
</table>

Tabla 38. Transformación owl:TransitiveProperty

<table>
<tr><th colspan="2">Transformación T21</th></tr>
<tr><th>OWL</th><th>UML</th></tr>
<tr><td rowspan="3">owl:SymmetricProperty</td><td></td></tr>
<tr><td>Representación lógica</td></tr>
<tr><td>$\forall x,y(P(x, y) \rightarrow P(y,x))$</td></tr>
<tr><th colspan="2">Descripción</th></tr>
<tr><td colspan="2">Esta propiedad indica que si el par (x,y) es instancia de P, entonces el par (y,x) también es instancia de P.</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td colspan="2">Transformado en UML: UML: Sí ☑ No ☐</td></tr>
</table>

Tabla 39. Transformaación owl:SymmetricProperty

A continuación se muestra un esquema de forma conceptual sobre la estructura de la aplicación.

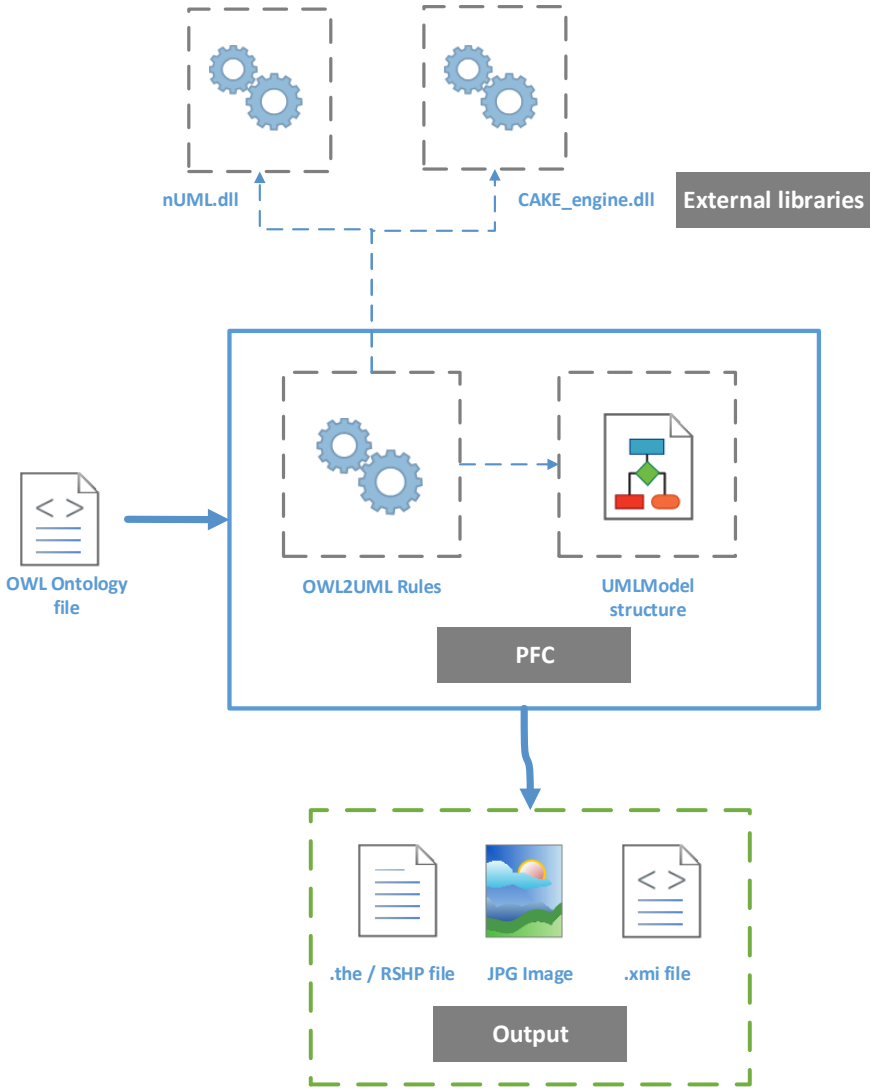


Ilustración 30. Esquema conceptual de la aplicación a diseñar

4 Análisis

En este capítulo se incluirá la especificación del sistema a desarrollar a alto nivel, recogiendo los requisitos, especificando qué debe hacer el sistema y el modelado del mismo. Se detallará toda esta información, incluyendo como debe resolverse el sistema en el apartado de Diseño.

3.1 Introducción

El objetivo del apartado de análisis del sistema es presentar una especificación sobre el sistema lo más detallada, que satisfaga las necesidades de información de los usuarios, así como servir de soporte para el desarrollo del sistema en que está motivado este proyecto.

Se dividirá este apartado dentro de un conjunto de tareas que conforman el análisis del sistema. Una de las actividades más destacadas en este apartado será la definición de una colección de requisitos que nos permitirá describir adecuadamente el sistema de información, y que nos pueda servir para comprobar que es completa la especificación de los modelos obtenidos en las actividades Identificación de Subsistemas de Análisis, Análisis de Casos de Uso, Análisis de Clases, Elaboración del Modelo de Datos, Elaboración del Modelo de Procesos y Definición de Interfaces de Usuario.

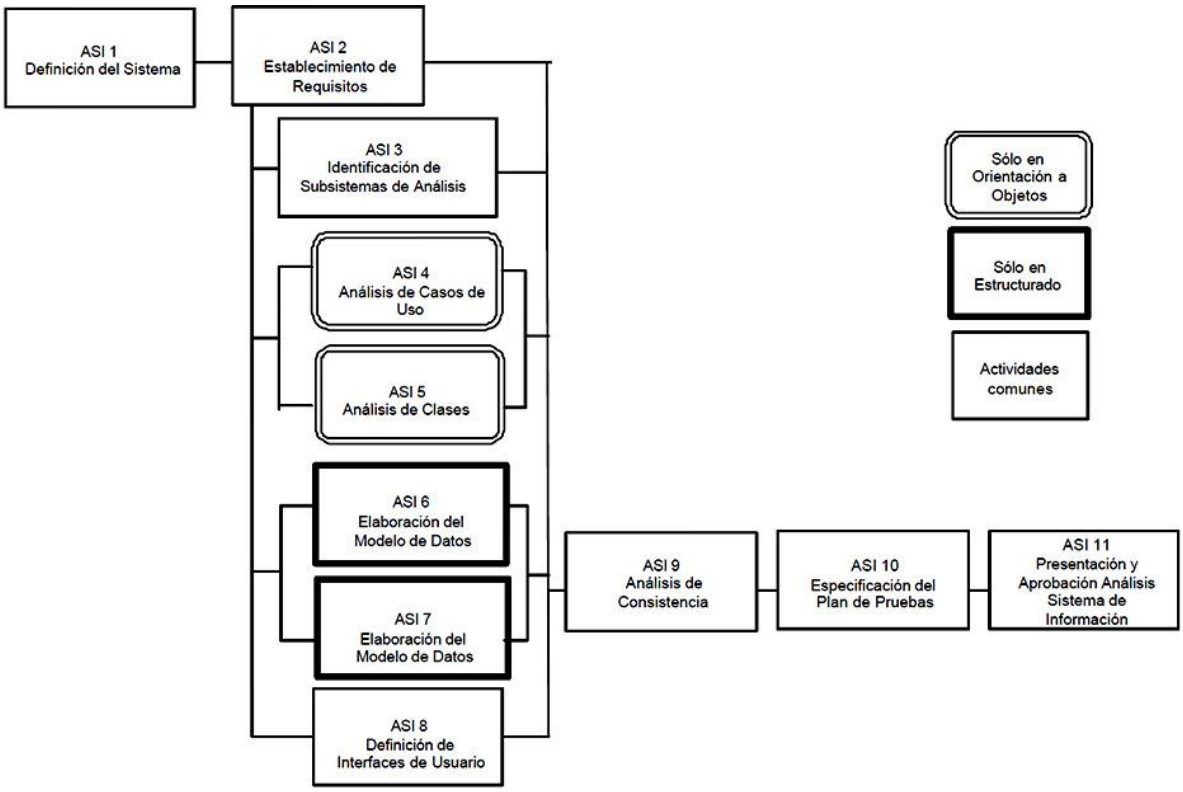


Ilustración 31. Tareas de la fase de Análisis

3.2 Definición del sistema

Esta actividad tiene como objetivo efectuar una descripción del sistema, delimitando su alcance, estableciendo las interfaces con otros sistemas e identificando a los usuarios representativos.

3.2.1 Determinación del alcance del sistema

El sistema que se desarrollará formará parte de un conjunto de aplicaciones que realizan la indexación de lenguajes al lenguaje de modelado UML. En concreto, esta aplicación estará delimitada fundamentalmente a la transformación de ontologías OWL (Web Ontology Language) , a fin de recuperar la información en modelos RSHP (Relationship Language), mediante un paso intermedio que consistirá en la transformación de dichas ontologías en diagramas de clases UML.

Hay que destacar que actualmente, la transformación de elementos UML a RSHP ya está desarrollada por lo que se realizará el uso de las librerías correspondientes para esta tarea. Teniendo esto en cuenta, el desarrollo del proyecto se centrará en primer lugar en determinar de forma teórica, basándonos en literatura científica y publicaciones disponibles, cuáles son las reglas y elementos de transformación idóneos para la transformación entre elementos OWL y UML.

A la hora de tomar decisiones sobre la transformación, se tendrá en cuenta no sólo la correspondencia entre los elementos de la ontología OWL y los diagramas de clases UML, sino también, qué es lo más favorable para poder recuperar la máxima cantidad de información en el modelo RSHP.

La aplicación dispondrá de una interfaz de usuario en la que se podrá especificar el archivo con formato OWL a recuperar, siendo este proceso transparente hasta la muestra de resultados en un modelo RSHP.

3.2.2 Identificación del entorno tecnológico

Como se ha comentado anteriormente, esta aplicación estará incluida dentro de otra aplicación que reúne varias aplicaciones de transformación a lenguaje de modelado UML. Por ello a la hora de definir el entorno tecnológico debemos tener en cuenta las necesidades sobre hardware y software donde se integrará esta aplicación.

- Procesador a 1.5GHz
- Memoria RAM 1GB
- Espacio disco duro: 25 GB
- Microsoft .NET Framework 4.0 instalado en la máquina
- Sistema operativo Windows XP o Windows 7

Especificación de estándares y normas

3.2.3.1 Restricciones generales

- Los ficheros utilizados como datos de entrada deberán tener el formato recomendado por el W3C para la definición de ontologías en OWL, es decir, se deberán cumplir las normas y recomendaciones de esta organización en cuanto a:
    - Definición de elementos de la ontología.
    - Correcta definición de espacio de nombres, formato y estructura de un fichero OWL.
- Para la transformación de ficheros con extensión .XMI se utilizarán librerías de CAKE.
- Para la transformación de estructuras de datos construidas en la aplicación a elementos UML, se necesitarán las librerías correspondientes para el acceso a los métodos de nUML.

3.2.3.2 Supuestos y dependencias

La aplicación que se va a desarrollar forma parte de un conjunto de otras aplicaciones que son utilizadas para la recuperación de información en UML y RSHP. Teniendo esto en cuenta, la aplicación se desarrollará para ser utilizada de dos formas: con una péquela interfaz para la selección del archivo por parte del usuario, y su uso integrándola en otra aplicación mediante librerías .dll correspondientes.

3.2.3.3 Entorno operacional

La aplicación será desarrollada en el siguiente entorno:

- Visual Studio 2010 Professional
- .NET Framework 4.0
- Sistema Operativo Wndows 7 - 64 bits

- Tortoise SVN 1.7.7 64-bit

3.2.3.4 Identificación de usuarios participantes y finales

Se define en este apartado los usuarios que intervendrán activamente en el proyecto, o pueden estar afectados por él, es decir los stakeholders, y además los usuarios finales del sistema. Definimos los siguientes:

- **Stakeholders:**

  - **The Reuse Company**: este sistema estará integrado en la aplicación UML Models perteneciente a The Reuse Company, por tanto, se colaborará con usuarios de esta compañía para la identificación de requisitos así como para el seguimiento del proyecto.

  - **Universidad Carlos III**: Este trabajo forma parte del Proyecto de Fin de Carrera en la Universidad Carlos III, por lo que también se trabajará con usuarios de dicha universidad para ajustar cualquier elemento del proyecto a los requisitos y necesidades académicas.

- **Usuarios finales:**

  - Los usuarios finales de esta aplicación será el personal de The Reuse Company, a fin de utilizarlo como medio para la recuperación de ontologías, utilizando otras aplicaciones en UML Models. Por tanto intervendrán:

    - Arquitectos y Analista Software.
    - Programadores.
    - Jefe de Proyecto
    - Responsable de Calidad
    - Cualquier otro miembro del equipo de The Reuse Company que requiera la recuperación de información de ontologías.

3.2.3.5 Estudio de la Seguridad Requerida en el Proceso de Análisis del Sistema de Información

Durante el proceso de análisis del sistema, se utilizarán recursos y elementos como librerías de CAKE o nUML que son necesarias para el desarrollo de este

sistema. Estos elementos así como la información disponible internamente sobre ellos es propiedad de The Reuse Company y su tratamiento será confidencial.

3.3 Establecimiento de requisitos

En este apartado se desarrolla la actividad de definición, análisis y validación de los requisitos a partir de la información facilitada por los usuarios, a fin de disponer de una colección bien definida de requisitos que permitan comprobar que los productos generados en las actividades ed modelización se ajustan a las necesidades de los usuarios.

A continuación se describen brevemente los requisitos que pueden ser definidos:

- **Requisitos Funcionales**: especifican "qué" tiene que hacer el software. Definen el propósito del software y se derivan de los casos de uso, que están derivados de los requisitos de capacidad del usuario.

- **Requisitos de Rendimiento**: especifican valores numéricos para variables de rendimiento, como por ejemplo tasas de transferencia, frecuencia, capacidad y velocidad de proceso.

- **Requisitos de Interfaz**: especifican hardware y/o software (como por ejemplo bases de datos) con el que el sistema o componentes del sistema deben interactuar o comunicarse. Los requisitos de interfaz se deben clasificar en hardware, software y de comunicaciones.

- **Requisitos de Operación**: Aquellos que van a indicar cómo va a realizar el sistema las tareas para las que ha sido construido, garantizando los niveles de servicio requeridos (tiempo de respuesta, número de usuarios, memoria utilizada, etc.). Los niveles de servicio se especifican formalmente en el proceso Implantación y Aceptación del Sistema.

- **Requisitos de Recursos**: especifican los límites superiores en recursos físicos tales como potencia de proceso, la memoria principal, espacio de disco, etc. Estos requisitos son necesarios cuando una ampliación de hardware posterior a la puesta en funcionamiento suponga un coste demasiado elevado, como puede ser el caso de muchos sistemas empotrados

- **Requisitos de Comprobación**. Especifican las limitaciones que afectan a cómo el software debe verificar los datos de entrada y salida. Incluyen requisitos para la simulación, la emulación, pruebas reales con entradas simuladas, pruebas reales con entradas verdaderas, etc.

- **Requisitos para la Aceptación de las Pruebas**: especifican las limitaciones en cómo el software debe ser validado, es decir, cómo se debe comprobar que el software cumple con los requisitos establecidos.

- **Requisitos de Documentación**: especifican los requisitos específicos del proyecto para la documentación, además de los contenidos en los estándares.

- **Requisitos de Seguridad**: especifican los requisitos para asegurar el sistema contra amenazas de confidencialidad, la integridad y la disponibilidad. Por ejemplo se especificarán los accesos que deban ser de sólo lectura, sistemas de autenticación de usuarios o sistemas protección de virus. También se puede especificar el nivel de la protección física para los equipos hardware.

- **Requisitos de Calidad**: especifican los atributos del software que aseguran que será adecuado para su propósito. Se deben utilizar métricas para medir la calidad de estos atributos. Tanto los atributos como las métricas a utilizar para medirlos se especifican en el Plan de Aseguramiento de Calidad.

- **Requisitos de mantenimiento**: especifican la facilidad que tendrá el software para reparar los defectos o adaptarlo a nuevos requisitos.

- **Requisitos de daño**: estos especifican cualquier requisito para reducir la posibilidad del daño que puede surgir del fracaso del software. Los requisitos de daño pueden identificar las funciones críticas cuyo fracaso puede ser perjudicial para personas o propiedades.

Los requisitos estarás definidos de acuerdo al siguiente formato estándar:

- **Identificación:** Código que identifica unívocamente el requisito. Debe respetarse el siguiente formato:

R-YY

En el código anterior tenemos que XXX es el identificador del tipo de requisito, e YY será el valor numérico para identificar inequívocamente el código del requisito.

XXX	TIPO DE REQUISITO	XXX	TIPO DE REQUISITO
SF	Requisito Funcional	ACE	Requisito de Aceptación
REN	Requisito de Rendimiento	DOC	Requisito de Documentación
INT	Requisito de Interfaz	SEG	Requisito de Seguridad
OPE	Requisito de Operación	CAL	Requisito de Calidad
REC	Requisito de Recurso	MAN	Requisito de Mantenimiento
COM	Requisito de Comprobación	DAÑ	Requisito de Daño
SIS	Requisito de Sistema		

- **Nombre:** Se representa de forma breve y lo más clara posible el requisito.
- **Prioridad:** Se incluye de la importancia del requisito. La prioridad puede ser *alta, media* o *baja*.
- **Fuente:** Origen del requisito de usuario, pudiendo ser The Reuse Company o Universidad Carlos III.
- **Necesidad:** Establece el nivel de importancia del requisito para el usuario. La necesidad puede ser *esencial, deseable* y *opcional*.
- **Claridad:** Evalúa el nivel en que un requisito es claro, evitando sobretodo la posibilidad de ambigüedad. La claridad puede tomar los valores *alta, media* y *baja.*
- **Verificabilidad:** Indica la medida en que se puede probar el software que da solución al requisito. Los posibles valores de la verificabilidad pueden ser *alta, media* y *baja.*
- **Estabilidad:** Establece el valor de la estabilidad del requisito. La estabilidad puede variar a lo largo de la vida del producto. Los posibles valores que puede tomar la estabilidad son *alta, media* y *baja.*

- **Descripción:** Descripción textual del requisito. Ésta debe ser breve pero intentando explicar de forma detallada y precisa el requisito descrito.

3.3.1 Obtención de requisitos

3.3.1.1 Requisitos funcionales

Identificador: RSF-000		
Nombre:	Nombre del requisito, siendo lo más preciso y breve posible.	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	Descripción textual del requisito obtenido.	
Trazabilidad:	Otros requisitos que tienen trazabilidad con el actual	

Tabla 40. RSF-000, modelo de tabla de requisitos

Identificador: RSF-001		
Nombre:	Procesamiento de la ontología	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá procesar una ontología desde el formulario seleccionando el archivo a procesar. Recibirá información sobre el éxito o en caso contrario problemas en la ejecución, así como los nuevos ficheros disponibles con la información recuperada.	
Trazabilidad:	RSIS-001, RSIS-002, RSIS-003, RSIS-004, RSIS-005, RSIS-006, RSIS-	

	007, RSIS-008, RSIS-009, RSIS-010, RSIS-011, RSIS-012, RSIS-013, RSIS-014, RSIS-015, RSIS-016, RSIS-017, RSIS-018, RSIS-019, RSIS-020, RSIS-021, RSIS-022, RSIS-023, RSIS-024, RIN-001, RIN-002, RCOM-001

Tabla 41. RSF-001 Precesamiento de la ontología

3.3.1.2 Requisitos de sistema

Identificador: RSIS-001		
Nombre:	Recuperación de clases OWL	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar clases atómicas/axiomas definidas en ontologías OWL, siendo incluidas tanto en el modelo UML como en el modelo RSHP.	
Trazabilidad:	RIN-001, RIN-002, RCOM-001	

Tabla 42. RSIS-001 Recuperación de clases OWL

Identificador: RSIS -002		
Nombre:	Recuperación de jerarquías OWL (rdfs:subClassOf)	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐

	Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar clases definidas en ontologías OWL, siendo incluidas tanto en el modelo UML como en el modelo RSHP.
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001

Tabla 43.RSIS-002 Recuperación de jerarquías en OWL

Identificador: RSIS -003	
Nombre:	Recuperación de clases equivalentes (owl:equivalentClass)
Prioridad: ☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	
Claridad: ☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que indiquen equivalencia entre clases, siendo incluidas tanto en el modelo UML como en el modelo RSHP.
Trazabilidad:	RSIS-001, RSIS-002, RIN-001, RIN-002, RCOM-001

Tabla 44. RSIS-003 Recuperación de clases equivalentes

Identificador: RSIS-004	
Nombre:	Recuperación de clases disjuntas (owl:disjointWith)
Prioridad: ☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional	
Claridad: ☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja	

Descripción:	El usuario podrá recuperar clases disjuntas de la ontología OWL siendo incluidas en el modelo UML.
Trazabilidad:	RSIS-001, RSIS -002, RIN-001, RIN-002, RCOM-001

Tabla 45. RSIS-004 Recuperación de clases disjuntas

Identificador: RSIS-005		
Nombre:	Recuperación de elementos tipo enumeración (owl:oneOf)	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que representan enumeración de instancias al definir una clase, e incluirlas en el modelo UML como relación de clase a instancias. Serán incluidas tanto en el modelo UML como en el modelo RSHP. Las instancias de este elemento podrían ser definidas de dos formas posibles, mediante el constructor “owl:Thing”, o declarando la instancia, teniendo el nombre del nodo igual que la clase instanciada.	
Trazabilidad:	RSIS-001, RSIS-019, RSIS-024, RIN-001, RIN-002, RCOM-001	

Tabla 46. RSIS-005 Recuperación de elementos de enumeración (owl:oneOf)

Identificador: RSIS-006		
Nombre:	Recuperación de elementos de tipo intersección (owl:intersectionOf)	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company

Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que representan definiciones de clases mediante intersección. Estas podrán ser mediante especialización, o relación ObjectProperty (o ambas a la vez). Serán incluidas tanto en el modelo UML como en el modelo RSHP.	
Trazabilidad:	RSIS -001, RSIS -002, RSIS-009, RSIS-013, RIN-001, RIN-002, RCOM-001, RSF-014	

Tabla 47. RSIS-006 Recuperación de elementos de tipo intersección (owl:intersectionOf)

Identificador: RSIS-007		
Nombre:	Recuperación de elementos de tipo unión (owl:unionOf)	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que son definidos como una unión de otros elementos (herencia múltiple). Serán incluidas en el modelo UML utilizando comentarios en el elemento correspondiente.	
Trazabilidad:	RSIS-001, RSIS-002, RSIS-025,RIN-001, RIN-002, RCOM-001	

Tabla 48. Recuperación de elementos tipo unión (owl:unionOf)

Identificador: RSIS-008	
Nombre:	Recuperación de elementos de tipo restricción (owl:allValuesFrom)

Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que son representan restricciones hacia una instancia o valor.	
Trazabilidad:	RSIS-001, RSIS-002, RSIS-005, RSIS-019, RSIS-024, RIN-001, RIN-002, RCOM-001, RSIS-013	

Tabla 49. RSIS-008 Recuperación de elementos de tipo restricción (owl:allValuesFrom)

Identificador: RSIS-009		
Nombre:	Recuperación de elementos de tipo restricción (owl:hasValue)	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que representan una restricción en cuanto a la relación de las clases y sus instancias.	
Trazabilidad:	RSIS-001, RSIS-019, RSIS-024, RIN-001, RIN-002, RCOM-001	

Tabla 50. RSIS-009 Recuperación de elementos de tipo restricción (owl:hasValue)

Identificador: RSIS-010		
Nombre:	Recuperación de restricciones de cardinalidad máxima (owl:maxCardinality)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que especifican la cardinalidad máxima de las relaciones tratadas.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001, RSIS-013, RSIS-019, RSIS-024	

Tabla 51. RSIS-010 Recuperación de cardinalidad máxima (owl:maxCardinality)

Identificador: RSIS-011		
Nombre:	Recuperación de restricciones de cardinalidad mínima (owl:minCardinality)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que especifican la cardinalidad mínima de las relaciones tratadas.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001, RSIS-013, RSIS-019, RSIS-024	

Tabla 52. RSIS-011 Recuperación de cardinalidad mínima (owl:minCardinality)

Identificador: RSIS-012		
Nombre:	Recuperación de restricciones de cardinalidad exacta (owl:cardinality)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar elementos de la ontología OWL que especifican la cardinalidad exacta de las relaciones tratadas.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001, RSIS-013, RSIS-019, RSIS-024	

Tabla 53. RSIS-012 Recuperación de cardinalidad exacta (owl:cardinality)

Identificador: RSIS-013		
Nombre:	Recuperación de propiedades de relación (owl:ObjectProperty)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar las propiedades de tipo “ObjectProperty” de la ontología OWL e incluirse en los modelos UML y RSHP.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001	

Tabla 54. RSIS-013 Recuperación de propiedades de relación (owl:OjectProperty)

Identificador: RSIS-014		
Nombre:	Recuperación de propiedades de relación (owl:DatatypeProperty)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar las propiedades de tipo “DataType” de la ontología OWL a fin de poder representarlas como atributos en UML.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001	

Tabla 55. RSIS-014 Recuperación de propiedades de relación (owl:DatatypeProperty)

Identificador: RSIS-015		
Nombre:	Recuperación de propiedades de especialización (rdfs:subPropertyOf)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar propiedades del tipo “rdfs:subPropertyOf” pudiendo heredar elementos de la propiedad padre.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001, RSIS-013, RSIS-014	

Tabla 56. RSIS-015 Recuperación de propiedades de especialización (rdfs:subPropertyOf)

Identificador: RSIS-016		
Nombre:	Recuperación de propiedades de relación inversa (owl:inverseOf)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar elementos de tipo “owl:inverseOf”, donde se especifica la relación inversa entre dos elementos especificada en una relación (ObjectProperty).	
Trazabilidad:	RSIS-001, RSIS-002, RIN-001, RIN-002, RCOM-001, RSIS-013	

Tabla 57. RSIS-016 Recuperación de propiedades de relación inversa (owl:inverseOf)

Identificador: RSIS-017		
Nombre:	Recuperación de propiedades funcionales (owl:FuntionalProperty)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar elementos de ontologías OWL donde la propiedad sea del tipo “owl:FunctionalProperty”, donde se establece una cardinalidad determinada en la relación.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001, RSIS-013	

Tabla 58. RSIS-017 Recuperación de propiedades funcionales (owl:FunctionalProperty)

Identificador: RSIS-018		
Nombre:	Recuperación de propiedades simétricas (owl:SymmetricProperty)	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar propiedades de ontologías en las que existan propiedades de tipo "SymmetricProperty". Esta transformación se obtendrá en un fichero con extensión .xmi (UML) y otro con extensión .the (RSHP), además de una representación gráfica o visual de los elementos recuperados.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001, RSIS-013	

Tabla 59. RSIS-018 Recuperación de propiedades simétricas (owl:symmetricProperty)

Identificador: RSIS-019		
Nombre:	Recuperación de instancias	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	El usuario podrá recuperar las instancias definidas mediante la propiedad "owl:Thing"	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001	

Tabla 60. RSIS-019 Recuperación de instancias

Identificador: RSIS-020		
Nombre:	Verificación del archivo como ontología OWL	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	Siempre que se ejecute el procesamiento de la ontología se verificará que la ontología cumple recomendaciones básicas en cuanto a formato y estructura de la misma.	
Trazabilidad:	RCOM-001, RIN-001, RIN-002	

Tabla 61. RSIS-020 Verificación del archivo OWL

Identificador: RSIS-021		
Nombre:	Inserción en modelo UML	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	Cuando el usuario ejecute la aplicación el sistema será capaz de insertar los elementos compatibles (estudio de viabilidad) en el modelo UML.	
Trazabilidad:	RSIS-001, RSIS-002, RSIS-003, RSIS-004, RSIS-005, RSIS-006, RSIS-007, RSIS-008, RSIS-009, RSIS-010, RSIS-011, RSIS-012, RSIS-013, RSIS-014, RSIS-015, RSIS-016, RSIS-017, RSIS-018, RSIS-019, RSIS-020, RSIS-021, RSIS-022, RSIS-023, RSIS-024, RIN-001, RIN-002, RCOM-001	

Tabla 62. RSIS-021 Inserción en modelo UML

Identificador: RSIS-022		
Nombre:	Inserción en modelo RSHP	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	Cuando el usuario ejecute la aplicación el sistema será capaz de insertar los elementos compatibles (estudio de viabilidad) en el modelo RSHP.	
Trazabilidad:	RSIS-001, RSIS-002, RSIS-005, RSIS-006, RSIS-008, RSIS-009, RSIS-010, RSIS-011, RSIS-012, RSIS-013, RSIS-014, RSIS-015, RSIS-016, RSIS-017, RSIS-018, RSIS-019, RSIS-020, RSIS-021, RSIS-022, RSIS-023, RSIS-024, RIN-001, RIN-002, RCOM-001	

Tabla 63. RSIS-022 Inserción en modelo RSHP

Identificador: RSIS-023		
Nombre:	Recuperación de atributos (sin propiedad owl)	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar los atributos definidos en las clases, cuando estén incluidos con una etiqueta que identifica su nombre, recogiendo también su valor.	
Trazabilidad:	RSIS -001, RIN-001, RIN-002, RCOM-001	

Tabla 64. RSIS-023 Recuperación de atributos

Identificador: RSIS -024		
Nombre:	Recuperación de instancias mediante declaración de la clase	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El usuario podrá recuperar las instancias cuando estas estén definidas en que la etiqueta de definición sea el propio nombre de la clase a la que pertenece.	
Trazabilidad:	RSIS-001, RIN-001, RIN-002, RCOM-001	

Tabla 65. RSIS-024 Recuperación de instancias mediante declaración de clase

Identificador: RSIS -025		
Nombre:	Inserción de comentarios en elementos UML	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	Teniendo en cuenta que algunos elementos OWL son recuperables en UML pero no en RSHP debido a la utilización de comentarios, se incluirán comentarios en los elementos UML que son especificados en la investigación de este documento.	
Trazabilidad:	RSIS -001, RSIS -007, RSIS -003, RSIS -004, RIN-001, RIN-002, RCOM-001	

Tabla 66. RSIS-025 Inserción de comentarios en elementos UML

Identificador: RSIS -026		
Nombre:	Recuperación de elementos de tipo “owl:sameAs”	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El sistema podrá recuperar elementos de tipo “owl:sameAs” con el que se establece la equivalencia exacta entre dos instancias.	
Trazabilidad:	RSIS-001, RSIS-019, RSIS -024, RIN-001, RIN-002, RCOM-001	

Tabla 67. RSIS-026 Recuperación de elementos tipo owl:sameAs

Identificador: RSIS -027		
Nombre:	Recuperación de elementos de tipo “owl:differentFrom”	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	El sistema podrá recuperar elementos de tipo “owl:differentFrom” con el que se establece la que dos instancias son distintas entre sí.	
Trazabilidad:	RSIS-001, RSIS-019, RSIS-024, RIN-001, RIN-002, RCOM-001	

Tabla 68. RSIS-027 Recuperación de elementos tipo owl:differentFrom

3.3 3.2.1.2 Requisitos de interfaz

Identificador: RIN-001		
Nombre:	Utilización de librarías CAKE para transformación a RSHP	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	La aplicación desarrollada en este proyecto podrá utilizar las librerías (dll) existentes para la utilización de CAKE en la transformación de los archivos con extensión (“.xmi”) a extensión “.the” (RSHP)	
Trazabilidad:		

Tabla 69. RIN-001 Utilización de librerías CAKE para transformación

Identificador: RIN-002		
Nombre:	Utilización de librerías nUML para la inserción de elementos en el modelo UML	
Prioridad: ☑ Alta ☐ Media ☐ Baja		**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja		**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad: ☐ Alta ☐ Media ☐ Baja		
Descripción:	La aplicación desarrollada en este proyecto podrá utilizar las librerías (dll) existentes para la utilización de nUML para la inserción de elementos en el modelo UML así como para la	

	generación de los archivos con extensión ".xmi".
Trazabilidad:	

Tabla 70. RIN-002 Utilización de librerías nUML para inserción en modelo

3.4 3.2.1.3 Requisitos de comprobación

Identificador: RCOM-001		
Nombre:	Verificación de ontologías de acuerdo a W3C	
Prioridad:	☑ Alta ☐ Media ☐ Baja	**Fuente:** ☑ UC3M ☑ The Reuse Company
Necesidad:	☑ Esencial ☐ Deseable ☐ Opcional	
Claridad:	☑ Alta ☐ Media ☐ Baja	**Verificabilidad:** ☑ Alta ☐ Media ☐ Baja
Estabilidad:	☐ Alta ☐ Media ☐ Baja	
Descripción:	En cualquier ejecución de la aplicación se podrá verificar que el archivo a procesar cumple con las recomendaciones de W3C para ontologías OWL, a fin de ser considerada una ontología válida. Si el archivo procesado no es válido se cancelará el proceso y se avisará al usuario.	
Trazabilidad:		

Tabla 71. RCOM-001 Verificación de ontologías

3.4.1 Especificación de casos de uso

Se especifican en este apartado los casos de uso, teniendo en cuenta que se capturan los requisitos funcionales del sistema y se recoge vocabulario del dominio. En los casos de uso uno o más actores interaccionan con el sistema que realiza algunas acciones.

A fin de poder realizarlo de la forma más clara y precisa posible, se dividirán las distintas partes del sistema en subsistemas, realizando los casos de uso sobre cada una de ellas.

En los casos que se van a presentar a continuación sólo existirá el rol de usuario.

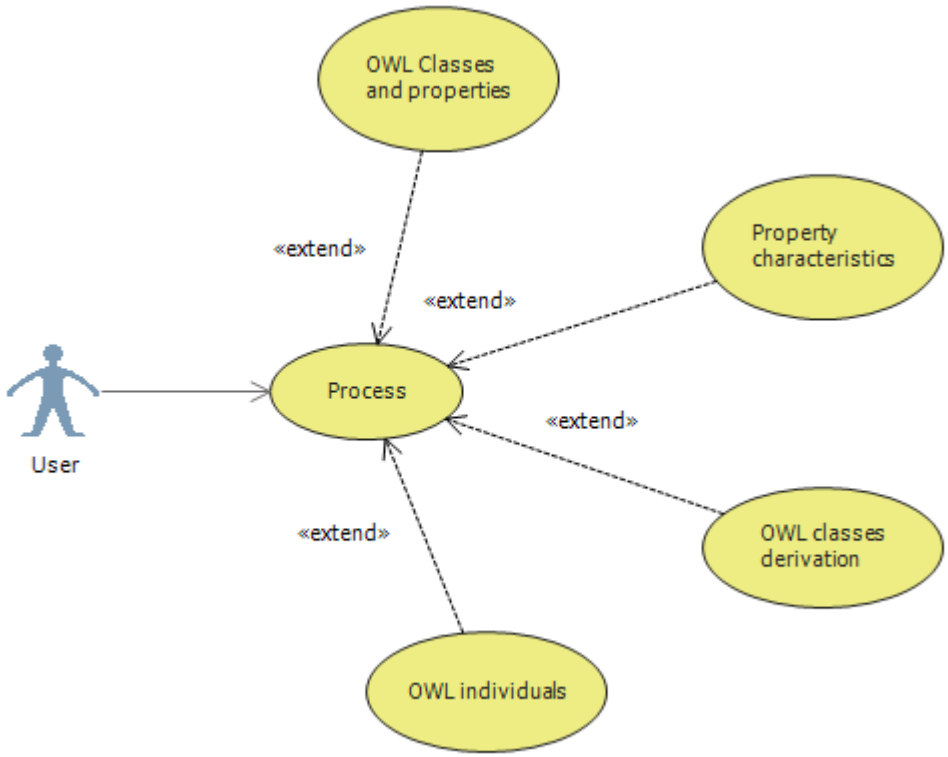


Ilustración 32. Diagrama general de casos de uso

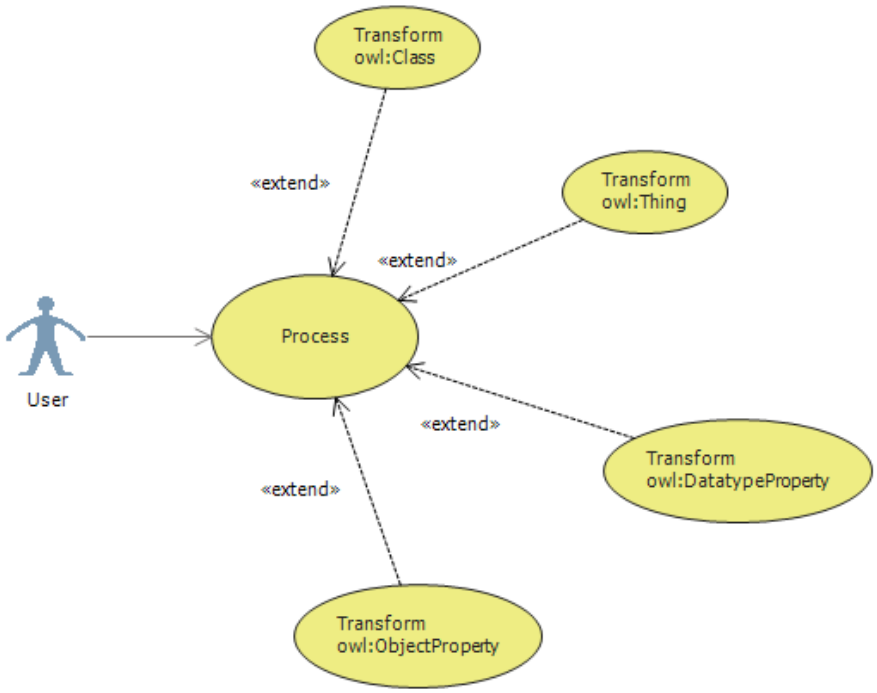


Ilustración 33. Diagrama de casos de uso: OWL classes and properties transformation

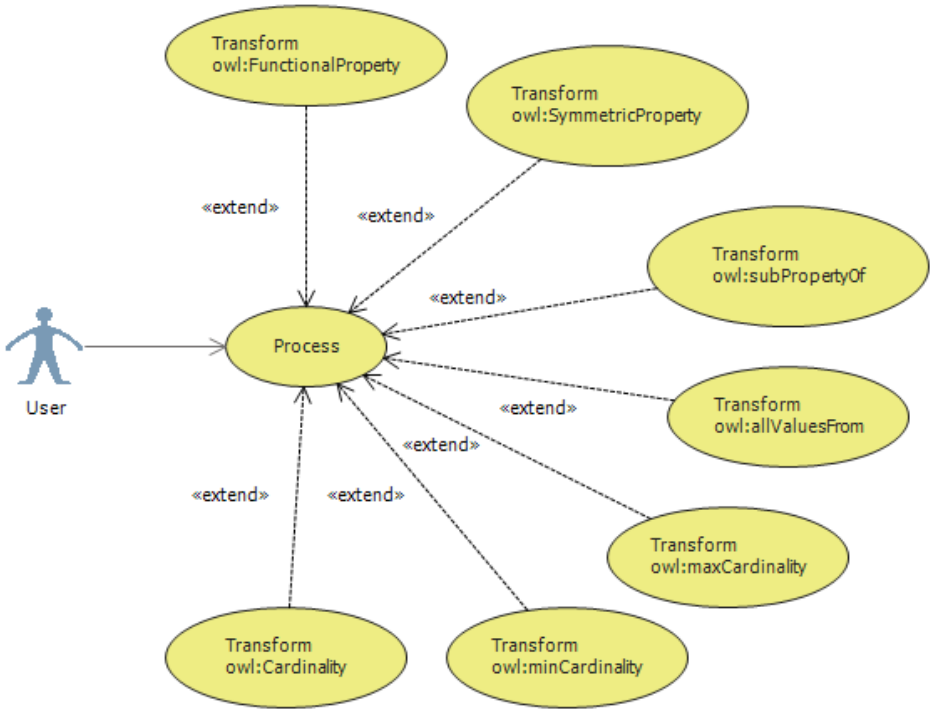


Ilustración 34. Diagrama de casos de uso: OWL Property characteristics

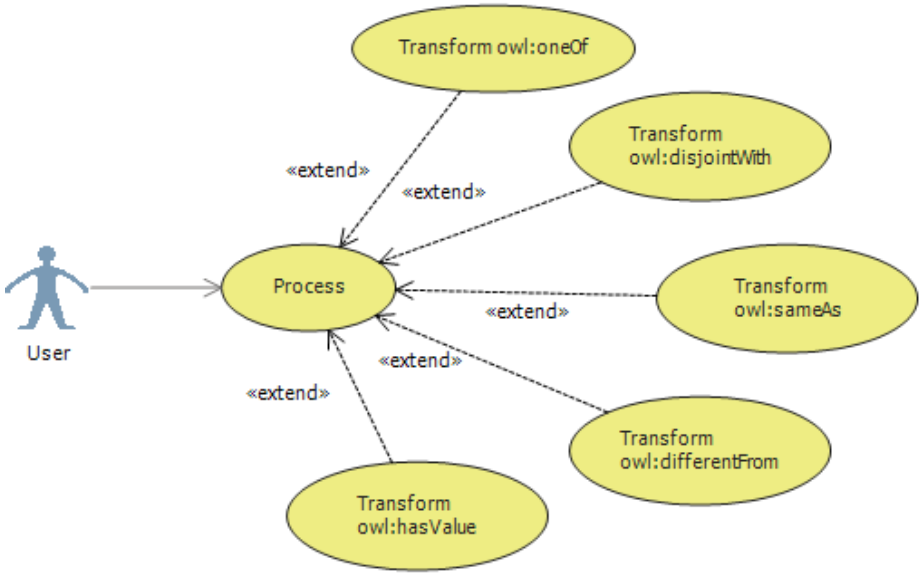


Ilustración 35.Diagrama de casos de uso: Individuals

Caso de uso: Recuperar clase (owl:Class)	
Identificador:	CU-001
Descripción:	El usuario puede recuperar clases desde una ontología OWL.
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir al menos una clase en la ontología.
Postcondiciones:	Se añade un elemento tipo clase al modelo UML. Se añade una clase al modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión ".owl" a recuperar 3- El usuario hace clic en "Procesar" 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta La clase ya existe en el modelo (no se inserta)

Tabla 72.CU-001 Recuperar clase

Caso de uso: Obtener jerarquía (rdfs:subClassOf)	
Identificador:	CU-002
Descripción:	El usuario puede recuperar las jerarquías de clases de una ontología OWL.
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir al menos una clase y una subclase de (entre sí) en la ontología.
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añade un elemento tipo clase al modelo UML (subclase) Se añade al modelo la jerarquía (generalización) entre ambos elementos Se añaden las clases al modelo RSHP. Se incluye la jerarquía en el modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión ".owl" a recuperar 3- El usuario hace clic en "Procesar" 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo (repetidos en la ontología)

Tabla 73. CU-002 Obtener jerarquía

Caso de uso: Obtener clases equivalentes (owl:equivalentClass)	
Identificador:	CU-003
Descripción:	El usuario puede recuperar clases equivalentes de ontologías OWL (owl:equivalentClass)
Actores:	Usuario

Precondiciones:	La ontología debe tener una estructura correcta. Debe existir en la ontología al menos dos clases Debe existir en la ontología una propiedad del tipo "owl:equivalentClass" definida correctamente
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añade un elemento tipo clase al modelo UML (clase) Se añade al modelo UML dos generalizaciones entre ambos elementos Se añaden los elementos UML al modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión ".owl" a recuperar 3- El usuario hace clic en "Procesar" 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología)

Tabla 74. CU-003 Obtener clases equivalentes

Caso de uso: Obtener clases disjuntas (owl:DisjointWith)	
Identificador:	CU-004
Descripción:	El usuario puede recuperar las clases disjuntas de ontologías OWL
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Deben existir al menos dos clases, para especificar la propiedad "disjointWith" entre ellas
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añade un elemento tipo clase al modelo UML (subclase) Se añade al modelo la jerarquía (generalización) entre ambos elementos

	Se añaden los elementos UML al modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión ".owl" a recuperar 3- El usuario hace clic en "Procesar" 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología)

Tabla 75. CU-004 Obtener clases disjuntas

Caso de uso: Recuperar enumeración (owl:oneOf)	
Identificador:	CU-005
Descripción:	El usuario puede recuperar enumeraciones de una ontología OWL
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir al menos la definición de una clase para la que existe la enumeración
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añaden las instancias correspondientes a la lista (enumeración) Se añaden las relaciones de instancia entre la clase y las instancias Se añaden los elementos al modelo UML Se añaden los elementos UML al modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión ".owl" a recuperar 3- El usuario hace clic en "Procesar" 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)

Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 76. CU-005 Recuperar enumeración

Caso de uso: Recuperar propiedad intersección (owl:intersectionOf)	
Identificador:	CU-006
Descripción:	El usuario puede recuperar definiciones de clases basadas en intersecciones
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir al menos la clase definida, y el resto de elementos utilizados en la propiedad de intersección
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) En función del tipo de intersección se añadirán distintos elementos (asociación, generalización) Se añaden los elementos al modelo UML Se añaden los elementos UML al modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión ".owl" a recuperar 3- El usuario hace clic en "Procesar" 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 77.CU-006 Recuperar propiedad intersección

Caso de uso: Recuperar restricción owl:allValuesFrom	
Identificador:	CU-007
Descripción:	El usuario puede recuperar propiedades del tipo “allValuesFrom”, obteniendo esta propiedad en UML y RSHP
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir al menos la definición de una clase para la que existe la propiedad allValuesFrom Debe existir el valor o instancia para el que se indica esta propiedad
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añaden el elemento de tipo “rango” para el que aplica la relación (clase) Se añaden las relaciones de instancia entre elementos Se añade la cardinalidad correspondiente a la propiedad sobre los elementos anteriores Se insertan los elementos al modelo UML Se añaden los elementos UML al modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 78. CU-007 Recuperar restricción allValuesFrom

Caso de uso: Recuperar cardinalidad mínima (owl:minCardinality)	
Identificador:	CU-008
Descripción:	El usuario puede recuperar restricciones sobre cardinalidad mínima
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir al menos la definición de una clase para la que existe la restricción de cardinalidad Debe existir la relación sobre la que se incluirá la restricción de cardinalidad Debe existir otra clase/instancia como rango de la relación
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añaden el elemento de tipo “rango” para el que aplica la relación (clase) Se añaden las relaciones de instancia entre elementos Se añade la cardinalidad mínima correspondiente a la restricción sobre los elementos anteriores Se insertan los elementos al modelo UML Se añaden los elementos UML al modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 79. CU-008 Recuperar cardinalidad mínima

Caso de uso: Recuperar cardinalidad exacta (owl:cardinality)	
Identificador:	CU-009
Descripción:	El usuario puede recuperar restricciones sobre cardinalidad exacta
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir al menos la definición de una clase para la que existe la restricción de cardinalidad Debe existir la relación sobre la que se incluirá la restricción de cardinalidad Debe existir otra clase/instancia como rango de la relación
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añaden el elemento de tipo “rango” para el que aplica la relación (clase) Se añaden las relaciones de instancia entre elementos Se añade la cardinalidad mínima correspondiente a la restricción sobre los elementos anteriores Se insertan los elementos al modelo UML Se añaden los elementos UML al modelo RSHP.
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 80. CU-009 recuperar cardinalidad exacta

Caso de uso: Recuperar propiedad unión (owl:unionOf)	
Identificador:	CU-010
Descripción:	El usuario puede recuperar propiedades de tipo unión de ontologías OWL en modelos UML
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir al menos la definición de una clase para la que se aplica la propiedad “owl:unionOf” Debe existir más de una clase, para poder aplicar el concepto de unión (instancia en alguna de las clases de especialización)
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añaden el resto de clases que forman parte del conjunto de generalización (clases) Se añade comentario {complete} al elemento de generalización Se insertan los elementos al modelo UML
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 81. CU-010 recuperar propiedad unión

Caso de uso: Recuperar instancia o valor (owl:hasValue)	
Identificador:	CU-011
Descripción:	El usuario puede recuperar propiedades de tipo “owl:hasValue” a fin de relacionar una clase con sus instancias o valores. Las

	instancias pueden ser definidas de varias formas.
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Puede existir al menos la definición de una clase para la que se aplica la propiedad “owl:hasValue” Puede existir la definición de una instancia mediante la definición de un nodo con el nombre de la clase que se instancia Debe existir la instancia o valor al que hace referencia la propiedad
Postcondiciones:	Se añade un elemento tipo clase al modelo UML (clase) Se añaden el resto de elementos (instancia o valor) al modelo UML Se añade la etiqueta “has_instance” (si existe) para identificarlo en el modelo UML Se añade la relación de tipo “has_instance” Se insertan los elementos al modelo UML Se insertan los elementos en el modelo RSHP
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 82. CU-011 Recuperar instancia o valor

Caso de uso: Recuperar owl:ObjectProperty	
Identificador:	CU-012
Descripción:	El usuario puede recuperar propiedades de tipo “owl:ObjectProperty” a fin de relacionar una clase con otras clases (relación dominio y rango)
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta Deben existir las clases sobre las que se aplica la propiedad “owl:ObjectProperty” Debe existir una propiedad “owl:ObjectProperty” definida
Postcondiciones:	Se añade la relación entre el dominio y el rango especificado mediante la propiedad. Se inserta la relación entre clases en el modelo UML Se insertan los elementos en el modelo RSHP
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 83. CU-012 Recuperar owl:ObjectProperty

Caso de uso: Recuperar owl:DatatypeProperty	
Identificador:	CU-013
Descripción:	El usuario puede recuperar propiedades de tipo “owl:DatatypeProperty” a fin de obtener los atributos o valores especificados para una clase.
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta Deben existir las clases sobre las que se aplica la propiedad “owl:DatatypeProperty” Debe existir una propiedad “owl:DatatypeProperty” definida
Postcondiciones:	Se añade la relación entre el dominio y el dato o valor especificado (owl:DtatypeProperty) Se inserta la relación entre clases en el modelo UML Se insertan los elementos en el modelo RSHP
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 84. CU-013 Recuperar owl:DatatypeProperty

Caso de uso: Recuperar owl:subPropertyOf	
Identificador:	CU-014
Descripción:	El usuario puede recuperar propiedades de tipo “rdfs:subPropertyOf” a fin de obtener información de otras propiedades (owl:ObjectProperty)
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta. Debe existir la propiedad “owl:ObjectProperty” sobre la que se hereda la información en la propiedad “owl:subPropertyOf”. Debe existir una propiedad “owl:subPropertyOf” definida.
Postcondiciones:	Se añade la información necesaria sobre la propiedad que hereda. Se inserta la información en el modelo UML Se insertan los elementos en el modelo RSHP
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 85. CU-014 Recuperar owl:subPropertyOf

Caso de uso: Recuperar owl:inverseOf	
Identificador:	CU-015
Descripción:	El usuario puede recuperar propiedades de tipo “owl:inverseOf” a fin de obtener las relaciones inversas entre objetos
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta Debe existir el par de elementos sobre el que se especifica la propiedad “owl:inverseOf” (clases) Deben existir las clases sobre las que se aplica la propiedad “owl:ObjectPropery” Debe existir una propiedad “owl:ObjectProperty” definida con la propiedad “owl:inverseOf”
Postcondiciones:	Se añade la relación con la navegación bidireccional (asociación) Se insertan los nombres de las relaciones en los elementos correspondientes Se insertan los elementos en el modelo UML Se insertan los elementos en el modelo RSHP
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 86. CU-015 Recuperar owl:inverseOf

Caso de uso: Recuperar owl:FunctionalProperty	
Identificador:	CU-016
Descripción:	El usuario puede recuperar propiedades de tipo “owl:FunctionalProperty” a establecer la máxima cardinalidad de una relación.
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta Deben existir los elementos (dominio y rango) que se especifican en la propiedad (owl:ObjectProperty) Debe existir la propiedad owl:ObjectProperty y los extremos especificados en la misma. Debe existir una propiedad “owl:FunctionalProperty” definida
Postcondiciones:	Se añade la cardinalidad máxima en la relación indicada por la propiedad, (owl:ObjectProperty) Se inserta la relación y cardinalidad entre clases en el modelo UML Se insertan los elementos en el modelo RSHP
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 87. CU-016 Recuperar owl:FunctionalProperty

Caso de uso: Recuperar owl:sameAs	
Identificador:	CU-017
Descripción:	El usuario puede recuperar propiedades de tipo “owl:sameAs”
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta Deben existir los elementos (instancias) necesarios para incluir esta relación Debe existir la propiedad “owl:sameAs” indicando las instancias a las que aplica dicha propiedad
Postcondiciones:	Se añade la relación entre instancias Se inserta la relación entre instancias en el modelo UML Se insertan los elementos en el modelo RSHP
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión “.owl” a recuperar 3- El usuario hace clic en “Procesar” 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 88. CU-017 Recuperar owl:sameAs

Caso de uso: Recuperar owl:differentFrom	
Identificador:	CU-018
Descripción:	El usuario puede recuperar propiedades de tipo

	"owl:differentFrom"
Actores:	Usuario
Precondiciones:	La ontología debe tener una estructura correcta Deben existir los elementos (instancias) necesarios para incluir esta relación Debe existir la propiedad "owl:differentFrom" indicando las instancias a las que aplica dicha propiedad
Postcondiciones:	Se añade la relación entre instancias Se inserta la relación entre instancias en el modelo UML Se insertan los elementos en el modelo RSHP
Escenario básico:	1- Accede a la aplicación 2- Selecciona el fichero con extensión ".owl" a recuperar 3- El usuario hace clic en "Procesar" 4- Se muestran los documentos completados por la aplicación (UML, RSHP y gráfico)
Excepciones:	La ontología no tiene la estructura correcta Los elementos ya existen en el modelo UML (repetidos en la ontología) La sintaxis OWL no es correcta

Tabla 89. CU-018 Recuperar owl:differentFrom

3.4.2 Descripción de la interacción de objetos

Pasamos a describir en esta apartado la interacción entre los objetos en el tiempo utilizando diagramas de secuencia para los correspondientes casos de uso. No se mostrarán todos los diagramas de secuencia teniendo en cuenta que son prácticamente iguales para el resto de casos de uso

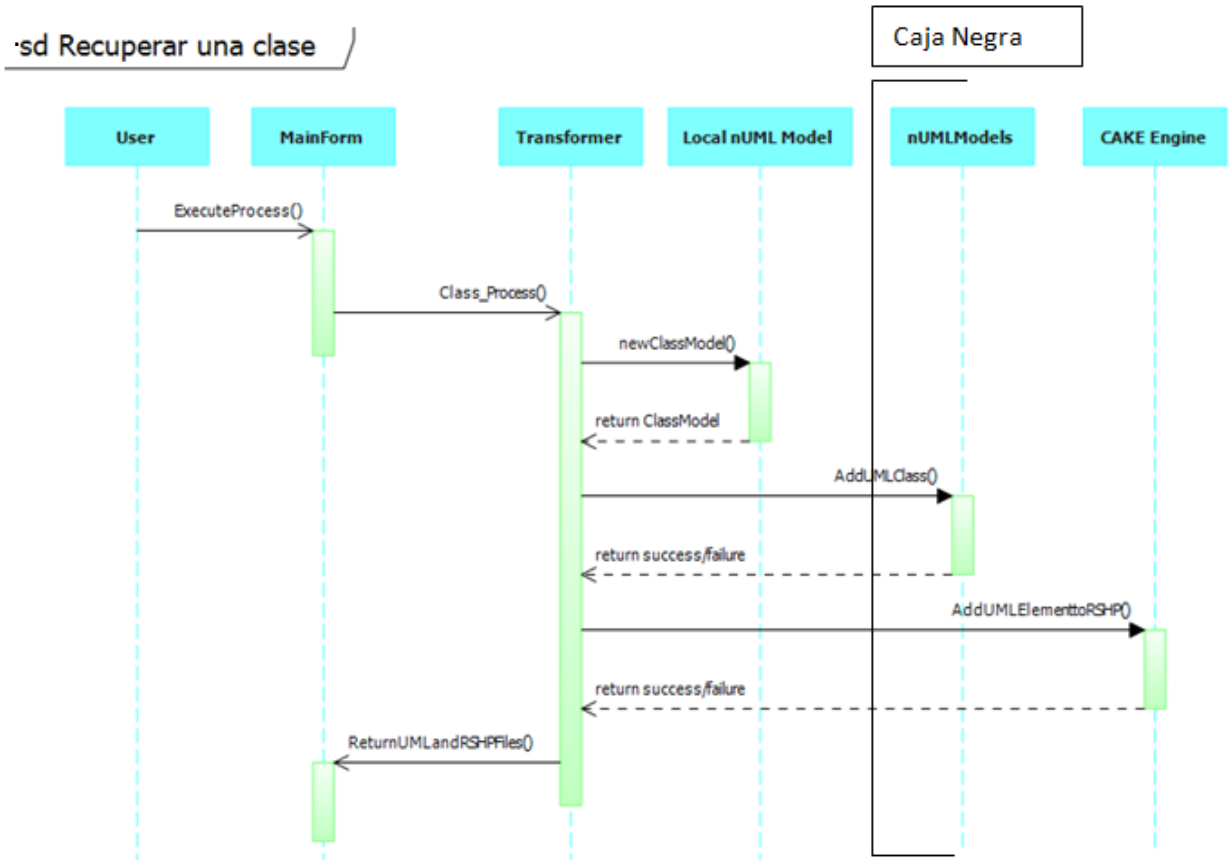


Ilustración 36. Diagrama de secuencia, Recuperar una clase

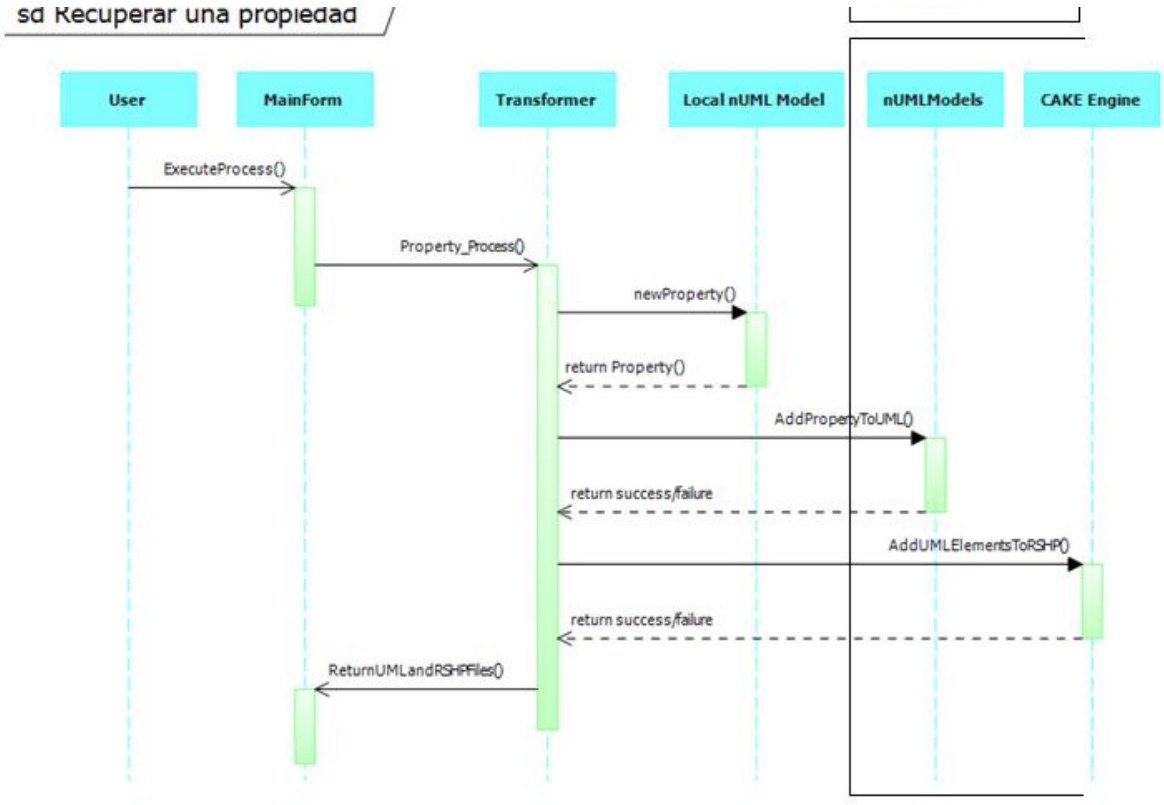


Ilustración 37. Diagrama de actividad, Recuperar una actividad

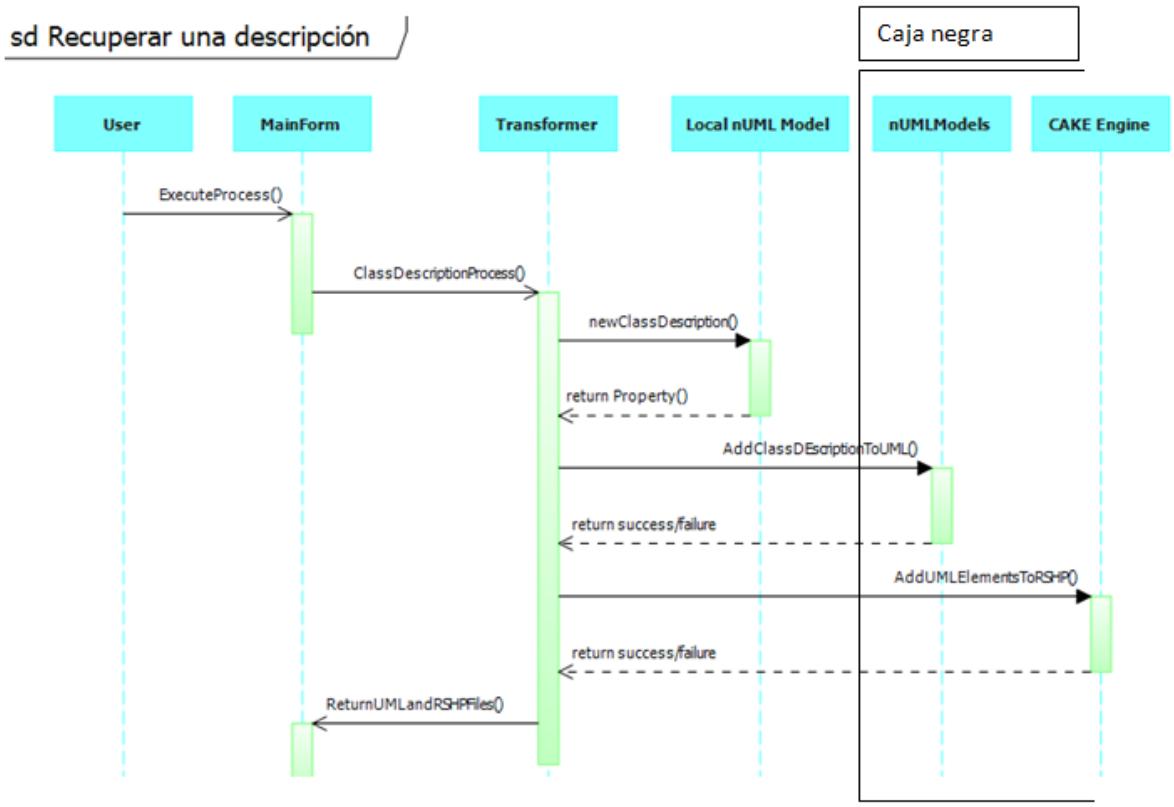


Ilustración 38. Diagrama de actividad recuperar una descripción

3.4.3 Análisis de clases

3.4.4 Identificación y especificación de clases

Se van a explicar en este apartado las clases que intervienen en la realización de los casos de uso descritos anteriormente. En la fase de diseño se realizará una especificación mucho más detallada. Este análisis constituye un nivel intermedio entre la captura de requerimientos y la de diseño.

Teniendo esto en cuenta podemos definir tres clases:

- **Clases de interfaz:** Modelan la interacción entre el sistema y los actores
- **Clases de entidad:** Modelan la información y el comportamiento asociado de conceptos (individuos, objetos, eventos) del mundo real
- **Clases de control:** Modelan la coordinación, secuencia, transacciones y control del flujo de la información.

3.4.5 Identificación de las clases asociadas a un caso de uso

Clases de entidad:

- **ClassType**: Clase que contiene los atributos de los elementos tipo clase.
- **Restriction**: Clase que contiene todos los elementos que identifican las restricciones sobre una clase.
- **RelationshipType**: Clase que contiene los atributos correspondientes a las relaciones de la ontología.
- **Generalization**: Clase que contiene los atributos correspondientes a los elementos necesarios para definir una generalización.

ClassType	
Responsabilidades:	Esta clase recoge los elementos necesarios para identificar cualquier tipo de clase en OWL. Tiene que tener también correspondencia con la estructura (atributos) que se esperan en el modelo nUML.
Atributos:	➢ Contiene los identificadores únicos de las clases ➢ Contiene información sobre el origen de la clase (tipo) ➢ Contiene información sobre las relaciones a las que pertenece simultáneamente ➢ Contiene información sobre su superclase
Operaciones:	-

Tabla 9: Definición de clase

Clases de interfaz:

- **MainForm**: Formulario que utiliza el usuario para interactuar con la aplicación.
- **User_Info**: Clase que incluye los atributos necesarios para mostrar al usuario la información de resultados de la ejecución.

User_Info	
Responsabilidades:	Esta clase recoge información sobre las estadísticas básicas de elementos recuperados desde la ontología, así como información sobre la verificación de la ontología OWL.

Atributos:	➢ Información sobre las cabeceras y namespaces del documento OWL. ➢ Información numérica de los elementos transformados
Operaciones:	➢ Función para la verificación del archivo OWL ➢ Función para contabilización de elementos transformados.

Tabla 91. Definición de información de usuario

Clases de control:

- **OWL2nUMLTransformer**: contiene la lógica del negocio del sistem

OWL2nUMLTransformer	
Responsabilidades:	Esta clase contiene la lógica del negocio para la recuperación de elementos OWL.
Atributos:	-
Operaciones:	➢ Funciones para la recuperación de clases ➢ Funciones para la recuperación de relaciones ➢ Funciones para la recuperación de restricciones ➢ Funciones para la recuperación de instancias ➢ Funciones para la inserción de elementos en el modelo nUML ➢ Funciones para el tratamiento de descripciones ➢ Funciones para el tratamiento de propiedades

Tabla 92. Definición de clase de transformación

3.5 Definición de Interfaces de Usuario

3.5.1 Definición de Interfaces de Usuario

3.5.2 Se describe a continuación la interfaz de usuario utilizada. Siendo coherentes con el resto de aplicaciones utilizadas dentro de UML Models.

3.5.3 Especificación de Principios Generales de la Interfaz

Windows Forms es la aplicación gráfica que se incluye como parte de Microsoft .NET Framework. Todos los elementos utilizados derivan de la clase Control, reutilizándose gran parte de los elementos en los distintos controles de la interfaz.

3.5.4 Identificación de Perfiles y Diálogos

Pasamos a analizar los cuadros de diálogo con los que el usuario podría interactuar con la aplicación. A través de estos se podrá obtener información sobre el éxito del procesamiento de las ontologías, así como la comunicación de posibles errores.

En primer lugar podemos observar el cuadro de diálogo que es utilizado en caso de éxito del proceso. El usuario es informado sobre la cantidad de elementos recuperados en el modelo UML, y se le da también la opción de abrir los documentos generados en este proceso:

Modelo RSHP(.the), modeleo UML (.xmi), imagen UML del modelo (.jpg).

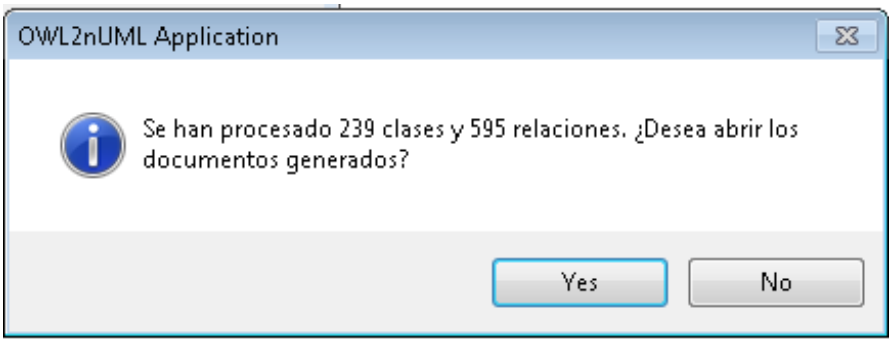


Ilustración 39. Mensaje de resumen

Por otra parte, si la ontología no puede ser procesada debido a que el documento no guarda las recomendaciones verificadas, así como por cualquier error de sintaxis, el usuario recibirá un mensaje informando de ello.

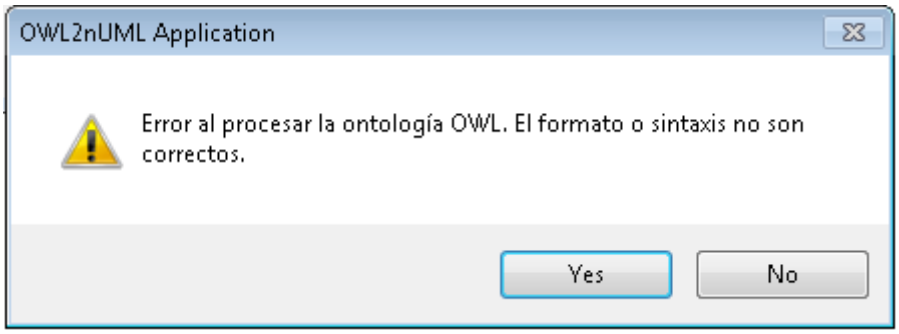


Ilustración 40. Mensaje de error

3.6 Especificación de Formatos Individuales de la Interfaz de Pantalla

No aplica

3.7 Especificación del Comportamiento Dinámico de la Interfaz

En el formulario principal de la aplicación el usuario puede seleccionar el documento con extensión ".owl" a fin de procesarlo. Se presenta un botón para ejecutar el procesamiento de la ontología y su transformación a UML y RSHP.

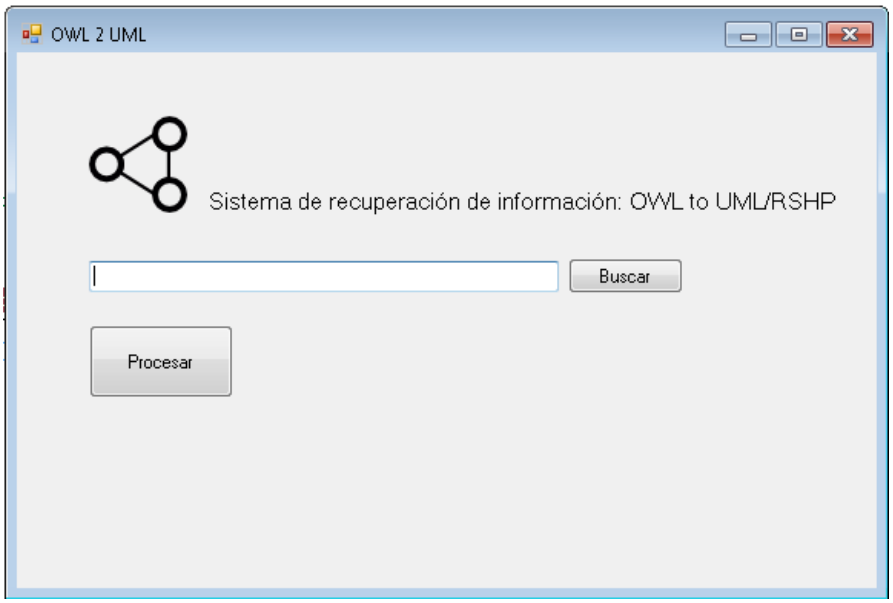


Ilustración 41. Formulario principal

3.8 Especificación de Formatos de Impresión

No aplica

Análisis de Consistencia y Especificación de Requisitos

Una de las actividades necesarias para comprobar la consistencia y coherencia de los requisitos recogidos durante esta fase del proyecto, es analizar que estos requisitos guardan trazabilidad entre ellos, es decir que se pueden identificar todos los elementos del producto a partir de un requisito.

Una de las herramientas más utilizadas en estos casos, es la matriz de trazabilidad, una tabla donde se va incluyendo la trazabilidad entre requisitos. Teniendo en cuenta que en estos casos las tablas pueden ser demasiado grandes, se incluyen en las tablas de requisitos un campo más donde se indica la trazabilidad de cada requisito.

Especificación del plan de pruebas

3.9 Definición del Alcance de las Pruebas

A fin de garantizar el correcto funcionamiento del sistema, así como para garantizar la calidad del mismo, se incluye una batería de pruebas de aceptación para verificar que el sistema cumple con los requisitos aportados por los usuarios.

Hay que tener en cuenta que por encontrarnos en la fase análisis, de momento al realizar las pruebas de aceptación no disponemos de los elementos necesarios para verificar el correcto funcionamiento con un alto grado de fiabilidad. Una vez realizadas las pruebas necesarias en la parte de diseño se podrá verificar más adecuadamente el buen funcionamiento del sistema desarrollado.

3.10 Definición de Requisitos del entorno de Pruebas

Se utilizará para la realización de las pruebas el mismo entorno que el utilizado durante la fase de desarrollo. Se incluye a continuación la información relativa a los requisitos software y hardware:

- Requisitos software:
  - Microsoft Framework 4.0
  - Visual Studio 2010 Professional
- Requisitos hardware:
  - Procesador Intel Pentium Dual Core 2.0GHz
  - 2048 MB de memoria RAM
  - 60 GB totales de disco duro

3.11 Definición de pruebas de aceptación del sistema

PA-001: Verificar el documento ontología	
Descripción	Se necesita comprobar que la aplicación verifica la estructura y formato de los elementos básicos de la ontología (namespaces, y formato recomendado). Este proceso se verifica justo antes de continuar con la recuperación de elementos de la ontología, por lo que podemos describir los siguientes pasos: 1. Se accede al formulario principal de la aplicación. 2. Se selecciona el archivo .owl a procesar mediante la utilización del botón “Buscar” 3. Se pulsa el botón “Procesar”
Resultado	El usuario recibe un cuadro de diálogo informando sobre el éxito del procesamiento de la ontología. En caso de error devuelve un mensaje informando sobre esta situación.
Requisitos relacionados	RCOM-001, RSF-001

Tabla 93. PA-001 Verificar el documento ontología

PA-002: Recuperar una jerarquía	
Descripción	Se comprueba que el sistema recupera correctamente las clases y las jerarquías especificadas en la jerarquía. A fin de seguir como es la secuencia de acciones se puede ver lo siguiente: 1. Se accede al formulario principal de la aplicación. 2. Se selecciona el archivo .owl a procesar mediante la utilización del botón “Buscar”

	3. Se pulsa el botón “Procesar”
Resultado	El proceso termina con un mensaje de éxito incluyendo información sobre la cantidad de elementos recuperados. Además, podemos ver la clase incluida en el modelo UML inspeccionando el archivo generado con extensión “.xmi”, y también el nuevo elemento en el fichero “.the”.
Requisitos relacionados	RSIS-001, RCOM-001

Tabla 94. PA-002 Recuperar una jerarquía

PA-003: Recuperar una relación objectProperty	
Descripción	Se comprueba que el sistema recupera correctamente las relaciones de tipo “owl:ObjectProperty” A fin de seguir como es la secuencia de acciones se puede ver lo siguiente: 1. Se accede al formulario principal de la aplicación. 2. Se selecciona el archivo .owl a procesar mediante la utilización del botón “Buscar” 3. Se pulsa el botón “Procesar”
Resultado	El proceso termina con un mensaje de éxito incluyendo información sobre la cantidad de elementos recuperados. Además, podemos ver las clases incluidas, y la asociación recuperadas en el modelo UML inspeccionando el archivo generado con extensión “.xmi” y también el nuevo elemento en el fichero “.the”
Requisitos relacionados	RSIS-001, RSIS-013

Tabla 95. PA-003 Recuperar una relación ObjectProperty

PA-004: Recuperar un elemento owl:equivalentClass	
Descripción	Se comprueba que el sistema recupera correctamente las clases equivalentes incluidas en la ontología. A fin de seguir como es la secuencia de acciones se puede ver lo siguiente: 1. Se accede al formulario principal de la aplicación. 2. Se selecciona el archivo .owl a procesar mediante la utilización del botón “Buscar” 3. Se pulsa el botón “Procesar”
Resultado	El proceso termina con un mensaje de éxito incluyendo información sobre la cantidad de elementos recuperados, que serán una generalización con un comentario de tipo “{Complete}”. Además, podemos ver la clase incluida en el modelo UML inspeccionando el archivo generado con extensión “.xmi”
Requisitos relacionados	RSIS-001, RSIS-002, RSIS-025

Tabla 96. PA-004 Recuperar un elemento owl:equivalentClass

PA-005: Recuperar un elemento owl:oneOf	
Descripción	Se comprueba que el sistema recupera correctamente la definición de una clase de acuerdo a las instancias. A fin de seguir como es la secuencia de acciones se puede ver lo siguiente: 1. Se accede al formulario principal de la aplicación. 2. Se selecciona el archivo .owl a procesar mediante la

	utilización del botón “Buscar” 3. Se pulsa el botón “Procesar”
Resultado	El proceso termina con un mensaje de éxito incluyendo información sobre la cantidad de elementos recuperados, que serán la clase y las instancias que pertenecientes a esta clase, relacionadas con una asociación. Además, podemos ver la clase incluida en el modelo UML inspeccionando el archivo generado con extensión “.xmi”
Requisitos relacionados	RSIS-001, RSIS-19, RSIS-24

Tabla 97. PA-005 Recuperar un elemento owl:oneOf

PA-006: Recuperar un elemento owl:maxCardinality	
Descripción	Se comprueba que el sistema recupera correctamente la cardinalidad máxima definida en las relaciones correspondientes. A fin de seguir como es la secuencia de acciones se puede ver lo siguiente: 1. Se accede al formulario principal de la aplicación. 2. Se selecciona el archivo .owl a procesar mediante la utilización del botón “Buscar” 3. Se pulsa el botón “Procesar”
Resultado	El proceso termina con un mensaje de éxito incluyendo información sobre la cantidad de elementos recuperados, que serán las clases y la relación con la cardinalidad máxima extraída, para el rango de la relación. Además, podemos ver los elementos recuperados en el modelo UML inspeccionando el archivo generado con extensión “.xmi” y el archivo con los elementos RSHP “.the”

Requisitos relacionados	RSIS-001, RSIS-013, RSIS-10

Tabla 98. PA-006 Recuperar un elemento owl:maxCardinality

PA-007: Recuperar un elemento owl:maxCardinality	
Descripción	Se comprueba que el sistema recupera correctamente la cardinalidad máxima definida en las relaciones correspondientes. A fin de seguir como es la secuencia de acciones se puede ver lo siguiente: 1. Se accede al formulario principal de la aplicación. 2. Se selecciona el archivo .owl a procesar mediante la utilización del botón “Buscar” 3. Se pulsa el botón “Procesar”
Resultado	El proceso termina con un mensaje de éxito incluyendo información sobre la cantidad de elementos recuperados, que serán las clases y la relación con la cardinalidad máxima extraída, para el rango de la relación. Además, podemos ver los elementos recuperados en el modelo UML inspeccionando el archivo generado con extensión “.xmi” y el archivo con los elementos RSHP “.the”
Requisitos relacionados	RSIS-001, RSIS-013, RSIS-10

Tabla 99. PA-007 Recuperar un elemento owl:maxCardinality

PA-008: Recuperar instancias	
Descripción	Se comprueba que el sistema recupera correctamente las instancias definidas en la ontología. A fin de seguir como es la secuencia de acciones se puede ver lo siguiente: 1. Se accede al formulario principal de la aplicación. 2. Se selecciona el archivo .owl a procesar mediante la utilización del botón “Buscar” 3. Se pulsa el botón “Procesar”
Resultado	El proceso termina con un mensaje de éxito incluyendo información sobre la cantidad de elementos recuperados, que serán las clases definidas y las instancias correspondientes a las mismas, siendo éstas también clases en el modelo recuperado. Además, podemos ver los elementos recuperados en el modelo UML inspeccionando el archivo generado con extensión “.xmi” y el archivo con los elementos RSHP “.the”
Requisitos relacionados	RSIS-001, RSIS-013, RSIS-019, RSIS-024

Tabla 100. PA-008 Recuperar instancias

4 Diseño

Realizaremos en este apartado el diseño del sistema, a partir de los requisitos y modelos reflejados en el apartado de Análisis.

4.1 Introducción

En este apartado se profundizará sobre los elementos analizados en el apartado de Análisis. Viendo los requisitos definidos en el documento mencionado, se realizará una descripción más detallada sobre los elementos del sistema que llevarán a cabo las correspondientes tareas.

Este sistema diseñado será descrito de tal manera que pueda ser también verificado de acuerdo a los requisitos anteriormente definidos.

4.2 Definición de arquitectura del sistema

En este apartado se define la arquitectura del sistema, teniendo en cuenta otros sistemas con los que trabaja este módulo. Además se describen las excepciones analizadas en el módulo, así como el formato y normas seguidas para la implementación.

4.2.1 Definiciones de niveles de arquitectura

El sistema a desarrollar estará basado en una arquitectura de tres capas. Esta arquitectura nos permite tener una separación entre los distintos módulos, que aportará mayor facilidad de cambios futuros en el sistema.

Capa de presentación (PL, Presentation Layer): Es la capa con la que interactúa el usuario. Captura la información del usuario y se comunica con la capa de negocio.

Capa de negocio (BLL, Business Logic Layer): Es en esta capa donde se ejecutan las reglas y funciones necesarias para procesar las solicitudes de la capa de presentación, representando la lógica del negocio.

Capa de datos (DAL, Data Access Logic): En esta capa se realizan las operaciones necesarias con la base de datos, de acuerdo a las solicitudes de la capa de negocio.

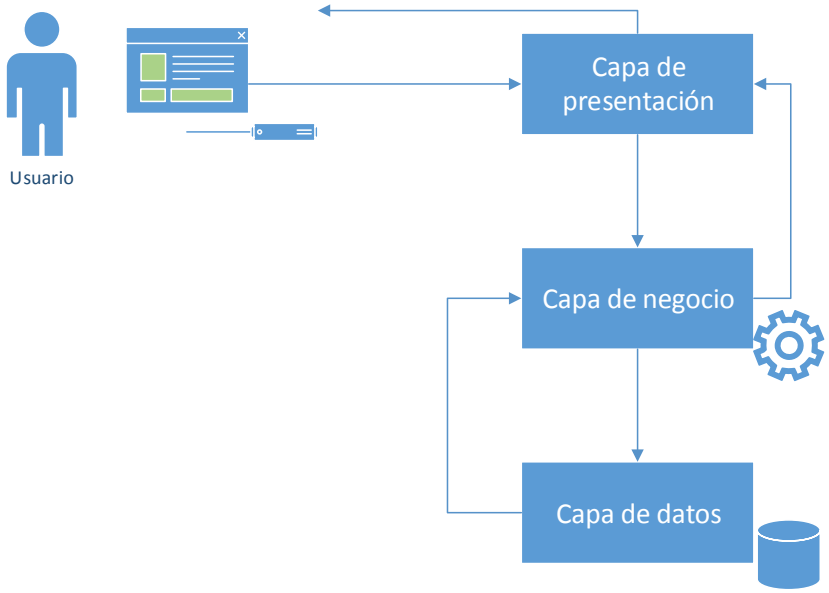


Ilustración 42. Representación de capas

4.2.2 Especificación de excepciones

Se han identificado las posibles excepciones que se podrían dar en el sistema, a fin de gestionar este tipo de fallos que pueden darse en el software.

Por un lado podemos encontrar excepciones originadas en la capa de negocio, siendo recogidas y almacenadas en un fichero de log para su análisis. Estas excepciones estarían originadas en las funciones o reglas utilizadas en esta capa, los mensajes originados en las excepciones así como la localización en el código de la instrucción donde se ha producido el error se incluirá en el fichero de Log.

Por otra parte, considerando las excepciones que pueden darse en la capa de presentación, pueden ser debidas a errores por la interacción de dicha capa y el usuario, debido a datos u opciones no consideradas. Estas excepciones serán tratadas mediante la presentación de mensajes de información o advertencia al usuario.

4.2.3 Especificación de estándares y normas de diseño y construcción

Para la implementación del sistema se siguen una serie de normas necesarias para facilitar la reutilización y comprensión del código por otros desarrolladores, guardando cierto estilo y homogeneidad con el resto de librerías incluidas en el proyecto UMLModels. Por tanto, se describen a continuación los siguientes elementos:

- **Atributos:** son escritos en minúscula de forma general. En caso de estar compuestos por más de una palabra se utilizará las segundas y sucesivas palabras con la primera letra en mayúscula.
- **Constantes:** e escriben en mayúsculas, y en caso de estar compuestas por más de una palabra éstas son separadas por guiones bajos.
- **Funciones:** se escribirá en mayúscula la primera letra de cada palabra que forma el nombre de la función, y el resto de letras en minúscula.
- **Clases:** son escritas con la primera letra de cada palabra utilizada en mayúscula y el resto de letras en minúscula.

Se deberán incluir todos los comentarios que el desarrollador estime oportunos para facilitar la comprensión y posterior utilización del código por otros desarrolladores. Además, cualquier norma que no esté recogida en este apartado podrá ser implementada de acuerdo a las necesidades del desarrollador.

4.3 Diseño de la arquitectura de diseño

Aunque el sistema diseñado en este proyecto no es de gran tamaño, se pueden separar algunos de sus elementos a fin de mejorar el análisis y la construcción del mismo. Uno de los objetivos fundamentales es permitir la escalabilidad de la aplicación, y por tanto su modificación y mejoras en el futuro. Por ello, es necesario definir una arquitectura que guarde coherencia entre los distintos subsistemas y además permita la modificación de los distintos elementos de este módulo del sistema UMLModels.

4.3.1 Diseño de subsistemas de diseño

El objetivo de este apartado es especificar de forma detallada y formal los distintos subsistemas de los que está compuesto el módulo diseñado en este proyecto. Cada uno de estos subsistemas tendrá unas tareas concretas en el sistema, dependiendo además de otros módulos diseñados en el proyecto UMLModels.

Todos estos elementos externos al módulo desarrollado en el proyecto actual de recuperación de información desde OWL a RSHP, serán utilizados como librerías que son necesarias para conseguir los objetivos del proyecto.

A continuación se observa en el diagrama de componentes, las distintas librerías utilizadas por la aplicación UMLModels para llevar a cabo las tareas de recuperación de información desde los lenguajes que puede procesar actualmente.

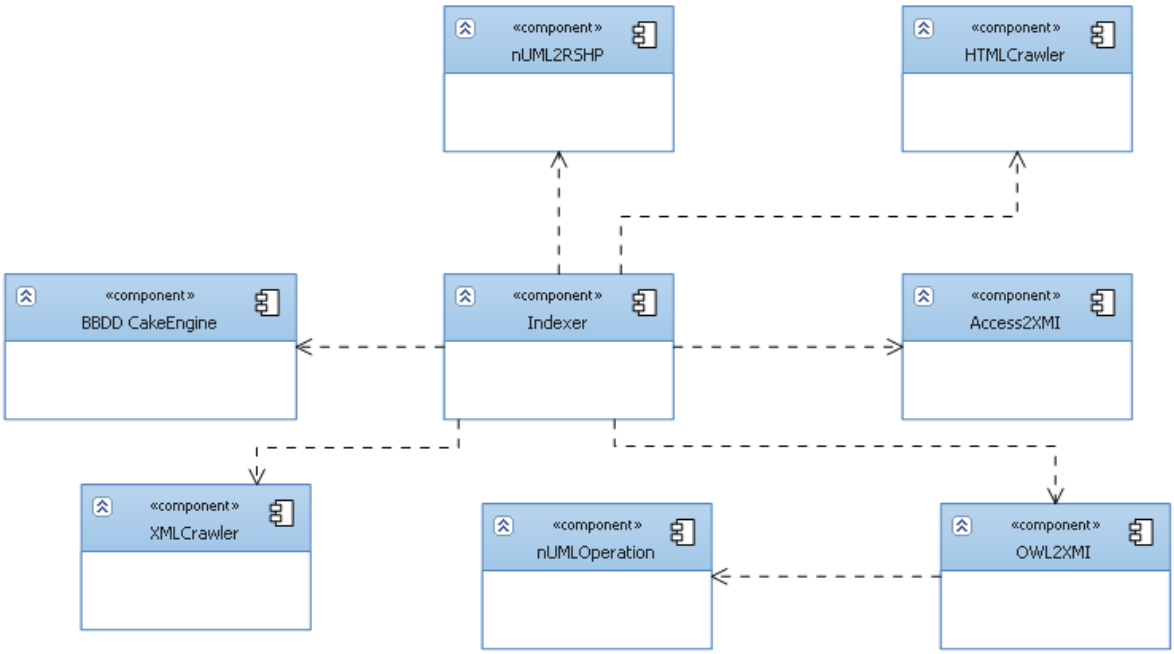


Ilustración 43. Diseño de componentes del sistema

Estos elementos descritos en el apartado anterior son considerados como caja negra, ya que no es el propósito del proyecto actual describirlos o diseñarlos. La nueva librería que será diseñada en este proyecto será incluida dentro del "indexer" siendo utilizada por esta aplicación para las tareas de recuperación de información desde OWL a XMI. Además, en este proyecto se extiende la funcionalidad, dando la posibilidad de transformar el modelo a RSHP (utilizando nUML2RSHP).

Podemos describir los siguientes componentes en el sistema mostrado:

- **BBDD CakeEngine**: esta librería se encarga de utilizar la base de datos de la que se extraen los elementos necesarios para identificar la transformación de los elementos UML en RSHP.
- **Indexer**: este elemento es utilizado para realizar las operaciones de indexing, interactuando con la BBDD de Cake Engine.
- **OWL2XMI**: esta librería es la desarrollada en este proyecto, para el tratamiento de los elementos OWL y su inserción en modelos UML (XMI) utilizando otras librerías ya existentes para realizar las operaciones necesarias.
- **XMLCrawler**: este elemento realiza las búsquedas web a fin de poder realizar las tareas de indexing.
- **nUMLOperation**: esta librería es utilizada para insertar los elementos ya normalizados en modelos UML (XMI).
- **nUML2RSHP**: esta libraría se encarga de la transformación de modelos UML (XMI) en modelos RSHP.
- **HTML Crawler**: aplicación utilizada para las operaciones de búsqueda en archivos HTML.
- **Access2XMI**: esta libraría realiza las transformaciones de bases de datos MS Access en modelos UML (XMI).

4.4 Diseño de clases

Continuando con las clases identificadas en la fase de análisis, se muestran a continuación las clases necesarias para el diseño del sistema:

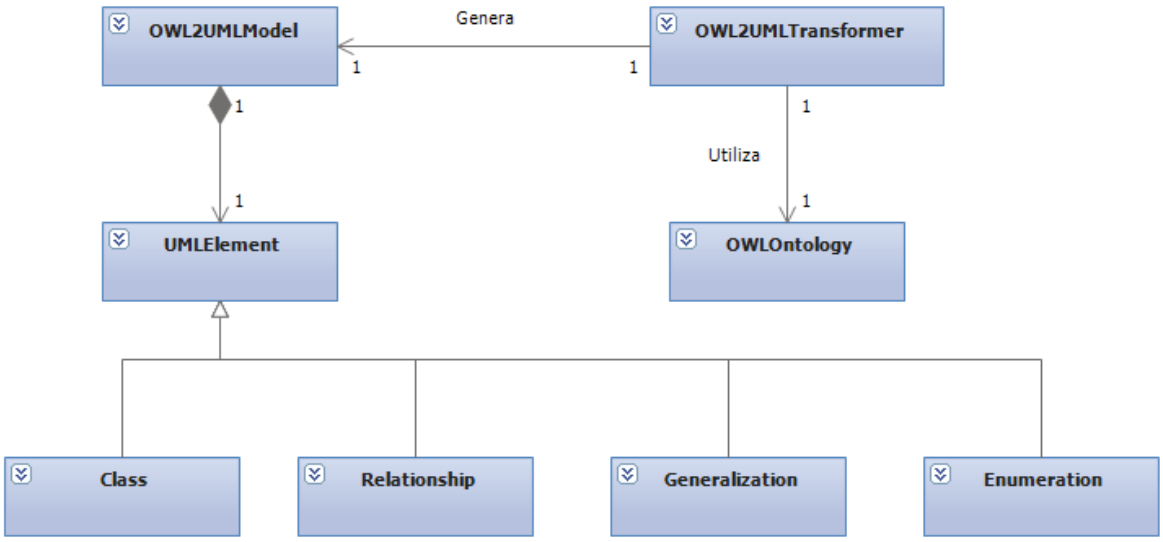


Ilustración 44. Diagrama de clases

4.4.1 Identificación de clases

Se describen a continuación las distintas clases utilizadas en el diseño, teniendo en cuenta la separación en capas utilizada para diseñar el sistema:

- **Clases de entidad:**

  - **UmlElement:** esta clase tiene los atributos identificados como básicos para cada uno de los elementos UML del modelo recuperado.

  - **Class:** esta clase tiene los atributos necesarios para identificar una clase en el modelo UML.

  - **Relationship:** esta clase tiene los atributos necesarios para identificar las relaciones extraídas de una ontología OWL.

  - **Generalization:** esta clase contiene los atributos utilizados para definir una generalización/especialización en UML.

  - **Enumeration:** Esta clase tiene los atributos necesarios para representar una enumeración en UML.

  - **Attribute:** Esta clase tiene los atributos necesarios para incluir la información sobre un atributo.

- **Clases de interfaz:**

  - **OwlOntology**: Esta clase contiene los atributos y funciones necesarios para la validación de los documentos OWL.

  - **ProccessForm**: Esta clase contiene los atributos y funciones necesarios para incluir mediante mensajes al usuario.

- **Clases de control**:

- **Owl2UmlTransformer**: Esta clase contiene los atributos y funciones necesarios para controlar la lógica del negocio utilizada en la aplicación, con el fin de recuperar ontologías OWL en modelos UML y RSHP.

4.4.2 Especificación de clases

En este apartado se describen de forma detallada las clases utilizadas en el diseño, mostrando los atributos y funciones necesarios:

UmlElement	
Descripción:	Clase que es utilizada para identificar los elementos básicos necesarios para la representación de elementos UML individuales.
Atributos.	- **public Type:** Este atributo nos sirve para identificar a qué tipo de elemento UML es de forma individual un elemento. - **public Id:** Este atributo se utiliza para indicar un identificador único entre todos los elementos tratados. - **public Name:** Se utiliza este atributo para insertar el nombre textual en caso que exista para el elemento UML.
Operaciones:	

Tabla 101. Descripción de clase UmlElement

Class	
Descripción:	Esta clase es utilizada para representar los elementos UML que se corresponden con una clase en OWL.
Atributos.	- public List<Attribute> Atrib_class Este atributo almacena todos los atributos que puede contener la clase en UML. - public string Superclass; Este atributo sirve para identificar la superclase de una determinada clase, en caso que exista.
Operaciones:	

Tabla 102. Descripción de clase (Modelo UML)

Relationship	
Descripción:	Esta clase sirve para representar todos los tipos de relaciones que pueden ser recuperadas en OWL, como pueden ser ObjectProperty.
Atributos.	- public string RelationshipName; Este atributo identifica el nombre de la relación tratada. - public string Domain; Este atributo identifica el nombre de la clase dominio de la relación. - public string Range; Este atributo identifica el nombre de la clase rango de la relación. - public string Min_cardin; En este atributo se guarda la cardinalidad mínima de la relación si existe. - public string Max_cardin; En este atributo se guarda la cardinalidad máxima de la relación si existe.

	- public string Cardin; En este atributo se guarda la cardinalidad exacta de la relación si existe. - public string parent_property; Este atributo sirve para almacenar la propiedad de la que hereda elementos la relación actual. - public string type; Este atributo se utiliza para identificar el tipo de relación de acuerdo a la propiedad OWL. - public string second_role; Atributo que sirve para especificar la navegabilidad de la relación - public string first_role; Atributo que sirve para especificar la navegabilidad de la relación.
Operaciones:	

Tabla 103. Descripción de clase relación

Generalization	
Descripción:	Esta clase es utilizada para representar los elementos UML que son una generalización entre dos clases.
Atributos.	- public string Superclass: Este atributo representa la superclase en la generalización. - public string SubClass: Este atributo representa la subclase en la generalización.
Operaciones:	

Tabla 104. Descripción de clase generalización

Instance	
Descripción:	Esta clase es utilizada para representar los elementos UML que en OWL son instancias, y son reconocidos en el modelo UML como clases.
Atributos.	- public string parentClass: Nos sirve para identificar la clase que ha instanciado este elemento.
Operaciones:	

Tabla 105. Descripción de clase instancia

Attribute	
Descripción:	Esta clase es utilizada para representar los atributos UML, recogiendo distintos datos de los mismo desde OWL.
Atributos.	- public string name; Representa el nombre del elemento atributo en UML - public string type; En este atributo se guarda el tipo de dato (String, Integer, etc..) extraído de un DatatypeProperty en OWL - public string value; Se guarda en el valor del atributo especificado en caso que exista
Operaciones:	

Tabla 106. Descripción de clase atribut

Enumeration	
Descripción:	Esta clase es utilizada para representar los elementos tipo enumeración en UML.
Atributos.	- List<Instance> Enumeration; Lista de las instancias guardadas en la enumeración, que en modelo UML tendrán una relación clase instancia.
Operaciones:	

Tabla 107. Descripción de clase enumeración

OwlOntology	
Descripción:	Esta clase es utilizada para representar la estructura básica que identifica los elementos esenciales en la definición de ontologías OWL.
Atributos.	- public string rdf_format; Atributo para identificar el espacio de nombres de rdf - public string rdfs_format; Atributo para identificar el espacio de nombres de rdf - public string xsd_format; Atributo para identificar el espacio de nombres de XML Schema - public string owl_format; Atributo para identificar el espacio de nombres de la especificación OWL - public string xmlns_format; Namespace para identificar el espacio de nombres del documento actual - public string base_format; Atributo para identificar con un nombre la ontología actual
Operaciones:	

Tabla 108. Descripción de clase ontología

Owl2UmlTransformer	
Descripción:	Esta clase es utilizada para incluir toda la lógica del negocio necesaria para procesar los elementos OWL y recuperarlos en un modelo intermedio UML.
Atributos.	
Operaciones:	- private bool IsOwlFile(Stream stream): Verifica el archivo que contiene la ontología OWL de acuerdo a las recomendaciones del W3C para continuar con el procesamiento de dicha ontología. - private static void GetRelatioshipsAndClasses(List<string> rel_list, List<ClassType> class_list, ref List<Relationship> relationship_list, XmlNodeList rshpNodeList, Model umlModel, string IDTypeClass, string resourceTypeClass, ref string relName, ref string className, ref string className2, string cardinality, string min_card, string max_card): Extrae los elementos de que se compone un ObjectProperty, identificando también si tiene alguna propiedad padre. - private static void ObjectProperty_inheritance(List<ClassType> class_list, List<Relationship> relationship_list, XmlNodeList subproperty_list): Se procesan las relaciones que heredan en elementos typo “subPropertyOf”. - private static void FunctionalProperty(List<Relationship> relationship_list, XmlNodeList property_type): tratamos las relaciones de tipo “FunctionalProperty”, para tener en cuenta las restricciones sobre cardinalidad. Se incluye el tipo correspondiente en la relación del modelo intermedio. - private static void GetClassRestrictions(List<string> rel_list, List<ClassType> class_list, ref List<Relationship> relationship_list, XmlNodeList restrictions, string OwlClassAttributeName, string IDTypeClass, string AboutTypeClass, ref string cardinality, string min_card, string max_card, Model umlModel): Se clasifican las restricciones tratadas en cada nodo y su correspondiente element OWL, sobre cardinalidad,

valor en instancias, y restricciones de valor (allValues).

- private static void AddClassesToNUmlModel(List<ClassType> class_list, XmlNodeList classList, Model umlModel, string IDTypeClass, string AboutTypeClass): Se añaden las clases de la ontología tanto al modelo intermedio, como al modelo UML utilizado en UMLModels.

- private static void AddSubClassesToNUmlModel(List<ClassType> class_list, XmlNodeList subclases, Model umlModel, string OwlClassAttributeName, string IDTypeClass, string AboutTypeClass, string resourceTypeClass): Se añaden las subclases al modelo intermedio y al modelo utilizado en UMLModels. Además, se insertan los elementos de generalización en el modelo, que se tienen en cuenta para la jerarquía y herencia.

- private static void AddIntersectionClassesToNUmlModel(List<ClassType> class_list, XmlNodeList intersectionClassList, Model umlModel, string AboutTypeClass): Se verifica que las clases declaradas en un nodo de tipo intersección son añadidas a los modelos UML.

- private static void AddInstancesToNUmlModel(List<ClassType> class_list, ref List<Relationship> relationship_list, XmlDocument xDoc, string IDTypeClass, ref string relName, string cardinality, string min_card, string max_card, Model umlModel): Se incluyen en el modelo las instancias declaradas en la ontología haciendo referencia a la clase en el nombre del nodo.

- private static void AddUnionClassesToNUmlModel(XmlNodeList unionOfsubClassList, Model umlModel, string OwlClassAttributeName, string IDTypeClass, string AboutTypeClass): Se añaden a los modelos UML los elementos de tipo unionOf en OWL.

- private static void Cardinality_inheritance(List<ClassType> class_list, List<objectprop> relationship_list): Se verfican los casos en que existe herencia de cardinalidad y se aplica dicha herencia.

- private static void Insert_attributes(XmlNodeList attributes_list, List<ClassType> class_list, string IdAttrib, XmlDocument xDoc, string IDdataType, Model umlModel, XmlNodeList classList): Se incluyen los atributos en las clases teniendo en cuenta las distintas definiciones para los atributos en una

	ontología. - private static void AddOtherClassesToNUmlModel(List<ClassType> class_list, XmlNodeList classList, Model umlModel, string resourceTypeClass, string AboutTypeClass, XmlNodeList equivalent_classList, string comment, string complete, string disjoint, XmlNodeList disjoint_list, XmlNodeList equivalent_classList2): Incluye en el modelo UML las clases que son definidas como equivalentes o disjuntas en OWL. - private static void AddOneOfInstancesToNUmlModel(Model umlModel, string OwlClassAttributeName, string IDTypeClass, string AboutTypeClass, XmlNodeList oneOf_list, List<objectprop> relationship_list, string instance_relationship, List<ClassType> class_list): Inserta en los modelos UML las instancias que están definidas en la definición de la clase mediante los elementos oneOf. - private static void AddEqualityInstancesNodes(Model umlModel, string OwlClassAttributeName, string IDTypeClass, string resourceTypeClass, XmlNodeList sameAs_List, List<objectprop> relationship_list, string sameAs_relationship, List<ClassType> class_list, string AboutTypeClass, XmlNodeList differentFrom_List, List<string> different_List, string DifferentFrom_relationship): Se añaden en los modelos UML las relaciones que son de tipo “sameAs” entre instancias. - private static void AddInverseOf(Model umlModel, string OwlClassAttributeName, string resourceTypeClass, string IDR, XmlNodeList inverseOf_list, List<objectprop> relationship_list, string instance_relationship, List<ClassType> class_list): Se insertan en los modelos UML las relaciones de tipo inverso (inverseOf en OWL). - private static void AddRelationshipsToNUmlModel(List<objectprop> relationship_list, List<objectprop> relationship_added, Model umlModel): Se insertan las relaciones que existen en el modelo intermedio en el modelo utilizdo en UMLModels.

Tabla 109. Descripción de clase OWL2UmlTransformer

4.5 Especificación técnica del plan de pruebas

Se realizarán en este apartado las pruebas unitarias correspondientes a la capa de lógica del negocio, donde se encuentran las funciones que contienen las reglas fundamentales para la recuperación de información de ontologías OWL.

Se analizarán por tanto las funciones y métodos más relevantes para el procesamiento de la ontología, incluyendo algunos resultados de las pruebas.

4.5.1 Especificación del entorno de pruebas

Para la realización de las pruebas se utilizará el mismo equipo y entorno que el utilizado con el desarrollo de la aplicación. Además, para la realización de las pruebas unitarias utilizaremos la aplicación NUnit 2.6.4, con la cual ejecutaremos las pruebas desarrolladas y preparadas en código C#.

4.5.2 Especificación técnica de niveles de prueba

Se realizarán pruebas independientes en función de los elementos a recuperar que son objetivo de las pruebas. Para ello se han preparado una serie de ontologías clasificadas en distintos archivos que contienen los elementos que se quieren recuperar en cada caso haciendo uso de las funciones correspondientes.

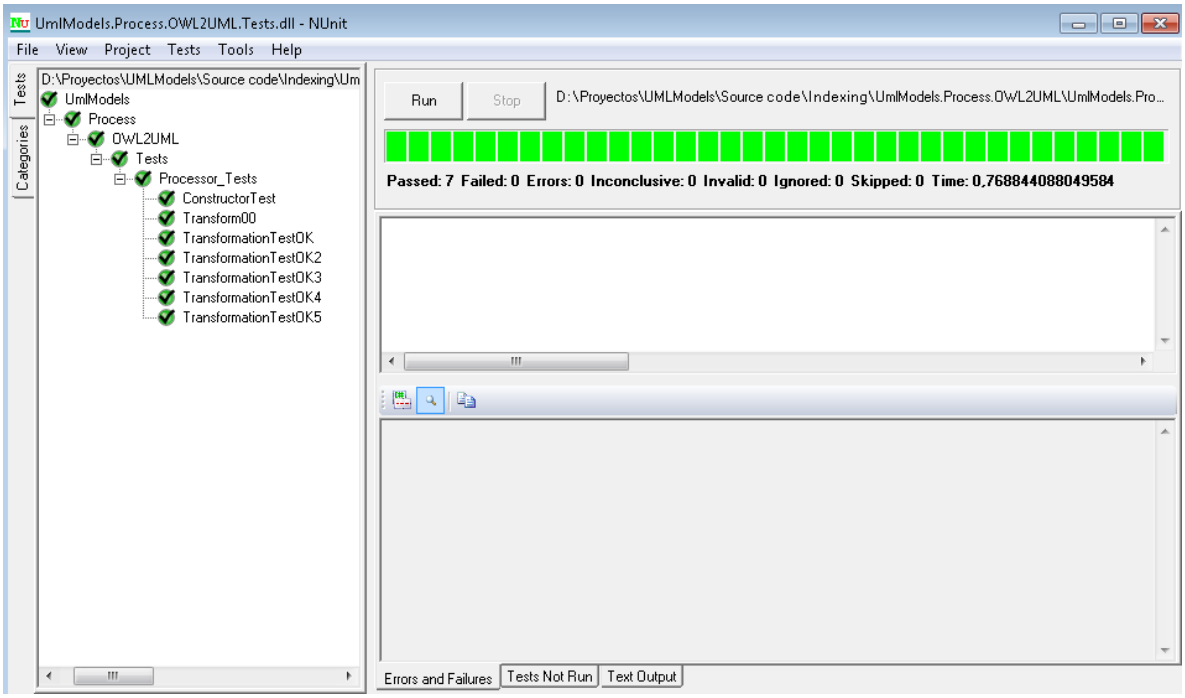


Ilustración 45. Muestra de pantalla de pruebas

PU-001: Recuperación de clases	
Descripción:	Se comprueba la recuperación de todas las clases definidas en la ontología teniendo en cuenta solamente el constructor OWL para definir clases.
Función :	AddClassesToNUmlModel(class_list, classList, umlModel, IDTypeClass, AboutTypeClass);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile1); • Se invoca AddClassesToNUmlModel(class_list, classList, umlModel, IDTypeClass, AboutTypeClass);
Especificaciones de entrada:	• Class_list: lista creada en el modelo intermedio que contiene los elementos tipo clase a tratar. • classList: lista de nodos XML que son seleccionados para tratar el nodo tipo clase. • umlModel: clase perteneciente a NUml que contiene los elementos que son almacenados después de introducirse en la capa de lógica de negocio. • IDTypeClass: Constante para la selección de etiquetas XML. • AboutTypeClass: Constante para la selección de etiquetas XML.

Especificaciones de salida:	Después de llamar a la función se insertan las clases en el modelo NUml.
Produce excepción:	No

Tabla 110. PU-001 Recuperación de clases

PU-002: Recuperación de owl:ObjectProperty	
Descripción:	Se comprueba la recuperación de todas las relaciones definidas en la ontología teniendo en cuenta solamente el constructor owl:ObjectProperty.
Función :	GetRelatioshipsAndClasses(rel_list, class_list, ref relationship_list, rshpNodeList, umlModel, IDTypeClass, resourceTypeClass, ref relName, ref className, ref className2, cardinality, min_card, max_card);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile2); • Se invoca GetRelatioshipsAndClasses(rel_list, class_list, ref relationship_list, rshpNodeList, umlModel, IDTypeClass, resourceTypeClass, ref relName, ref className, ref className2, cardinality, min_card, max_card);
Especificaciones de entrada:	• rel_list: lista de relaciones en el modelo intermedio para almacenar los dominios y rangos tratados. • classList: lista de clases del modelo intermedio con las clases almacenadas y tratadas. • relationship_list: lista del modelo intermedio donde se almacenan todos los objetos de tipo “ObjectProperty” tratados. • rshpNodeList: Lista de nodos XML donde se tienen todos los elementos “ObjectProperty” de la ontología. • umlModel: clase perteneciente a NUml que contiene los elementos que son almacenados después de introducirse en la capa de lógica de negocio. • IDTypeClass: Constante para la selección de etiquetas XML.

	• resourceTypeClass: Constante para la selección de etiquetas XML. • relName: Nombre de la relación recuperada. • className: Nombre del dominio recuperado. • className2: Nombre del rango recuperado.
Especificaciones de salida:	Después de llamar a la función se insertan en el modelo UML intermedio los elementos fundamentales del nodo "ObjectProperty", siendo éstos el nombre de la relación, el dominio y el rango.
Produce excepción:	No

Tabla 111. PU-002 Recuperación de owl:objectPeroperty

PU-003: Recuperación de rdfs:subClassOf	
Descripción:	Se comprueba la recuperación de todas las jerarquías de clases y subclases contenidas en la ontología.
Función :	AddSubClassesToNUmlModel(class_list, subclases, umlModel, OwlClassAttributeName, IDTypeClass, AboutTypeClass, resourceTypeClass);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile3); • Se invoca AddSubClassesToNUmlModel(class_list, subclases, umlModel, OwlClassAttributeName, IDTypeClass, AboutTypeClass, resourceTypeClass);
Especificaciones de entrada:	• class_list: lista del modelo intermedio que contiene todas las clases de la ontología recuperada • subclases: Lista de nodos XML que contienen todas las subclases definidas en la ontología • umlModel: clase perteneciente a NUml que contiene los elementos que son almacenados después de introducirse en la capa de lógica de negocio. • IDTypeClass: Constante para la selección de etiquetas XML. • resourceTypeClass: Constante para la selección de etiquetas XML. • OwlClassAttributeName: Constante para la selección de etiquetas XML. • AboutTypeClass: Constante para la selección de etiquetas XML.

Especificaciones de salida:	Después de llamar a la función se insertan las clases y subclases que forman jerarquías en el modelo UML.
Produce excepción:	No

Tabla 112. PU-003 Recuperación de rdfs:subClassOf

PU-004: Recuperación de owl:oneOf	
Descripción:	Se comprueba la recuperación de todas las jerarquías de clases y subclases contenidas en la ontología.
Función :	AddOneOfInstancesToNUmlModel(umlModel, OwlClassAttributeName, IDTypeClass, AboutTypeClass, oneOf_list, relationship_list, instance_relationship, class_list);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile4); • Se invoca AddOneOfInstancesToNUmlModel(umlModel, OwlClassAttributeName, IDTypeClass, AboutTypeClass, oneOf_list, relationship_list, instance_relationship, class_list);
Especificaciones de entrada:	• umlModel: clase perteneciente a NUml que contiene los elementos que son almacenados después de introducirse en la capa de lógica de negocio. • OwlClassAttributeName: Constante para la selección de etiquetas XML. • IDTypeClass: Constante para la selección de etiquetas XML. • AboutTypeClass: Constante para la selección de etiquetas XML. • oneOf_list: Lista de nodos XML que contiene todos los elementos del tipo “owl:oneOf” que se tratarán en este caso. • relationship_list: lista del modelo intermedio que contiene los elementos que representan una relación, en este caso relación clase-instancia. • instance_relationship: nombre de la relación de instancia de forma constante en la ontología. • class_list: lista del modelo intermedio que contiene todas las clases de la ontología recuperada.

Especificaciones de salida:	Después de llamar a la función se insertan las relaciones entre la clase y las instancias definidas en el modelo intermedio.
Produce excepción:	No

Tabla 113. PU-004 Recuperación de owl:oneOf

PU-005: Tratamiento de cardinalidad	
Descripción:	Se comprueba la recuperación de todas las jerarquías de clases y subclases contenidas en la ontología.
Función :	ProcessCardinality(XmlNode nodo, List<Relationship> relationship_list, string relName, string className_aux, List<ClassType> class_list);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile5); • Se invoca ProcessCardinality(node, relationship_list, relationship, className_aux, class_list);
Especificaciones de entrada:	• node: nodo que contiene la restricción de cardinalidad. • relationship_list: lista del modelo intermedio que contiene los elementos que representan una relación, en este caso relación clase-instancia. • class_list: lista del modelo intermedio que contiene todas las clases de la ontología recuperada. • relationship: relación del nodo que se está tratando. • className_aux: Clase padre que se considera el dominio de la restricción.
Especificaciones de	Después de llamar a la función se insertan en las relaciones del

salida:	modelo intermedio las restricciones de cardinalidad correspondientes.
Produce excepción:	No

Tabla 114. PU-005 Tratamiento de cardinalidad

PU-006: Recuperación de elementos owl:DatatypeProperty	
Descripción:	Se comprueba la recuperación de los atributos y valores especificados para una clase.
Función :	Insert_attributes(XmlNodeList attributes_list, List<ClassType> class_list, string IdAttrib, XmlDocument xDoc, string IDdataType, Model umlModel, XmlNodeList classList);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile6); • Se invoca Insert_attributes(attributes_list, class_list, IdAttrib, xDoc, IDdataType, umlModel, classList);
Especificaciones de entrada:	• attributes_list: lista de nodos XML de tipo “DatatypeProperty” • class_list: lista del modelo intermedio que contiene todas las clases de la ontología recuperada. • IdAttrib: relación del nodo que se está tratando. • xDoc: Clase que contiene todos los nodos XML de la ontología.
Especificaciones de salida:	Después de llamar a la función se insertan en los atributos, y los tipos de datos y valores, en caso que existan, en las clases correspondientes del modelo intermedio.
Produce excepción:	No

Tabla 115. PU-006 Recuperación de elementos owl:DatatypeProperty

PU-007: Recuperación de instancias	
Descripción:	Se comprueba la recuperación de los atributos y valores especificados para una clase.
Función :	Insert_attributes(XmlNodeList attributes_list, List<ClassType> class_list, string IdAttrib, XmlDocument xDoc, string IDdataType, Model umlModel, XmlNodeList classList);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile6); • Se invoca Insert_attributes(attributes_list, class_list, IdAttrib, xDoc, IDdataType, umlModel, classList);
Especificaciones de entrada:	• attributes_list: lista de nodos XML de tipo "DatatypeProperty" • class_list: lista del modelo intermedio que contiene todas las clases de la ontología recuperada. • IdAttrib: relación del nodo que se está tratando. • xDoc: Clase que contiene todos los nodos XML de la ontología.
Especificaciones de salida:	Después de llamar a la función se insertan en los atributos, y los tipos de datos y valores, en caso que existan, en las clases correspondientes del modelo intermedio.
Produce excepción:	No

Tabla 116. PU-007 Recuperación de instancias

PU-008: Recuperación de elementos owl:inverseOf	
Descripción:	Se comprueba la recuperación de los atributos y valores especificados para una clase.
Función :	Insert_attributes(XmlNodeList attributes_list, List<ClassType> class_list, string IdAttrib, XmlDocument xDoc, string IDdataType, Model umlModel, XmlNodeList classList);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile6); • Se invoca Insert_attributes(attributes_list, class_list, IdAttrib, xDoc, IDdataType, umlModel, classList);
Especificaciones de entrada:	• attributes_list: lista de nodos XML de tipo "DatatypeProperty" • class_list: lista del modelo intermedio que contiene

	todas las clases de la ontología recuperada. • IdAttrib: relación del nodo que se está tratando. • xDoc: Clase que contiene todos los nodos XML de la ontología.
Especificaciones de salida:	Después de llamar a la función se insertan en los atributos, y los tipos de datos y valores, en caso que existan, en las clases correspondientes del modelo intermedio.
Produce excepción:	No

Tabla 117. PU-008 Recuperación de elementos owl:inverseOf

PU-009: Recuperación de relación entre instancias (owl:sameAs; owl:differentFrom)	
Descripción:	Se comprueba la recuperación de los atributos y valores especificados para una clase.
Función :	Insert_attributes(XmlNodeList attributes_list, List<ClassType> class_list, string IdAttrib, XmlDocument xDoc, string IDdataType, Model umlModel, XmlNodeList classList);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile6); • Se invoca Insert_attributes(attributes_list, class_list, IdAttrib, xDoc, IDdataType, umlModel, classList);
Especificaciones de entrada:	• attributes_list: lista de nodos XML de tipo “DatatypeProperty” • class_list: lista del modelo intermedio que contiene todas las clases de la ontología recuperada. • IdAttrib: relación del nodo que se está tratando. • xDoc: Clase que contiene todos los nodos XML de la ontología.
Especificaciones de salida:	Después de llamar a la función se insertan en los atributos, y los tipos de datos y valores, en caso que existan, en las clases correspondientes del modelo intermedio.
Produce excepción:	No

Tabla 118. Recuperación de relación entre instancias (owl:sameAs; owl:differentFrom)

PU-010: Recuperación de elementos owl:DatatypeProperty	
Descripción:	Se comprueba la recuperación de los atributos y valores especificados para una clase.
Función :	Insert_attributes(XmlNodeList attributes_list, List<ClassType> class_list, string IdAttrib, XmlDocument xDoc, string IDdataType, Model umlModel, XmlNodeList classList);
Pasos:	• Se invoca la clase de transformación, metamodel = owlTransformer.Transform(OwlFile6); • Se invoca Insert_attributes(attributes_list, class_list, IdAttrib, xDoc, IDdataType, umlModel, classList);
Especificaciones de entrada:	• attributes_list: lista de nodos XML de tipo "DatatypeProperty" • class_list: lista del modelo intermedio que contiene todas las clases de la ontología recuperada. • IdAttrib: relación del nodo que se está tratando. • xDoc: Clase que contiene todos los nodos XML de la ontología.
Especificaciones de salida:	Después de llamar a la función se insertan en los atributos, y los tipos de datos y valores, en caso que existan, en las clases correspondientes del modelo intermedio.
Produce excepción:	No

Tabla 119. Recuperación de elementos owl:DatatypeProperty

5 Planificación

En este apartado del proyecto se realizará una planificación de las distintas fases en que está compuesto este trabajo. En la planificación y estimación se analizará y valorará si los objetivos y el tiempo disponible para conseguirlos son suficientes y realistas. Además, durante el seguimiento del proyecto serán necesario disponer de una planificación a fin de valorar posibles desviaciones en las distintas etapas y actividades.

Al calcular la estimación podríamos utilizar herramientas o procedimientos como por ejemplo COCOMO II, puntos de función de Albretch. Estas técnicas utilizan distintos datos y elementos, y mediante cálculos matemáticos, y una aplicación disponible para COCOMO II, se realiza la estimación del esfuerzo en personas/mes así como una predicción en costes de acuerdo a las características del proyecto.

Debemos tener en cuenta para este proyecto de recuperación de ontologías OWL en UML y RSHP, que debido a las características y el alcance de dicho proyecto, no resulta necesario utilizar estas técnicas de estimación, ya que el tamaño no resulta grande, y puede realizarse de acuerdo a la experiencia profesional en la realización de aplicaciones y gestión de proyectos realizada en los estudios de Ingeniería Técnica Informática.

Para el proyecto se debe realizar una estimación, basándonos también en la carga de trabajo que corresponde a los créditos ECTS del proyecto de fin de carrera en la titulación ITIG, siendo esta cantidad de unas 180 horas.

Se tendrá en cuenta para la estimación del proyecto, así como para el posterior "Debriefing", que el alumno se ha encontrado trabajando a tiempo completo en una empresa mientras realizaba el PFC, por ello la concentración de las horas de trabajo en los fines de semana. De acuerdo a estas condiciones, se ajustará la jornada de dedicación al proyecto en cantidades inferiores a la jornada completa de dedicación al mismo.

Con el fin de tener una visión global de la planificación para las tareas en las que se ha dividido el proyecto, se presentan las mismas con la cantidad total de horas estimadas, así como las fechas de inicio y fin previstas para cada una de ellas:

Tarea	Horas estimadas	Fecha inicio	Fecha fin
Estado del arte e investigación	65	2/8/2014	27/9/2014
Planificación	4	27/9/204	27/9/2014
Presupuesto	3	27/9/204	27/9/2014
Análisis	56	4/10/2014	22/11/2014

Diseño	32	1/11/2014	20/12/2015
Implementación	140	13/12/2014	25/4/2015
Pruebas	23	14/2/2015	16/5/2015
Seguimiento	9	13/12/2015	16/5/2015
Debriefing	3	16/5/2015	16/5/2015

Tabla 120. Resumen de tareas

Como se ha comentado anteriormente, el proyecto se ha planificado para trabajar en él durante los fines de semana, ya que el alumno se encuentra trabajando a tiempo completo en una empresa. Al considerarse los fines de semana, unos meses se tienen cuatro y en otros casos se tienen cinco, con una jornada de ocho horas. El trabajo en el proyecto se desarrolla a lo largo de un curso académico para compaginar la vida profesional y el trabajo académico que representa el proyecto de fin de carrera.

Mes	Días de trabajo	Jornada laboral (horas/día)	Total horas
Agosto	5	8	40
Septiembre	4	8	32
Octubre	4	8	32
Noviembre	5	8	40
Diciembre	4	8	32
Enero	4	8	32
Febrero	4	8	32
Marzo	5	8	40
Abril	4	8	32
Mayo	3	(*) no competo	23
			335

Tabla 121. Horas por mes

Se muestran a continuación las tareas realizadas clasificadas en meses y semanas:

Agosto					
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**	**Semana 5 (1 día)**
Estado del arte e investigación	8	8	8	8	8

Tabla 122. Planificación agosto

Estimación para agosto: Se comienza el proyecto con la investigación de los temas fundamentales para desarrollar la aplicación planteada, incluyendo también información sobre elementos, aplicaciones y definiciones necesarias para introducir las tecnologías utilizadas en el proyecto. Se comienza durante este mes trabajando sólo en esta tarea a fin de concretar mejor el camino a seguir.

Septiembre				
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**
Estado del arte e investigación	8	8	8	1
Planificación	0	0	0	4
Presupuesto	0	0	0	3

Tabla 123. Planificación septiembre

Estimación para septiembre: En este segundo mes se continúan analizando las tecnologías utilizadas para la realización del proyecto, así como publicaciones que son necesarias para las decisiones que se tomarán posteriormente en otras fases.

Además, se realizan las tareas de planificación y presupuesto.

Octubre				
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**
Análisis	8	8	8	8

Tabla 124. Planificación octubre

Estimación para octubre: En octubre se comienza con la fase de análisis de proyecto, ocupando estas el mes sin llegar a entrar en tareas distintas.

Noviembre					
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**	**Semana 5 (1 día)**
Análisis	6	6	6	6	0
Diseño	2	2	2	2	8

Tabla 125. Planificación noviembre

Estimación para noviembre: Se trabaja en las tareas de la fase de análisis hasta la cuarta semana de noviembre, comenzando la fase de diseño antes de terminar la parte de análisis.

Diciembre				
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**
Diseño	8	4	4	0
Implementación	0	3	4	8
Seguimiento	0	1	0	0

Tabla 126. Planificación diciembre

Estimación para diciembre: En este mes se termina con la fase de diseño, y se comienza con implementación. Se realiza seguimiento del proyecto.

Enero				
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**
Implementación	8	8	8	7
Seguimiento	0	0	0	1

Tabla 127. Planificación enero

Estimación para enero: En este mes se dedica el esfuerzo fundamentalmente a la fase de implementación, realizando también seguimiento del proyecto.

Febrero				
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**
Implementación	8	8	7	8
Pruebas	0	0	1	0

Tabla 128. Planificación febrero

Estimación para febrero: Durante el mes de febrero se continúa con la fase de implementación, siendo ésta la más extensa del proyecto. Además, se comienzan las pruebas durante este mes.

Marzo					
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**	**Semana 5 (1 día)**
Implementación	8	7	7	7	7
Pruebas	0	0	1	1	1

Seguimiento	0	1	0	0	0

Tabla 129. Planificación marzo

Estimación para marzo: Al igual que en el mes anterior, se continúa con la fase de implementación y al mismo tiempo se realizan pruebas en las funcionalidades terminadas en el sistema. Durante este mes también se realiza seguimiento del sistema.

Abril				
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**
Implementación	7	8	8	4
Pruebas	1	0	0	3
Seguimiento	0	0	0	1

Tabla 130. Planificación abril

Estimación para abril: Durante este mes, se continúa con las tareas de desarrollo, realizando también pruebas sobre las funcionalidades realizadas hasta el momento. Se realiza seguimiento del proyecto.

Mayo					
Tarea	**Semana 1 (1 día)**	**Semana 2 (1 día)**	**Semana 3 (1 día)**	**Semana 4 (1 día)**	**Semana 5 (1 día)**
Pruebas	6	6	3	0	0
Seguimiento	2	2	1	0	0
Debrief	0	0	3	0	

Tabla 131. Planificación mayo

Estimación para mayo: Durante este mes, se termina la fase de pruebas, realizándose también el seguimiento final y análisis de desviaciones que hayan ocurrido en el proyecto.

Teniendo en cuenta la planificación mostrada anteriormente, se consideran como hitos finales del proyecto la finalización y entrega del mismo, prevista para el día **16 de mayo de 2015.**

A continuación se muestra el diagrama de Gantt para la planificación realizada:

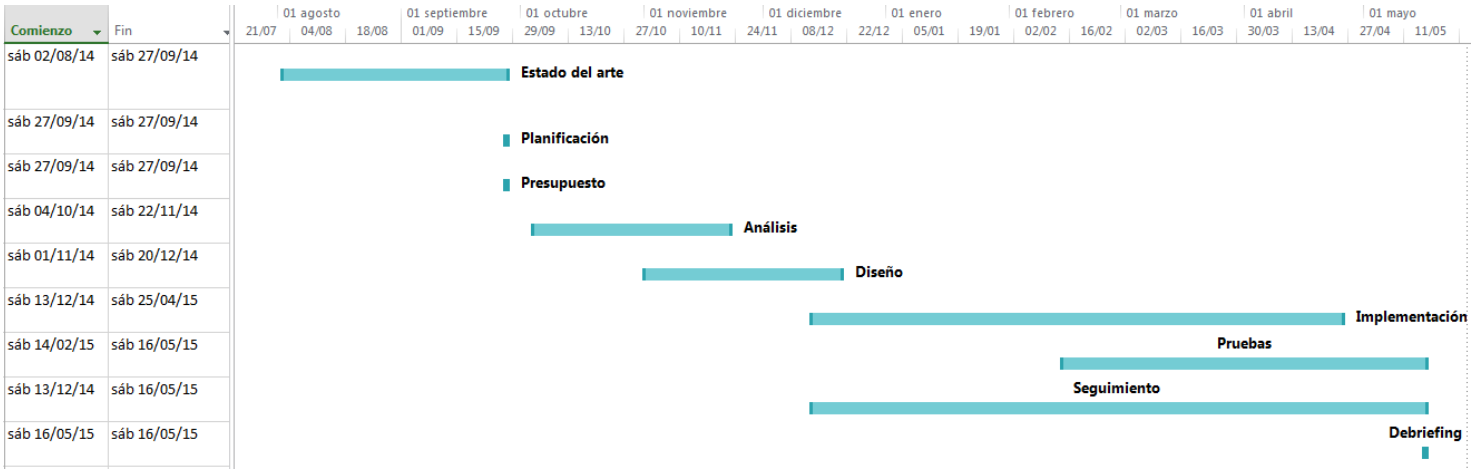


Ilustración 46. Diagrama de Gantt, Planificación

6 Presupuesto

Se realizará en este punto una estimación de los costes basándonos en la planificación de las horas correspondientes a cada tarea, y teniendo otros aspectos en cuenta como pueden ser gastos de material, u otros gastos derivados de la propia actividad que requiere la realización del proyecto.

Se considera que en el proyecto intervendrán distintos roles que serán descritos a continuación. Para el cálculo de los salarios asociados a estos roles se asumen 22 días laborables al mes y 8 horas de trabajo diarias. Para el cálculo exacto del precio de hora de trabajo habría que estimar con respecto al convenio correspondiente. Estos datos son indicados sobre un modelo general para la estimación de los costes laborales, pero como se ha explicado en la planificación, la distribución de horas para la realización del proyecto no será una jornada completa de trabajo.

Rol	Sueldo bruto anual	Sueldo bruto por hora
Jefe de Proyecto	60.000 €	28,40 €
Analista	45.000 €	21,30 €
Programador	36.000 €	17,05 €

Ilustración 47. Presupuesto por roles

Se desglosa a continuación las horas dedicadas por cada uno de los roles involucrados en el proyecto en cada una de las tareas planificadas:

	Jefe de proyecto	Analista	Programador
Estado del arte e investigación	25	40	0
Planificación	4	0	0
Presupuesto	3	0	0
Análisis	25	31	0
Diseño	12	20	0
Implementación	0	0	140
Pruebas	0	0	23
Seguimiento	9	0	0
Debrief	2	1	0
Horas totales rol	80	92	163
Coste total rol	2272 €	1959,60 €	2779,15 €
		Total	7010,15 €

Ilustración 48. Distribución de horas en roles y actividades

Siguiendo con el desglose del presupuesto se incluyen a continuación otros costes relacionados con material necesario, así como servicios utilizados (conexión a internet, etc…), y transporte.

Para el caso del ordenador portátil, con el fin de extraer el coste de la utilización del mismo durante la realización del proyecto se consideran tres años (365 días/año) de utilización del recurso.

Material	Cantidad	Comentarios	Coste
Ordenador portátil	1 (42 días de utilización)	((CostePC/DiasUtiles)* DíasDeUtilización) (800/1095)*42	30,68 €
Conexión a internet	1 línea	(Coste/MesesTotales) * DíasDeUtilización ((50€*10meses)/365)*42	71,21 €
Sistema operativo (Windows 7)	1	Licencia Microsoft	130 €
Paquete Office	1	Licencia Microsoft	109 €
MS Visual Studio	1	Licencia Microsoft	1553 €
Material de oficina	- 7Cd´s - 2 Cuadernos	-	15 €
Transporte	10 meses (42 días)	42días*4€	168 €
Dietas	42 días	Coste*NumDias 6*42	252 €
		Total	2328,89 €

Ilustración 49. Presupuesto para materiales y licencias

En la siguiente tabla se dispone de la información sobre el coste final del proyecto teniendo en cuenta los distintos conceptos aplicables. Se ha considerado un margen de un 13% para los materiales o necesidades del proyecto que no hayan sido identificadas de forma explícita en este documento, así como un margen de riesgo para atender a contingencias que pudieran producirse en el proyecto.

Concepto	Cantidad	Precio	Total
Costes directos del proyecto	1	7010,15 € + 2328,89 €	9339,04 €
Costes indirectos del proyecto (13%)	1	1214,08 €	10553,12 €
Margen de riesgo (8%)	1	844,25 €	11397,37 €
Beneficio (12%)	1	1367,68 €	12765,05 €
IVA (21%)	1	2680,66 €	15445,71€
		TOTAL	**15445,71€**

Ilustración 50. Resumen presupuesto

7 Resultados

En este punto se mostrarán algunos resultados obtenidos mediante la aplicación desarrollada en este proyecto. El objetivo en este caso, es proporcionar una visión global sobre dichos resultados así como una explicación de los mismos a alto nivel que faciliten la comprensión de este trabajo en toda su extensión.

Como se ha podido explicar en otros capítulos de este proyecto, la aplicación procesa como elemento de entrada una ontología OWL, procesándolo para obtener los siguientes elementos de salida:

- **Modelo UML** (archivo .xmi): Las reglas desarrolladas en la aplicación OWL2UML, tienen como objetivo principal procesar la correspondencia entre la ontología definida en lenguaje OWL, y construir el modelo de dicha correspondencia en un modelo de diagrama de clases UML, siendo éste un archivo .XMI. Para obtener este archivo de salida, la aplicación desarrollada ha utilizado librerías diseñadas anteriormente en el área de Knowledge Reuse de la Universidad Carlos III, que realizan la tarea de la construcción del archivo XMI de acuerdo a los estándares correspondientes.

- **Modelo RSHP** (archivo .the): Una vez es obtenido el modelo UML explicado en el paso anterior, la aplicación desarrollada en este proyecto utiliza las librerías de CAKE Engine para realizar la transformación del archivo ".XMI"(UML) en otro ".THE" (RSHP).

- **Imagen UMLModels**: El archivo (.XMI) obtenido inicialmente mediante las reglas diseñas para la transformación OWL2UML, es utilizado por la aplicación UMLModels, que genera los modelos UML de forma visual en una imagen ".JPG".

Se mostrarán a continuación ejemplos que forman parte de la ontología utilizada por el W3C en las distintas guías utilizadas por esta organización (Wine.owl). Se ha decidido utilizar esta ontología para aprovechar la riqueza de casos de que se dispone, y la posibilidad de verificar múltiples ejemplos revisados por el W3C. No se ha utilizado en esta demostración la ontología completa pero sí un conjunto de elementos demostrativos de las transformaciones realizadas.

7.1 Transformación 1: Jerarquías y atributos

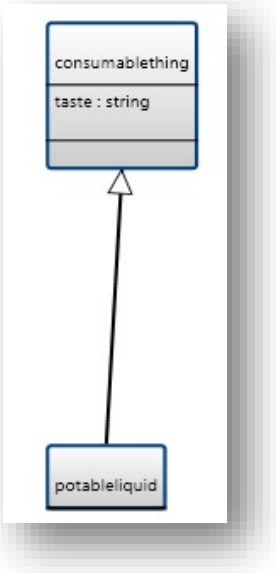
Código OWL

```
<owl:Class rdf:ID="PotableLiquid">
<rdfs:subClassOf rdf:resource="#ConsumableThing" />
</owl:Class>

<owl:DatatypeProperty rdf:ID="Taste">
<rdfs:domain rdf:resource="#ConsumableThing" />
<rdfs:range  rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Código XMI obtenido

```
<?xml version="1.0" encoding="utf-8"?>
<xmi:XMI xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmi:version="2.1"
xmlns:uml="http://schema.omg.org/spec/uml/2.0" xmlns:xmi="http://schema.omg.org/spec/xmi/2.1">
  <xmi:Documentation exporter="http://www.umlmodels.com" exporterVersion="2.0"
exporterID="UMLModels Serializer UML2.0/XMI2.1" owner="UMLModels" shortDescription="Assembly
v2.0.5584.28070" />
  <uml:Model xmi:id="a7ee1110-08b5-40a9-977a-df75b7c85d41" name="mimodelo">
    <ownedMember xmi:type="uml:Class" xmi:id="00c8e427-43ee-4d95-a9b1-60843ec4621f"
name="potableliquid">
      <generalization xmi:id="f5a8d614-ea9b-4b01-aa5c-10615c30e35a" isSubstitutable="false">
        <general xmi:idref="4439b3e7-6c32-40c1-90c9-6ee04e3c7823" />
      </generalization>
    </ownedMember>
    <ownedMember xmi:type="uml:Class" xmi:id="090f87dc-1d60-4a24-b5e3-cb1fa22e9773"
name="consumablething">
      <ownedAttribute xmi:id="0482d666-eb83-4383-a08a-465db4acba98" name="taste : string" />
    </ownedMember>
  </uml:Model>
  <uml:Class xmi:id="4439b3e7-6c32-40c1-90c9-6ee04e3c7823" xmi:href="#090f87dc-1d60-4a24-b5e3-
cb1fa22e9773" />
</xmi:XMI>
```

Diagrama UML Obtenido



Comentarios

Como se puede ver en el ejemplo anterior, se han recuperado correctamente las clases descritas en la ontología. Por otra parte, se han incluido también en este ejemplo, los elementos de tipo “DatatypeProperty” que representan los atributos en la equivalencia UML que hemos establecido, recuperando el nombre del atributo así como el tipo de dato.

7.2 Relaciones y cardinalidad

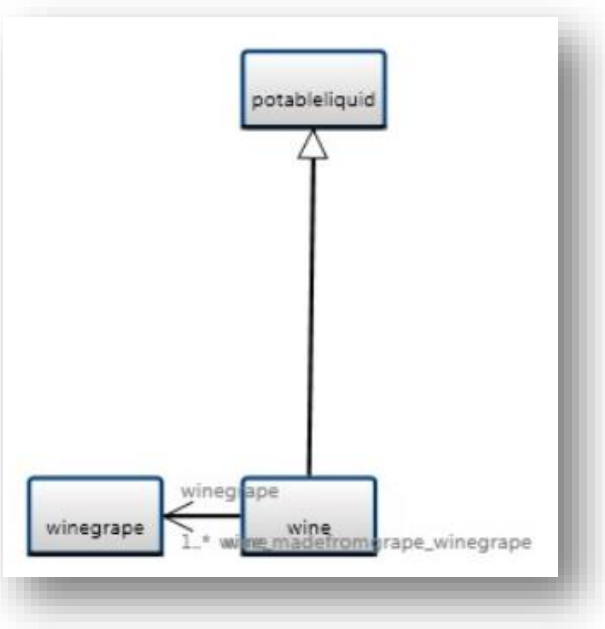
Código OWL

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#madeFromGrape"/>
      <owl:minCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Código XMI obtenido

```
<uml:Model xmi:id="0b18f4bd-ce41-4f7c-b8c8-ddc49433056a" name="mimodelo">
    <ownedMember xmi:type="uml:Class" xmi:id="d74765f9-893c-45b2-99ce-713888a8f955" name="wine">
      <generalization xmi:id="18ddd13b-7d7a-4a5e-9873-f1c0af96d9c2" isSubstitutable="false">
        <general xmi:idref="9a6c992f-f0c6-42e7-8c1a-065070488c23" />
      </generalization>
      <ownedAttribute         xmi:id="4c26c8d6-0622-497e-978b-6ea35fec8f2d"         name="winegrape"
upper="4294967295">
        <type xmi:idref="df55e6c1-f0db-40ed-923d-cca6a49aa8b2" />
        <association xmi:idref="9828f495-0065-4acd-b2e1-366b26907ccc" />
      </ownedAttribute>
    </ownedMember>
    <ownedMember          xmi:type="uml:Class"          xmi:id="d3849af9-5d97-47f3-82e9-640b84a26cbc"
name="winegrape" />
    <ownedMember          xmi:type="uml:Class"          xmi:id="0edea4c0-0373-4396-863a-46a07fd55ce3"
name="potableliquid" />
    <ownedMember       xmi:type="uml:Association"       xmi:id="a4779ca7-6163-4bb9-9b83-5c809f4dc1f5"
name="wine_madefromgrape_winegrape">
      <memberEnd xmi:idref="4c26c8d6-0622-497e-978b-6ea35fec8f2d" />
      <ownedEnd    xmi:id="da05d17e-c85d-4648-a21a-58d0e5b5125b"    name="wine"    lower="4294967294"
upper="4294967294">
        <type xmi:idref="d74765f9-893c-45b2-99ce-713888a8f955" />
        <association xmi:idref="a4779ca7-6163-4bb9-9b83-5c809f4dc1f5" />
      </ownedEnd>
    </ownedMember>
  </uml:Model>
  <uml:Association    xmi:id="9828f495-0065-4acd-b2e1-366b26907ccc"    xmi:href="#a4779ca7-6163-4bb9-
9b83-5c809f4dc1f5" />
  <uml:Class    xmi:id="9a6c992f-f0c6-42e7-8c1a-065070488c23"    xmi:href="#0edea4c0-0373-4396-863a-
46a07fd55ce3" />
  <uml:Class    xmi:id="df55e6c1-f0db-40ed-923d-cca6a49aa8b2"    xmi:href="#d3849af9-5d97-47f3-82e9-
640b84a26cbc" />
</xmi:XMI>
```

Diagrama UML Obtenido



Comentarios

En este caso se realiza la transformación del elemento ObjectProperty desde OWL, obteniendo la relación entre dos clases. Además se ha obtenido la cardinalidad mínima especificada en la deficinición de la clase Wine siendo ésta incluida en el modelo mostrado. La relación en este caso, es necesariamente unidireccional siendo así especificado en las propiedades “ObjectProperty” de OWL. Aunque en este caso se muestra la transformación para la cardinalidad mínima, el sistema está diseñado para pode realizarlo también para la máxima y exacta.

Todas las relaciones entre clases, a fin de ser almacenadas como únicas como requisito del sistema, tienen un formato Clase1_nombreRelacion_Clase2 no pudiéndose repetir en el modelo UML al que pertenecen.

7.3 Herencia en propiedades, propiedad funcional, intersección y restricciones de valor

Código OWL

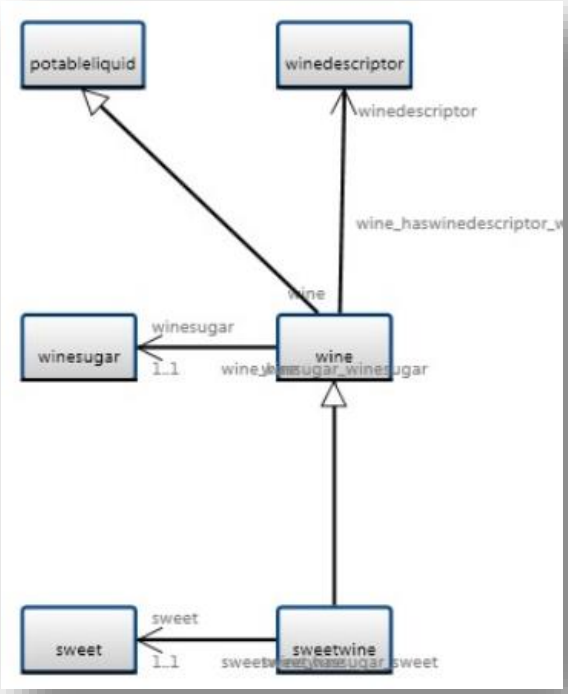
```
 <owl:Class rdf:ID="SweetWine">
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Wine" />
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasSugar" />
        <owl:hasValue rdf:resource="#Sweet" />
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>

    <owl:ObjectProperty rdf:ID="hasWineDescriptor">
    <rdfs:domain rdf:resource="#Wine" />
    <rdfs:range  rdf:resource="#WineDescriptor" />
  </owl:ObjectProperty>

 <owl:ObjectProperty rdf:ID="hasSugar">
    <rdf:type rdf:resource="&owl;FunctionalProperty" />
    <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
    <rdfs:range rdf:resource="#WineSugar" />
  </owl:ObjectProperty>
```

Código XMI obtenido

```
<uml:Model xmi:id="208b8eac-7806-4807-8e1a-06c57c50ce80" name="mimodelo">
    <ownedMember xmi:type="uml:Class" xmi:id="0871b6ee-2d44-4393-957e-4a6b899f9d95" name="wine">
      <generalization xmi:id="01e8a057-caa4-43b6-be7e-cabd8aef9d41" isSubstitutable="false">
        <general xmi:idref="13e3a91e-cc45-4aad-8209-8e742ccda880" />
      </generalization>
      <ownedAttribute xmi:id="c80c1d08-0c17-424c-878f-9c3168d7615d" name="winedescriptor"
lower="4294967294" upper="4294967294">
        <type xmi:idref="c0f91075-aca2-4b2a-aae0-eeb41cf74c0a" />
        <association xmi:idref="09416946-ac67-4982-9e8a-9b33c8280f68" />
      </ownedAttribute>
      <ownedAttribute xmi:id="8d5032bf-cdda-43a7-98ac-af3d262bd2e9" name="winesugar">
        <type xmi:idref="f0805d5a-29cc-449e-b8eb-d5938aeb792d" />
        <association xmi:idref="50d5f472-3b04-4ec9-8e91-f4d6d972b27c" />
      </ownedAttribute>
    </ownedMember>
    <ownedMember xmi:type="uml:Class" xmi:id="e74f7a8c-b3f6-479a-a39e-526886870766"
name="winedescriptor" />
    <ownedMember xmi:type="uml:Class" xmi:id="bcf703bd-b72c-4439-82e2-9a9b878441ce"
name="winesugar" />
    <ownedMember xmi:type="uml:Class" xmi:id="7d7c5ade-e7aa-444c-ade3-c29c6f3e4c05"
name="sweetwine">
      <generalization xmi:id="a2b8dfb8-0eca-43ba-af3a-48ff27ef5212" isSubstitutable="false">
        <general xmi:idref="0871b6ee-2d44-4393-957e-4a6b899f9d95" />
      </generalization>
      <ownedAttribute xmi:id="f3253d52-1b21-4e6e-ad3b-250560e5d7c0" name="sweet">
        <type xmi:idref="b8fda854-ffde-43bd-86ad-2b74639148e6" />
        <association xmi:idref="3c6b4fbd-89c1-472d-9f55-18398410b901" />
      </ownedAttribute>
    </ownedMember>
    <ownedMember xmi:type="uml:Class" xmi:id="9bdf278d-f701-470d-9529-7f56e017d973"
name="potableliquid" />
    <ownedMember xmi:type="uml:Association" xmi:id="57a3f860-7def-4972-906a-d13739ed7ca0"
name="wine_haswinedescriptor_winedescriptor">
      <memberEnd xmi:idref="c80c1d08-0c17-424c-878f-9c3168d7615d" />
      <ownedEnd xmi:id="f43e7677-2749-4616-9b5d-5cc75869133f" name="wine" lower="4294967294"
upper="4294967294">
        <type xmi:idref="0871b6ee-2d44-4393-957e-4a6b899f9d95" />
        <association xmi:idref="57a3f860-7def-4972-906a-d13739ed7ca0" />
      </ownedEnd>
    </ownedMember>
```

```
    <ownedMember xmi:type="uml:Association" xmi:id="de15b3a1-0198-4bee-b7fc-12962a38d4a7"
name="wine_hassugar_winesugar">
      <memberEnd xmi:idref="8d5032bf-cdda-43a7-98ac-af3d262bd2e9" />
      <ownedEnd xmi:id="b93ee5fa-fca7-4de2-abcd-2cff9591c48c" name="wine" lower="4294967294"
upper="4294967294">
        <type xmi:idref="0871b6ee-2d44-4393-957e-4a6b899f9d95" />
        <association xmi:idref="de15b3a1-0198-4bee-b7fc-12962a38d4a7" />
      </ownedEnd>
    </ownedMember>
    <ownedMember xmi:type="uml:Class" xmi:id="b5063215-ae67-4dff-9ff4-4bdc48cc58d3" name="sweet" />
    <ownedMember xmi:type="uml:Association" xmi:id="b9a5fdce-0dd9-45cc-9683-d2f0b5583877"
name="sweetwine_hassugar_sweet">
      <memberEnd xmi:idref="f3253d52-1b21-4e6e-ad3b-250560e5d7c0" />
      <ownedEnd xmi:id="65c9bc9a-328d-4d73-8e6c-588c15ef4d27" name="sweetwine" lower="4294967294"
upper="4294967294">
        <type xmi:idref="7d7c5ade-e7aa-444c-ade3-c29c6f3e4c05" />
        <association xmi:idref="b9a5fdce-0dd9-45cc-9683-d2f0b5583877" />
      </ownedEnd>
    </ownedMember>
  </uml:Model>
  <uml:Association xmi:id="09416946-ac67-4982-9e8a-9b33c8280f68" xmi:href="#57a3f860-7def-4972-
906a-d13739ed7ca0" />
  <uml:Class xmi:id="13e3a91e-cc45-4aad-8209-8e742ccda880" xmi:href="#9bdf278d-f701-470d-9529-
7f56e017d973" />
  <uml:Class xmi:id="b8fda854-ffde-43bd-86ad-2b74639148e6" xmi:href="#b5063215-ae67-4dff-9ff4-
4bdc48cc58d3" />
  <uml:Association xmi:id="3c6b4fbd-89c1-472d-9f55-18398410b901" xmi:href="#b9a5fdce-0dd9-45cc-
9683-d2f0b5583877" />
  <uml:Class xmi:id="c0f91075-aca2-4b2a-aae0-eeb41cf74c0a" xmi:href="#e74f7a8c-b3f6-479a-a39e-
526886870766" />
  <uml:Class xmi:id="f0805d5a-29cc-449e-b8eb-d5938aeb792d" xmi:href="#bcf703bd-b72c-4439-82e2-
9a9b878441ce" />
  <uml:Association xmi:id="50d5f472-3b04-4ec9-8e91-f4d6d972b27c" xmi:href="#de15b3a1-0198-4bee-
b7fc-12962a38d4a7" />
```

Diagrama UML Obtenido



Comentarios

En este ejemlo se pueden observar la transformación de varios elementos OWL.

1) Se ha tenido en cuenta la relación entre propiedades, es decir, puede observarse que la propiedad “hasSugar” no incluye un dominio en su relación, por lo que al estar identificada como subpropiedad de otra propiedad, se realiza una búsqueda del dominio necesario para establecer la relación.

2) La propiedad “hasSugar” está definida como tipo funcional (FunctionalProperty), esto es tratado como una cardinalidad exacta de 1 por el sistema.

3) Para la definición de la clase SweetWine, se utiliza una intersección. Este elemento es tratado por el sistema desarrollado de tal manera que se puedan relacionar e integrar las características descritas en la clase. En este caso la intersección para definir la clase es realizada como una subclase de otra clase, y una restricción de valor sobre la clase actual.

4) Por otra parte, en la definición de las clases en OWL podemos encontrar restricciones de valor en su deficinición mediante “hasValue”, que en este caso es tratado como una relación a una instancia.

7.4 Elementos UnionOf

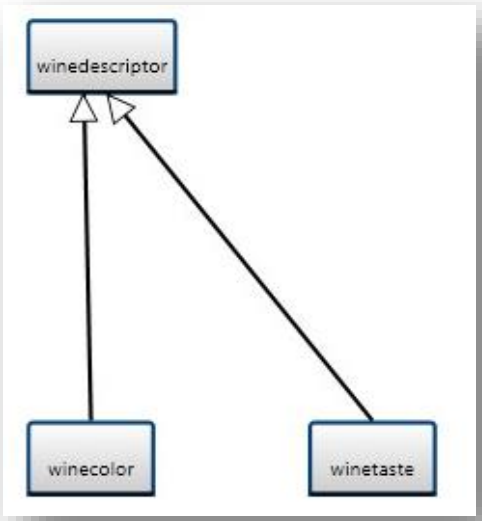
Código OWL

```
 <owl:Class rdf:ID="WineDescriptor">
    <rdfs:comment>Made WineDescriptor unionType of tastes and color</rdfs:comment>
    <owl:unionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#WineTaste" />
      <owl:Class rdf:about="#WineColor" />
    </owl:unionOf>
  </owl:Class>
```

Código XMI obtenido

```
<uml:Model xmi:id="9baa8386-6672-4eb7-a028-55ccc42a8b64" name="mimodelo">
    <ownedMember          xmi:type="uml:Class"          xmi:id="4ea81445-9eaa-4d3f-bf61-940b7fd3cbe5"
name="winedescriptor" />
    <ownedMember          xmi:type="uml:Class"          xmi:id="9022776f-67c2-4ad0-860a-faece77c7c99"
name="winetaste">
      <generalization xmi:id="f8e38d78-ea56-47d5-bec7-47a480b60405" isSubstitutable="false">
        <general xmi:idref="4ea81445-9eaa-4d3f-bf61-940b7fd3cbe5" />
        <ownedComment xmi:id="50310809-6314-4116-a769-7e39297f6768">
          <annotatedElement xmi:idref="f8e38d78-ea56-47d5-bec7-47a480b60405" />
          <body>{complete}</body>
        </ownedComment>
      </generalization>
    </ownedMember>
    <ownedMember          xmi:type="uml:Class"          xmi:id="a1dd8f3e-729f-4bb3-a094-a74f8d2daa40"
name="winecolor">
      <generalization xmi:id="6ee515fb-17b0-4c4e-ace6-b376e322ee2a" isSubstitutable="false">
        <general xmi:idref="4ea81445-9eaa-4d3f-bf61-940b7fd3cbe5" />
        <ownedComment xmi:id="10063db7-9b7d-4837-8f2d-9b7e9743ec78">
          <annotatedElement xmi:idref="6ee515fb-17b0-4c4e-ace6-b376e322ee2a" />
          <body>{complete}</body>
        </ownedComment>
      </generalization>
    </ownedMember>
  </uml:Model>
```

Diagrama UML Obtenido



Comentarios

La transformación del elemento UnionOf da como resultado el código XMI anterior. Esta generalización debe entenderse como "completa" usando el comentario correspondiente en el código XMI. Aunque en la imagen anterior no es mostrado, puede observarse en el código XMI la inclusión del comentario "{Complete}" a fin de representar que la superclase está estrictamente definida con las dos subclases incluidas.

7.5 Transformación del elemento InverseOf

Código OWL

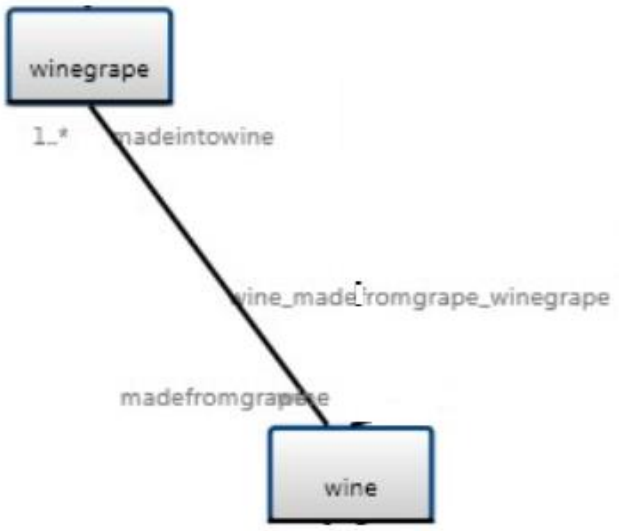
```
<owl:ObjectProperty rdf:ID="madeIntoWine">
   <owl:inverseOf rdf:resource="#madeFromGrape" />
 </owl:ObjectProperty>

  <owl:ObjectProperty rdf:ID="madeFromGrape">
   <rdfs:subPropertyOf rdf:resource="&food;madeFromFruit" />
   <rdfs:domain rdf:resource="#Wine" />
   <rdfs:range rdf:resource="#WineGrape" />
 </owl:ObjectProperty>

 <owl:Class rdf:ID="WineGrape">
   <rdfs:subClassOf rdf:resource="&food;Grape" />
 </owl:Class>
```

Código XMI obtenido

```
<uml:Model xmi:id="25918fbf-ed0e-4d17-a18d-4f08753e5f28" name="mimodelo">
    <ownedMember xmi:type="uml:Class" xmi:id="eebf01b8-741f-43de-8490-3e711b0936a4" name="wine">
      <ownedAttribute       xmi:id="b881a237-4288-49f2-b517-429d92932195"       name="madeintowine"
lower="4294967294" upper="4294967294">
        <type xmi:idref="b2b2d2f6-7fa2-417f-a176-035eb4a75ac3" />
```

```
        <association xmi:idref="abce6eab-112a-4552-9a36-9b0e74373303" />
      </ownedAttribute>
    </ownedMember>
    <ownedMember          xmi:type="uml:Class"          xmi:id="71075ecf-c355-4c30-a716-85d176736654"
name="winegrape">
      <generalization xmi:id="b0f25ef4-0352-4463-a095-bc5330ca93cb" isSubstitutable="false">
        <general xmi:idref="7d1090ad-e911-49be-a69a-d98e940b505e" />
      </generalization>
      <ownedAttribute       xmi:id="b861afac-3815-4a81-8c49-ce186ea544e6"       name="madefromgrape"
lower="4294967294" upper="4294967294">
        <type xmi:idref="eebf01b8-741f-43de-8490-3e711b0936a4" />
        <association xmi:idref="abce6eab-112a-4552-9a36-9b0e74373303" />
      </ownedAttribute>
    </ownedMember>
    <ownedMember xmi:type="uml:Class" xmi:id="874df30a-c630-43d4-a728-55051d76a20f" name="grape" />
    <ownedMember xmi:type="uml:Class" xmi:id="a3fe87c7-4519-4c45-90ba-74b0ac9cc510" name="" />
    <ownedMember       xmi:type="uml:Association"       xmi:id="900cf668-abfb-44c5-b45f-05877309db0a"
name="wine_madefromgrape_winegrape">
      <memberEnd xmi:idref="b881a237-4288-49f2-b517-429d92932195" />
      <memberEnd xmi:idref="b861afac-3815-4a81-8c49-ce186ea544e6" />
    </ownedMember>
  </uml:Model>
  <uml:Association    xmi:id="abce6eab-112a-4552-9a36-9b0e74373303"    xmi:href="#900cf668-abfb-44c5-
b45f-05877309db0a" />
  <uml:Class    xmi:id="7d1090ad-e911-49be-a69a-d98e940b505e"    xmi:href="#874df30a-c630-43d4-a728-
55051d76a20f" />
  <uml:Class    xmi:id="b2b2d2f6-7fa2-417f-a176-035eb4a75ac3"    xmi:href="#71075ecf-c355-4c30-a716-
85d176736654" />
```

Diagrama UML Obtenido



(*) No se incluye el grafo completo por razones de espacio

Comentarios

Como puede observarse, la relación inversa entre dos clases se ha representado como una asociación bidireccional en la que el nombre de los roles de cada clase es incluido en el final de la navegación de cada una de ellas.

7.6 Enumeración de instancias/individuals

Código OWL

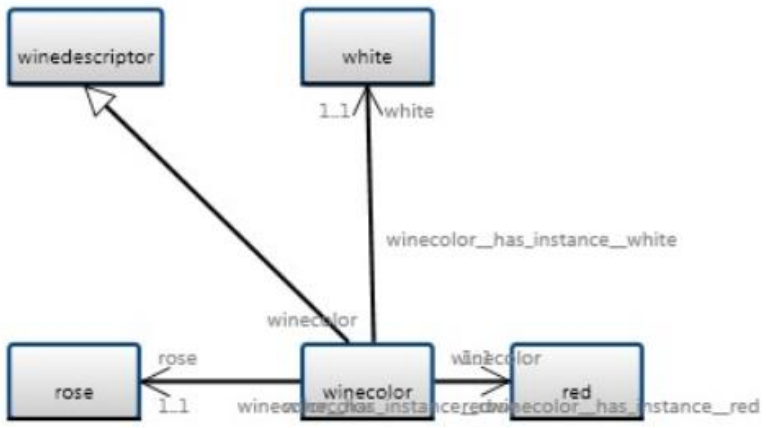
```
 <owl:Class rdf:ID="WineColor">
    <rdfs:subClassOf rdf:resource="#WineDescriptor" />
    <owl:oneOf rdf:parseType="Collection">
      <owl:Thing rdf:about="#White" />
      <owl:Thing rdf:about="#Rose" />
      <owl:Thing rdf:about="#Red" />
    </owl:oneOf>
  </owl:Class>
```

Código XMI obtenido

```
<uml:Model xmi:id="59e716ff-793d-4761-91de-9c97ec5430cb" name="mimodelo">
    <ownedMember         xmi:type="uml:Class"         xmi:id="2153e06a-b4ab-4f7f-a00f-3658f15bcef9"
name="winecolor">
      <generalization xmi:id="ccba32c8-1e7c-4d06-9d72-e71a74f25e8b" isSubstitutable="false">
        <general xmi:idref="c4390b26-6b5f-48e4-920d-787b1bd86144" />
      </generalization>
      <ownedAttribute xmi:id="02038819-55c5-4201-bb71-177f1c97a4c6" name="white">
        <type xmi:idref="8cfda477-06fd-43ac-a119-613848048009" />
        <association xmi:idref="25aefd7a-b657-4b7c-8530-a5f8507dd045" />
      </ownedAttribute>
      <ownedAttribute xmi:id="690aa571-c88c-4c82-98f8-044d084de68e" name="rose">
        <type xmi:idref="68bc9347-284a-4588-8743-e0c93b22de74" />
        <association xmi:idref="2d5bd751-46e5-4be4-9687-0b9ae9df0098" />
      </ownedAttribute>
      <ownedAttribute xmi:id="1f6de9ea-f411-4dd1-896c-24391be52afe" name="red">
        <type xmi:idref="70108f8b-95e6-4c67-8cd6-183273540be2" />
        <association xmi:idref="a7bf4c6e-43e5-45fb-b82d-07fdb6ad4986" />
      </ownedAttribute>
    </ownedMember>
    <ownedMember         xmi:type="uml:Class"         xmi:id="60480d00-2a08-4bc9-b6e2-36135648dcf1"
name="winedescriptor" />
    <ownedMember xmi:type="uml:Class" xmi:id="b8c086a5-f465-457b-9fd1-aabe15583ea8" name="white" />
    <ownedMember xmi:type="uml:Class" xmi:id="9a9df642-2e56-4c36-aeda-78d207b6d7b4" name="rose" />
    <ownedMember xmi:type="uml:Class" xmi:id="7d3d66ad-d661-4738-9146-7dbd6d4b0a4e" name="red" />
    <ownedMember       xmi:type="uml:Association"       xmi:id="5f5b1085-54c3-419d-98b4-e79272375207"
name="winecolor__has_instance__white">
      <memberEnd xmi:idref="02038819-55c5-4201-bb71-177f1c97a4c6" />
      <ownedEnd  xmi:id="4cfc57e6-932c-469a-9d02-a6f5622c82fc"  name="winecolor"  lower="4294967294"
upper="4294967294">
        <type xmi:idref="2153e06a-b4ab-4f7f-a00f-3658f15bcef9" />
        <association xmi:idref="5f5b1085-54c3-419d-98b4-e79272375207" />
      </ownedEnd>
    </ownedMember>
    <ownedMember       xmi:type="uml:Association"       xmi:id="185a5853-6981-4fab-bea0-d9b6ea17a941"
name="winecolor__has_instance__rose">
      <memberEnd xmi:idref="690aa571-c88c-4c82-98f8-044d084de68e" />
      <ownedEnd  xmi:id="fa112540-d2d5-4acf-b99e-66f27bcc5e91"  name="winecolor"  lower="4294967294"
upper="4294967294">
        <type xmi:idref="2153e06a-b4ab-4f7f-a00f-3658f15bcef9" />
        <association xmi:idref="185a5853-6981-4fab-bea0-d9b6ea17a941" />
      </ownedEnd>
    </ownedMember>
    <ownedMember       xmi:type="uml:Association"       xmi:id="be051c3f-ef26-485a-bf45-9ff701b61632"
name="winecolor__has_instance__red">
      <memberEnd xmi:idref="1f6de9ea-f411-4dd1-896c-24391be52afe" />
      <ownedEnd  xmi:id="f3e50957-536e-4978-9a8a-eec54e01631d"  name="winecolor"  lower="4294967294"
upper="4294967294">
        <type xmi:idref="2153e06a-b4ab-4f7f-a00f-3658f15bcef9" />
        <association xmi:idref="be051c3f-ef26-485a-bf45-9ff701b61632" />
      </ownedEnd>
    </ownedMember>
  </uml:Model>
  <uml:Class    xmi:id="c4390b26-6b5f-48e4-920d-787b1bd86144"    xmi:href="#60480d00-2a08-4bc9-b6e2-
36135648dcf1" />
  <uml:Class    xmi:id="70108f8b-95e6-4c67-8cd6-183273540be2"    xmi:href="#7d3d66ad-d661-4738-9146-
7dbd6d4b0a4e" />
  <uml:Association    xmi:id="25aefd7a-b657-4b7c-8530-a5f8507dd045"    xmi:href="#5f5b1085-54c3-419d-
```

```
98b4-e79272375207" />
  <uml:Class    xmi:id="8cfda477-06fd-43ac-a119-613848048009"    xmi:href="#b8c086a5-f465-457b-9fd1-
aabe15583ea8" />
  <uml:Class    xmi:id="68bc9347-284a-4588-8743-e0c93b22de74"    xmi:href="#9a9df642-2e56-4c36-aeda-
78d207b6d7b4" />
  <uml:Association    xmi:id="2d5bd751-46e5-4be4-9687-0b9ae9df0098"    xmi:href="#185a5853-6981-4fab-
bea0-d9b6ea17a941" />
  <uml:Association    xmi:id="a7bf4c6e-43e5-45fb-b82d-07fdb6ad4986"    xmi:href="#be051c3f-ef26-485a-
bf45-9ff701b61632" />
```

Diagrama UML Obtenido



Comentarios

Las clases pueden ser también definidas en las ontologías OWL, las instancias que pueden tener. En este caso no encontramos en un principio con dos opciones: Realizar una enumeración sobre una clase, o considerarlas como relaciones a instancias.
Finalmente se seleccionó ésta última opción ya que de esta manera no perderíamos información si el código XMI es reutilizado en otros lenguajes/modelos como RSHP.
Por tanto la representación de la clase Winecolor tiene las relaciones a las instancias que podrían aparecer en un modelo recuperado desde una ontología OWL.

7.7 Intersección de multiples clases, herencia múltiple

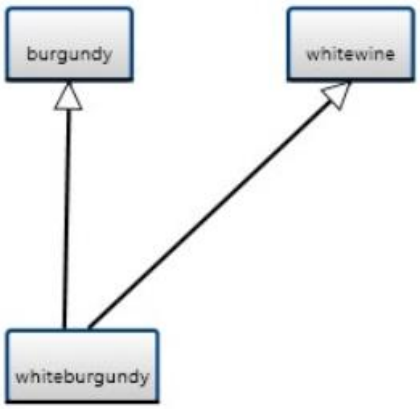
Código OWL

```
  <owl:Class rdf:ID="WhiteBurgundy">
    <owl:intersectionOf rdf:parseType="Collection">
      <owl:Class rdf:about="#Burgundy" />
      <owl:Class rdf:about="#WhiteWine" />
    </owl:intersectionOf>
  </owl:Class>
```

Código XMI obtenido

```
<uml:Model xmi:id="334bb91c-6f69-4ac3-a4a5-7be4ca2e7d1d" name="mimodelo">
    <ownedMember          xmi:type="uml:Class"          xmi:id="f40b88d2-cb85-458f-801f-08b2de1efd0e"
name="whiteburgundy">
      <generalization xmi:id="9a69dd4d-0a2c-4f7e-900f-3fb2cc67ecc1" isSubstitutable="false">
        <general xmi:idref="7338ad11-1970-4e32-aeb3-fd433a7a0b62" />
      </generalization>
      <generalization xmi:id="bce52fdf-1539-41df-96ee-313a7e4edb3f" isSubstitutable="false">
        <general xmi:idref="567da35a-afe3-43e1-8c4a-272236cd68a5" />
      </generalization>
    </ownedMember>
    <ownedMember xmi:type="uml:Class" xmi:id="574cdcc8-1c3e-4dc9-b5f1-9fe25abf0cfc" name="burgundy"
/>
    <ownedMember          xmi:type="uml:Class"          xmi:id="1a213544-0411-4a64-b7fb-22715461e407"
name="whitewine" />
  </uml:Model>
  <uml:Class    xmi:id="7338ad11-1970-4e32-aeb3-fd433a7a0b62"    xmi:href="#574cdcc8-1c3e-4dc9-b5f1-
9fe25abf0cfc" />
  <uml:Class    xmi:id="567da35a-afe3-43e1-8c4a-272236cd68a5"    xmi:href="#1a213544-0411-4a64-b7fb-
22715461e407" />
```

Diagrama UML Obtenido



Comentarios

Teniendo en cuenta que en UML es posible la herencia múltiple, la transformación ha sido en este caso utilizando este recurso. En OWL pueden existir intersecciones que utilizan distintas propiedades y recuersos en el lenguaje para aportar la máxima expresividad, y en este caso se define una clase como la intersección de otras dos resultan el ejemplo mostrado.

7.8 SameAs, DifferentFrom: Relaciones entre instancias

Código OWL

```
<owl:Class rdf:ID="FootballTeam">
   <owl:sameAs rdf:resource="http://sports.org/US#SoccerTeam"/>
</owl:Class>
```

Código XMI obtenido

Diagrama UML Obtenido

Comentarios

En OWL como se ha explicado anteriormente, se pueden expresear la igualdad o diferencia entre instancias, aportando información sobre si las instancias son iguales, o totalmente distintas en caso que se quiera especificar. Esta propiedad se ha transformado a UML estableciendo una relación en la que tenemos que:

- Owl:SameAs: Si dos instancias son iguales estableceremos una relación unidireccional dominio-rango en la que la relación tendrá el nombre Instancia1_SameAs_Instancia2
- Iwl:differentFrom: Si dos instancias son indicadas como distintas se tendrá una relación unidireccional en la que esta relación tendrá el nombre Instancia1_DifferentFrom_Instancia2

(*) En el ejemplo anterior se muestra sólo el ejemplo pasa el caso de “owl:sameAs” por considerarse muy similar al de owl:differentFrom.

Modelo UMLModels

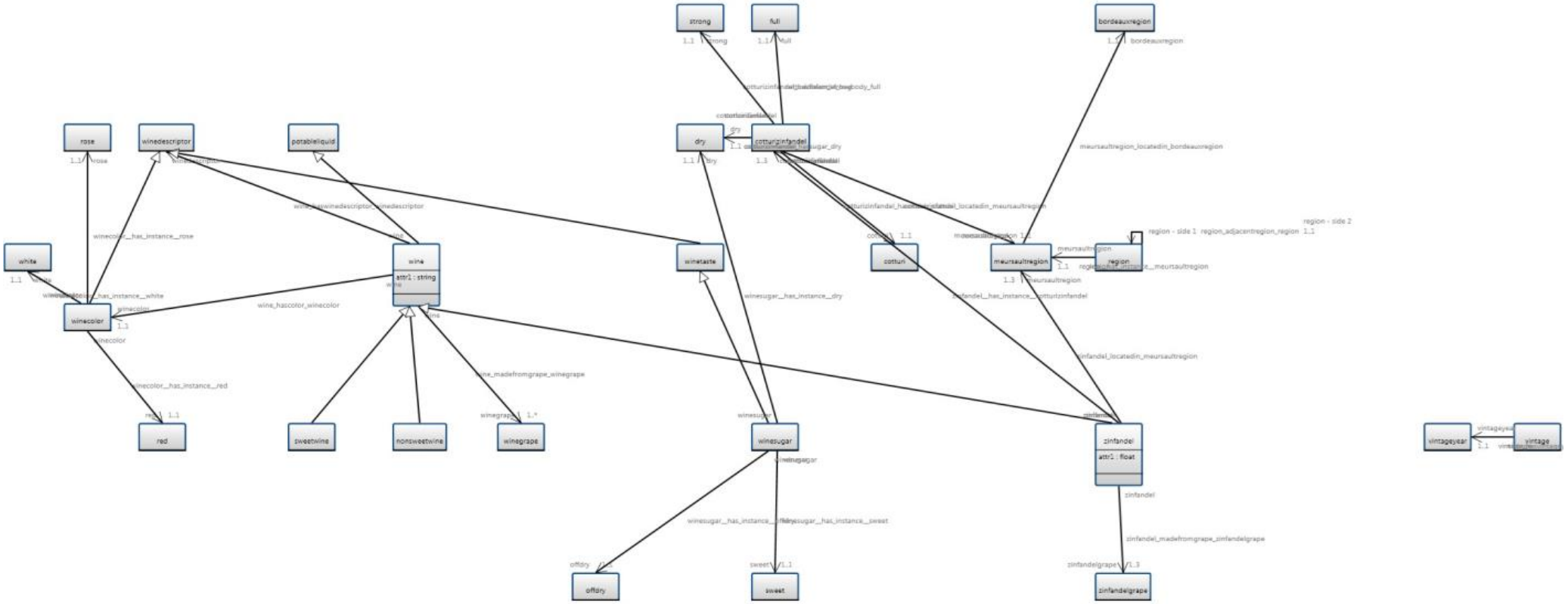


Ilustración 51. Ejemplo transformaciónOWL2UML

8 Conclusiones y trabajos futuros

En este capítulo se incluirán aspectos como la evaluación del trabajo realizado y los resultados obtenidos, análisis de la planificación inicial con respecto a los datos reales en la realización del proyecto y posibles trabajos futuros.

8.1 Conclusiones

8.1.1 Resultados de la planificación

El objetivo en este caso, es analizar y comparar la planificación inicialmente establecida para este proyecto, comparándola con los resultados finales en este aspecto teniendo en cuenta las distintas etapas del proyecto.

La planificación inicial se desarrolló para un curso académico completo, teniendo en cuenta que sólo se dispondrían de algunas horas a la semana para la realización del proyecto, ya que el alumno se encontraba trabajando a tiempo completo durante la realización del proyecto.

Inicialmente se planificó el final del proyecto para el 16 de mayo de 2015, siendo finalizado el día 30 de mayo. Este desfase de dos semanas, es debido a que la etapa de análisis ha llevado más tiempo del inicialmente previsto, y a que el alumno ha estado compaginando la realización del proyecto con su actividad laboral, por lo que en algunas etapas se han producido retrasos. Además, la planificación se realizó para un curso académico completo, por lo que hay que esperar que cuanto mayor sea el plazo para la planificación, nos dice la experiencia que mayor puede ser la probabilidad de fallo en la planificación.

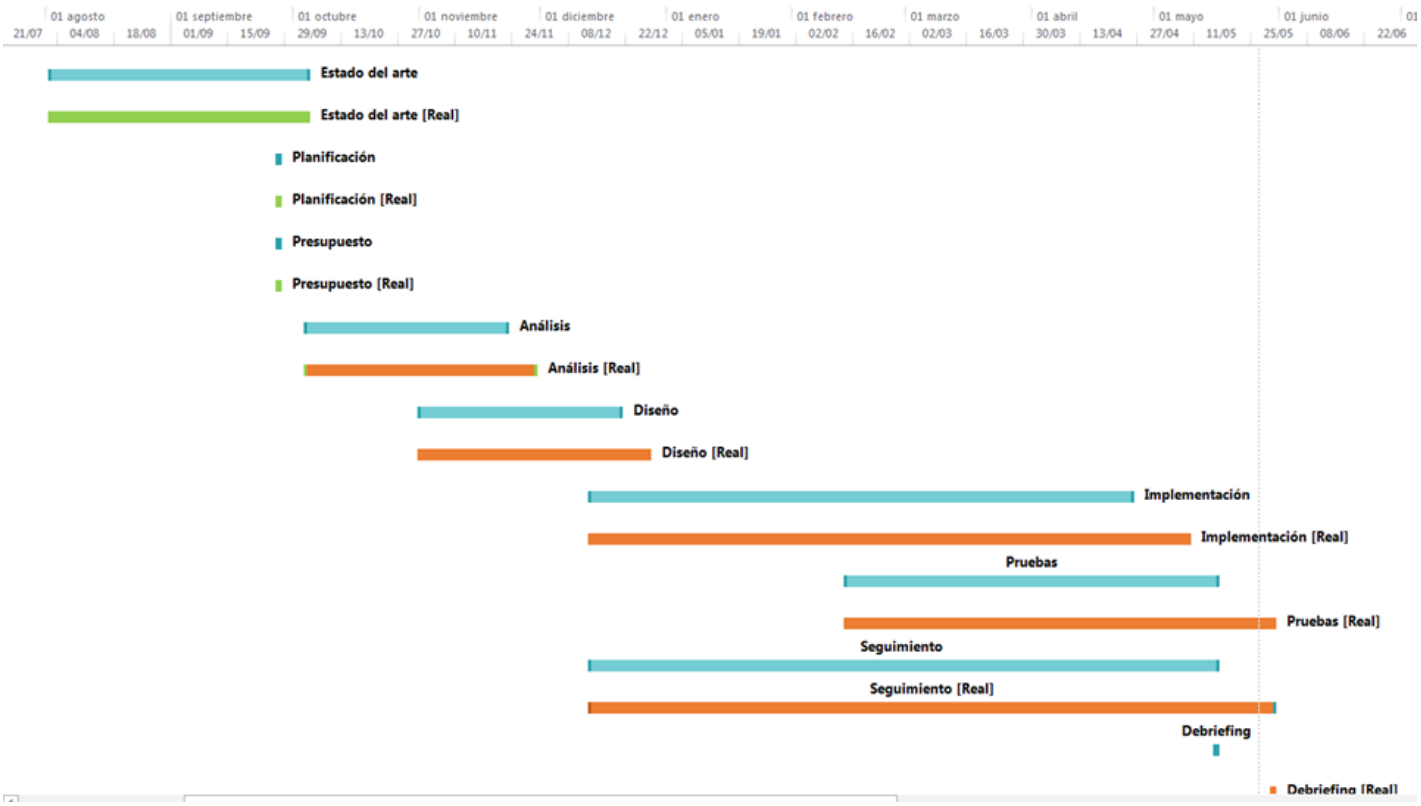


Ilustración 52. Debrief tiempos real

8.1.2 Evaluación del trabajo realizado

En este apartado se realizará un análisis personal del trabajo desarrollado durante la realización del proyecto, teniendo en cuenta aspectos como la planificación, la investigación sobre los elementos y tecnologías fundamentales para conseguir los objetivos, el uso de nuevas herramientas y la valoración de los resultados obtenidos.

En primer lugar, centrándonos en la planificación del proyecto, es importante indicar lo realmente importante que es la planificación y metodología seguida en la realización de un proyecto. Durante la realización de los distintos cursos de la carrera, he podido observar como nuevas metodologías iban tomando posición en el ámbito de la gestión de proyectos software. Este es por ejemplo el caso de las metodologías ágiles, como puede ser Scrum. Aunque la formación recibida ha sido en gran parte la conocida como tradicional, durante la realización del proyecto se han sabido adaptar y utilizar algunos aspectos de las metodologías ágiles, sin llegar a seguir estas metodologías de forma estricta, como por ejemplo la realización de las pruebas simultáneamente al desarrollo de software. Hay que señalar también, que la planificación en este proyecto ha estado muy condicionada por tener que realizarse el proyecto, a la vez que me encontraba trabajando a tiempo completo. De todas formas, esta situación me ha ayudado, a

tener que ajustar y compaginar mejor varias tareas y objetivos en paralelo, ampliando mi experiencia en estos aspectos.

Otra de las situaciones a analizar, es la investigación realizada en el estado del arte del proyecto. Desde un principio, tomé contacto con las tecnologías asociadas al tema principal de este proyecto, que es la web semántica y la representación de ontologías. El tema de la web semántica, era totalmente desconocido para mí, ya que anteriormente nunca me había encontrado con una actividad relacionada. Debido a esto, y a que en un principio mi visión sobre el proyecto no estaba muy madurada, llegué a emplear bastante tiempo concretando y sintetizando los objetivos y requisitos fundamentales a tener en cuenta. Gracias al apoyo de mi tutora, y de otros compañeros en el departamento, conseguí establecer adecuadamente los elementos necesarios para el éxito del proyecto. Esto demuestra, el buen resultado que puede aportar la colaboración entre las personas involucradas en el proyecto, y la determinación de alcanzar los objetivos.

Durante la etapa de implementación, fue necesario obtener experiencia en la tecnología utilizada (.NET), ya que no la había utilizado hasta ese momento. Este proceso de aprendizaje, a la par que realizaba otras actividades profesionales, me ayudó a mejorar las estrategias de aprendizaje de las tecnologías utilizadas en distintos ámbitos profesionales.

Finalmente, viendo los resultados obtenidos, es decir, la materialización de lo que en un principio era sólo una visión, o una planificación, aparece la satisfacción de haber conseguido los objetivos inicialmente establecidos.

8.1.3 Trabajos futuros

Analizando ahora los posibles trabajos futuros, podemos ver que hay algunos aspectos interesantes en los que poder trabajar como por ejemplo, la ampliación de la aplicación para futuras revisiones de OWL, en las que se incluyan nuevos elementos que permitan aportar más significado a las ontologías web.

Además, teniendo en cuenta que en este proyecto, la aplicación ha sido diseñada como un módulo de otra aplicación más grande, que es UMLModels, se podrían incluir nuevas aplicaciones que realizasen la transformación desde otros lenguajes que puedan ser recuperados por crawlers.

9 Anexo

En este apartado se incluye un glosario sobre los términos y expresiones utilizados en el proyecto, así como las referencias y fuentes utilizadas para las labores de investigación.

9.1 Glosario y definiciones

Definiciones

Diagrama de Gantt: herramienta gráfica que se utiliza para representar el tiempo previsto para distintas actividades o tareas.

ECTS: European Credit Transfer and Accumulation System (ECTS) o Sistema Europeo de Transferencia y Acumulación de Créditos. La carga lectiva de clases y otras actividades como prácticas etc, son medidas con este sistema en las universidades europeas, a fin de tener una referencia común y poder realizar procesos de convalidación u otros procesos en que pueda ser necesario.

Indexación: se refiere a los distintos métodos o actividades realizadas mediante software para la recuperación y almacenamiento de datos de la Web para distintos fines.

Metodología ágil: metodologías en ingeniería del software que utilizan el desarrollo iterativo e incremental.

Nunit: es un framework open source utilizado para la realización de pruebas sobre el código en .NET.

Open source: se denomina así al software desarrollado libremente, sin estar sujeto a licencias o limitaciones legales. Se denomina así también al software que se puede modificar, reutilizar o redistribuir libremente.

Scrum: Metodología de gestión de proyectos, dentro de las llamadas metodologías ágiles.

Semántica: se refiere a los aspectos del significado sentido o interpretación de las expresiones, las palabras o representaciones formales.

Software: conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar distintas tareas en una computadora.

Stakeholder: afectado o colaborador en el proyecto.

UMLModels: aplicación utilizada en el departamento para la transformación de varios lenguajes a UML (XMI).

Acrónimos

BLL: Business Logic Layer

BT: Broader Term

Cake: Computer Aided Knowldege Environment

DAL: Data Access Layer

DAML: DARPA Agent Markup Language

DNS: Domain Name System

HTML: Hypertext Markup Language

HTTP: Hypertext Transfer Protocol

NT: Narrower Term

OCL: Object Constraint Language

ODM: Ontology Definition Metamodel

OMG: Object Management Group

OWL: Ontology Web Language

RDF: Resource Description Framework

RT: Related Term

RIF: Rule Interchange Format

RSHP: Relationship Language

SPARQL: SPARQL Protocol and RDF Language

UML: Unified Modeling Language

URL: Uniform Resource Locator

URN: Uniform Resource Name

URI: Uniform Resource Identifier

W3C: World Wide Web Consortium

XML: eXtensible Markup Language

WWW: World Wide Web

9.2 Referencias

[1]: [WEB1] Tim Berners-Lee,

- Enlace: http://www.w3.org/People/Berners-Lee/
- Última visita: febrero 2015

[2]: [WEB2] URIs, URLs, and URNs: Clarifications and Recommendations 1.0

- Enlace: http://www.w3.org/TR/uri-clarification/
- Última visita: febrero 2015

[3]: [WEB3] World Wide Web,

- Enlace: http://es.wikipedia.org/wiki/World_Wide_Web
- Última visita: febrero 2015

[4]: [WEB4] Diagrama de Venn, Wikipedia,

- Enlace: http://es.wikipedia.org/wiki/Diagrama_de_Venn
- Última visita: febrero 2015

[5]: [WEB5] URI, Wikitel,

- Enlace: http://wikitel.info/wiki/URI
- Última visita: febrero 2015

[6]: [WEB6] The World wide web: Past, Present and Future, Tim Berners-Lee.

- Enlace: http://www.w3.org/People/Berners-Lee/1996/ppf.html
- Última visita: febrero 2015

[7]: [WEB7] Dan Connolly Format Negotiation,

- Enlace: http://www.w3.org/Talks/9902arch-wap/slide8-0.html
- Última visita: febrero 2015

[8]: [WEB8] W3C, Content negotiation,

- Enlace: http://www.w3.org/Protocols/rfc2616/rfc2616-sec12.html
- Última visita: febrero 2015

[9]: [WEB9] Hipertexto, Wikipedia,

- Enlace: http://es.wikipedia.org/wiki/Hipertexto
- Última visita: febrero 2015

[10]: [WEB10] What is Web 2.0, O'Reilly

- Enlace: http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html?page=1
- Última visita: marzo 2015

[11]: [WEB11] La Web 2.0, Wikipedia

- Enlace: http://es.wikipedia.org/wiki/Web_2.0
- Última visita: marzo 2015

[12]: [WEB12] La Web 2.0, Ministerio de Educación Cultura y Deporte

- Enlace:http://www.ite.educacion.es/formacion/materiales/155/cd/modulo_1_Iniciacionblog/concepto_de_web_20.html
- Última visita: marzo 2015

[13]: [WEB13] Semantic Web, W3C

- Enlace: http://www.w3.org/standards/semanticweb/
- Última visita: marzo 2015

[14]: [WEB14] Resource Description Framework (RDF), W3C

- Enlace: http://www.w3.org/RDF/
- Última visita: marzo 2015

[16]: [WEB15] Linked Data, Tim Berners-Lee, W3C

- Enlace: http://www.w3.org/DesignIssues/LinkedData.html
- Última visita: marzo 2015

[17]: [WEB16] Inferencia Diccionario Real Academia de la Lengua Española

- Enlace: http://lema.rae.es/drae/?val=inferencia
- Última visita: marzo 2015

[18]: [WEB17] Inference, W3C:

- Enlace: http://www.w3.org/standards/semanticweb/inference
- Última visita: marzo 2015

[19]: [WEB18] Aplicaciones verticales en la Web Semántica, W3C:

- Enlace: http://www.w3.org/standards/semanticweb/applications
- Última visita: marzo 2015

[20]: [WEB19] The Semantic Web, Tim Berners-Lee, James Hendler and Ora Lassila, Scientific American:

- Enlace: http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html
- Última visita: marzo 2015

[21]: [WEB20] Ontología, Wikipedia

- Enlace: http://es.wikipedia.org/wiki/Ontolog%C3%ADa
- Última visita: marzo2015

[22]: [WEB21] Taxonomía, Thesaurus Advisory Group, Miquel Centelles, universidad Pompeu Fabra:

- Enlace: http://www.upf.edu/hipertextnet/numero-3/taxonomias.html
- Última visita: marzo 2015

[23]: [WEB22] Unified Modeling Language, OMG

- Enlace: http://www.uml.org/
- Última visita: marzo 2015

[24]: Paul Kogut, Stephen Cranefield, Lewis Hart, Mark Dutra, Kenneth Baclawski and Mieczyslaw Koka, Jeffrey Smith, UML for ontology Development

- Enlace: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.94.4409&rep=rep1&type=pdf

[25]: Brian R Gaines , An Interactive Visual Language for Term Subsumption Languages

- Enlace: http://ijcai.org/Past%20Proceedings/IJCAI-91-VOL2/PDF/032.pdf

[26]: Sara Brockmans, Rafael Bolz, Andreas Eberhart, and Peter Löffler Visual Modeling of OWL DL Ontologies Using UML, Institute AIFB

- Enlace: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.7742&rep=rep1&type=pdf

[27]: Kilian Kiko and Colin Atkinson, A detailed comparison of UML and OWL, University of Mannheim

- Enlace: https://sonet.ecoinformatics.org/semtools-svn/reference/UMLToOWLAtkinson.pdf

[28]: Stephen Cranefield , UML and the Semantic Web, University of Otago

- Enlace: https://ourarchive.otago.ac.nz/bitstream/handle/10523/1005/dp2001-04.pdf?sequence=3

[29]: [WEB23] UML Profile for RDF and OWL, OMG Wiki.

- Enlace: www.omgwiki.org
- Última visita: marzo 2015

[30]: [WEB24] W3C, OWL Owl Ontology Language, Reference

- Enlace: http://www.w3.org/TR/owl-ref/
- Última visita: marzo 2015

[31]: [WEB25] W3C, OWL, Owl Ontology Language, Guide,

- Enlace: http://www.w3.org/TR/owl-guide/
- Última visita: marzo 2015

[32]: [WEB26] Ontology Definition Metamodel, OMG

- Enlace: http://www.omg.org/spec/ODM/1.1/PDF/
- Última visita: marzo 2015

[33]: Morgan Kaufman Publishers, Logical Foundations of Artificial Intelligence, 1987, Genesereth & Nilsson, 1987, Standford University

- Enlace: http://kryten.mm.rpi.edu/COURSES/LOGAIS02/nonmonotonic.pdf

[34]: A Translation Approach to Portable Ontology Specifications, Thomas R. Gruber, Standford University, 1993

[35]: Sonia Sánchez-Cuadrado, Jorge Morato, Ontología,

- Enlace: http://glossarium.bitrum.unileon.es/Home/ontologia

[36]: The Semantic Web, James Hendler, 2001

- Enlace: http://wi-consortium.org/wicweb/pdf/wi-hendler.pdf
- Última visita: marzo 2015

[37]: B. Chandrasekaran and John R. Josephson, Ohio State University V. Richard Benjamins, University of Amsterdam, What are ontologies and why do we need them?

[38]: Tom Gruber, What is an ontology?, 1992

- Enlace: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html
- Última visita: marzo 2015

[39]: Formal ontology, conceptual analysis and knowledge representation, Nicola Guarino, 1995

- Enlace: http://nemo.nic.uoregon.edu/wiki/images/7/79/Guarino_IJHCS1995_Formal_Onto_conceptual_analysis.pdf

[40]: Introducción al razonamiento sobre ontologías, Prof. Iván J. Flores Vitelli, Universidad Central de Venezuela

- Enlace: http://computacion.ciens.ucv.ve/escueladecomputacion/teachingDocuments/downloadFile/9

[41]: Gómez, Fernandez y Corcho, Ontological Engineering and the Semantic Web ,2004

- Enlace: http://www.exa.unicen.edu.ar/escuelapav/cursos/corcho/01_introduction.pdf

[42]: Universidad Nacional de Rosario, Aplicación de lógicas Descriptivas

- Enlace: http://www.dsi.fceia.unr.edu.ar/downloads/iia/presentaciones/Ontologias&LD.pdf

[43]: Modelo Entidad-Relación, Wikipedia

- Enlace: http://es.wikipedia.org/wiki/Modelo_entidad-relaci%C3%B3n

[44]: Mohammad Mustafa Taye, Web-Based Ontology Languages and its Based Description Logics, Department of Computer Information Systems Faculty of Information Technology Philadelphia University, Jordan

- Enlace: http://ijj.acm.org/volumes/volume2/issue2/ijjvol2no1.pdf

[45]: Vocabularies, W3C:

- Enlace: http://www.w3.org/standards/semanticweb/ontology
- Última visita: marzo 2015

[46]: [BOOK] Trafford Publishing , 2005, OWL: Representing Information Using The Web Ontology Language, Lee W. Lacy, TRAFFORD

[47]: RDF, Resource Description Framework

- Enlace: http://www.w3.org/RDF/
- Última visita: marzo 2015

[48]: Andreas Grünwald, Evaluation of UML to OWL Approaches and Implementation of a Transformation Tool for Visual Paradigm and MS Visio

- Enlace: ftp://ftp.heanet.ie/disk1/sourceforge/u/um/uml2owl/bachelor_thesis_uml2owl.pdf

[49]: [BOOK] Springer Publishing Company, 2010, Database Theory and Application: International Conference, DTA 2009, Xu et al

[50]: [WEB27] ATL Use Case - ODM Implementation (Bridging UML and OWL),

- Enlace: http://www.eclipse.org/atl/usecases/ODMImplementation/

[51]: [WEB28] UML Backend, Protégé, Standford University,

- Enlace: http://protegewiki.stanford.edu/wiki/UML_Backend
- Última visita: abril 2015

[52]: [WEB29] OWLGrEd, Institute of Mathematics and Computer Science, University of Latvia

- Enlace: http://owlgred.lumii.lv/
- Última visita: abril 2015

[53]: Dragan Gasevic, Dragan Djurić, Vladan Devedžić, Violeta Damjanović, Converting UML to OWL Ontologies, (OUP)

- Enlace: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.231.2181&rep=rep1&type=pdf

[54]: Radar Networks & Nova Spivack, 2007

- Enlace: http://novaspivack.typepad.com/RadarNetworksTowardsAWebOS.jpg

[55]: Ejemplo ontología OWL “Camera”.

- Enlace: http://jmvidal.cse.sc.edu/talks/xmlrdfdaml/owlexample.html

[56]: Diagramas de clases UML.

- Enlace: http://www.usmp.edu.pe/publicaciones/boletin/fia/info67/UML.pdf

[57]: Wiki Semantic Web, Ontostudio.

- Enlace: http://semanticweb.org/wiki/OntoStudio

[58]: VisioOWL.

- Enlace: https://sites.google.com/site/semanticsimulations2/home/visioowl
- Última visita: abril 2015

[59]: UML2OWL Application, Sebastian Leinhos

- Enlace: http://diplom.ooyoo.de/index.php?page=about
- Última visita: abril 2015

[60]: Twouse application, UML2OWL.

- Enlace: https://code.google.com/p/twouse/

[61]: CODIP, UML2OWL Transformation,

- Enlace: http://projects.semwebcentral.org/projects/codip/

[62]: UML2OWL Tool.

- Enlace: http://www.sfu.ca/~dgasevic/projects/UMLtoOWL/

[63]: Schema, DTS's y XML Schema

- Enlace: http://www.hipertexto.info/documentos/dtds.htm#schema

[64]: Carmen Fernández Chamizo, Jorge Gómez Sanz, Juan Pavón Mestras, Desarrollo de Agentes Software, Universidad Complutense de Madrid

- Enlace: http://www.fdi.ucm.es/profesor/jpavon/doctorado/sma.pdf

[65]: SPARQL, Wikipedia

- Enlace: https://es.wikipedia.org/wiki/SPARQL

[66]: Juan Llorens, Jorge Morato, Gonzalo Genova, RSHP: an information representation model based on relationships, Computer Science Department. Universidad Carlos III de Madrid. Spain

- Enlace:http://www.researchgate.net/publication/220046031_RHSP_an_Information_Representation_Model_Based_on_Relationship