



Universidad de Valladolid
Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención Ingeniería de Software

**DESARROLLO DE UN MODELO ONTOLÓGICO PARA LA
INTEGRACIÓN DE DATOS DE UNA UNIDAD DE
CLIMATIZACIÓN EN UN EDIFICIO INTELIGENTE**

Autor:

D. Ángel Nava Esteban

Tutores:

D. Carlos J. Alonso González

Dña. Yania Crespo González-Carvajal

Para todos y cada uno los que suman en mis recuerdos...

"Mientras dibujas la rama de un árbol, debes escuchar el aire"

Proverbio japonés

Agradecimientos

A mis amigos, compañeros de trabajo y de carrera, por su apoyo a lo largo del proceso.

A mi tutor Carlos, por abrirme una nueva área de conocimiento en el mundo de las ontologías y proponerme el trabajo.

A mi tutora Yania, por toda su constancia y dedicación, gracias por las enseñanzas y consejos ofrecidos.

Gracias a todos.

Resumen

El crecimiento de las tecnologías de *IoT* aplicadas en todo tipo de situaciones y procesos de la vida cotidiana ha incrementado la cantidad de datos que generamos y almacenamos. Englobar los sensores que conforman estructuras para su gestión eficiente, como puede ser un edificio inteligente (*Smart Building*), e integrar este desarrollo en las agendas de los gobiernos para crear las llamadas ciudades inteligentes (*Smart Cities*) ha implicado a multitud de organizaciones públicas y privadas. Esta escalada de datos ha generado múltiples paradigmas y formatos dependientes de cada impulsor del proyecto.

Por otro lado, el paradigma energético actual y la crisis climática, hacen que la eficiencia energética y la reducción de emisiones de CO₂ aumenten de escala en un esfuerzo por aumentar la eficiencia. Organismos oficiales como la Unión Europea han creado comités y estándares para abordar el problema. El análisis energético del consumo de un edificio, expone que el mayor porcentaje de uso se dedica a la climatización, superando incluso cuotas del 40% sobre el total. Una mala gestión de los múltiples sistemas implicados genera costes energéticos y mayores emisiones de CO₂, por lo que son de gran interés los datos que puede proporcionar un *Smart Building* a la hora de realizar estudios sobre gestión eficiente.

El objetivo de este Trabajo Fin de Grado es valorar la adaptación de un entorno de procesamiento de datos *IoT* basado en datos tabulados, a las tecnologías de procesamiento de información basadas en el uso de ontologías. El uso de ontologías garantiza la independencia entre los datos y su formato (representación sintáctica) mediante el uso de una semántica común compartida con una amplia comunidad. Esto permite una fácil compatibilidad e integración entre plataformas o programas externos que posibilita, entre otras cosas, interactuar con nuevos elementos ajenos al dominio inicial. Otra ventaja la encontramos en la posibilidad que ofrece esta tecnología para obtener nuevo conocimiento aplicando inferencia sobre los datos, lo que amplía la capacidad de actuación del sistema frente a los tratamientos tradicionales sobre datos tabulados.

La fuente de datos tomados como referencia para el proceso de adaptación pertenecen a la AHU101 (Unidad de Climatización número 101) del *Alice Perry Engineering Building* situado en Galway, Irlanda. Se fundamentará y analizará el estándar ISO SAREF (la ontología Smart Applications REference), su variante para Smart Building (SAREF4BLDN) y otros estándares relacionados como OWL-Time. Se modelizarán y describirán mediante el estándar OWL (variante OWL2-DL) y se comprobará su validez para obtener consultas y estadísticas descriptivas básicas mediante entornos gráficos.

Abstract

The boosting of IoT technologies applied to all kinds of issues and processes in regular life has increased the data amount generated and stored. Involving sensors that conform structures, like a building, for example, working out its efficient management (Smart Building), and leading it into the government agendas to develop Smart Cities has implicated multitude organizations, public and private. This increasing amount of data has generated different paradigms and formats depending on the developer's approach.

On the other hand, the current energy paradigm and the climate crisis make energy efficiency and CO₂ emission reductions scale up in an effort to increase efficiency. Official organizations, like the European Union, have developed committees and standards to deal with the problem. The energy analysis of the consumption of a building show that the main percentage is dedicated to air conditioning, even exceeding 40% of the prices as a whole. Poor management of the multiple systems involved generates unnecessary energy costs and CO₂ emissions, thus the data extracted from Smart Building can provide us with a lot to carry out efficient management studies.

This Final Degree Project's focus is to assess the fitting of an IoT data processing environment based on tabular data in to information processing technologies based on ontologies. Ontologies guarantees data independence from formats (syntactic representation) through a common semantic lay which shared with other knowledge areas, this causes integration and extends compatibility between platforms or external applications to allow, for example, interacting with new elements out from the initial domain. Another advantage is, in the possibility offered by this technology, to obtain new knowledge by applying data inference, this fact expands the system's capacity in comparation to traditional treatments of tabulated data

The data source in reference for the process belongs to AHU101 (Air Conditioning Unit No. 101) located in Alice Perry Engineering Building located in Galway, Ireland. The ISO SAREF standard (the Smart Applications REFERENCE ontology), its variant for Smart Building (SAREF4BLDN) and other related standards such as OWL-Time will be argued and analyzed. Data source will be modeled and described using the OWL standard (OWL2-DL variant) and checked its viability in order to obtain statistical analyzes using graphic environments.

Índice general

Agradecimientos.....	iii
Resumen.....	v
Abstract	vii
Índice general.....	ix
Lista de figuras	xiv
Lista de tablas	xvii
1. Introducción	1
1.1. Contexto	1
1.2. Objetivos	2
1.2.1. Objetivos de desarrollo.....	2
1.2.2. Objetivos personales	2
1.3. Estructura de la memoria	3
2. Contexto General.....	4
2.1. Metodología BIM (Building Information Modeling)	4
2.1.1. Open BIM y el formato IFC (Industry Foundation Classes).....	5
2.1.2. Arquitectura IFC, estándar EXPRESS	7
2.1.3. Arquitectura IFC, estándar XML (XSD)	10
2.2. Evolución de la World Wide Web	10
2.3. La Web Semántica	11
2.4. Esquemas de Representación del Conocimiento	15
2.4.1. Taxonomías y Tesauros.....	15
2.4.2. Ontologías	16
2.4.3. Modelado ontológico.....	18
2.4.4. Lenguajes ontológicos	19
2.4.5. Razonamiento ontológico.....	20
2.4.6. Herramientas de gestión de ontologías	21
2.4.7. Metodologías para el desarrollo y diseño de ontologías	22
2.5. Tecnologías y lenguajes estándar asociados a la Web Semántica	26
2.5.1. URI / IRI	26
2.5.2. XML / XML Schema.....	27

2.5.3. RDF / RDFS.....	27
2.5.4. OWL / OWL2	30
2.5.5. SPARQL.....	35
2.5.6. SWRL	36
3. Contexto específico.....	38
3.1. Sistemas de Climatización	38
3.1.1. Climatizador AHU (AIR HANDLING UNIT)	39
3.1.2. Representación BIM para HVAC-AHU	42
3.2. IFC OWL.....	44
3.3. SAREF.....	45
3.4. SAREF4BLDG	50
4. Metodología.....	53
4.1. Introducción	53
4.2. Escenarios.....	53
4.3. Glosario de procesos y actividades	55
4.4. Ciclos de vida.....	57
4.4.1. Modelo en cascada.....	57
4.4.2. Modelo iterativo-incremental.....	58
4.5. Pautas metodológicas.....	59
4.5.1. Especificación de requisitos	59
4.5.2. Planificación.....	62
4.5.3. Reutilización de recursos no ontológicos (NOR).....	62
4.5.4. Búsqueda de ontologías	63
4.5.5. Reutilización de ontologías de dominio.....	65
4.5.6. Reutilización de declaraciones ontológicas	66
4.5.7. Evaluación de ontologías	68
4.5.8. Alineación entre ontologías	70
4.5.9. Reingeniería de recursos no ontológicos (NOR)	71
4.5.10. Modularización.....	73
4.5.11. Conceptualización: utilización de patrones de diseño de ontologías	75
4.5.12. Evolución.....	76
4.5.13. Localización (adaptación lingüística y cultural).....	77
4.6. Otros recursos	78
4.6.1. NeOn Toolkit	78
4.6.2. Biblioteca para ODP	78
4.6.3. Biblioteca para reingeniería de PR-NOR	78

5. Plan de proyecto	80
5.1. Especificación de requisitos	80
5.1.1. Tarea 1. Identificar el propósito, alcance y lenguaje de implementación	80
5.1.2. Tarea 2. Identificar los usuarios finales previstos	80
5.1.3. Tarea 3. Identificar los usos previstos	80
5.1.4. Tarea 4. Identificar los requisitos	81
5.1.5. Tarea 5. Agrupar requisitos funcionales.....	81
5.1.6. Tarea 6. Validación de los requisitos	82
5.1.7. Tarea 7. Priorización de los requisitos.....	85
5.1.8. Tarea 8. Extracción de terminología y sus frecuencias	86
5.1.9. ORSD. Documento de Especificación de Requisitos	86
5.2. Planificación	90
5.2.1. Tarea 1. Selección del modelo del ciclo de vida	90
5.2.2. Tarea 2. Selección del conjunto de escenarios.....	91
5.2.3. Tarea 3. Actualización del plan inicial	91
5.2.4. Tarea 4. Establecimiento de restricciones y asignación de recursos	92
6. Plan de riesgos y presupuestos	93
6.1. Gestión de riesgos	93
6.1.1. Plan de riesgos	94
6.2. Presupuesto	97
6.2.1. Presupuesto simulado	97
6.2.2. Presupuesto real.....	98
7. Tecnologías y software empleados	99
7.1. Protégé	99
7.2. LODE.....	99
7.3. WebVOWL	99
7.4. Oops!	100
7.5. Python.....	100
7.6. Anaconda	100
7.7. Jupyter Notebook	100
7.8. Apache Spark.....	101
7.9. JSON API	101
7.10. Owready2.....	101
7.11. Ipwidgets	101
7.12. Numpy.....	101
7.13. Matplotlib	102

8. Desarrollo de la ontología	103
8.1. Reutilización de ontologías de dominio	103
8.1.1. Tarea 1. Búsqueda de ontologías de dominio	103
8.1.2. Tarea 2. Evaluación de Ontologías de Dominio	104
8.1.3. Tarea 3. Selección de Ontología de Dominio	105
8.1.4. Tarea 4. Integración de ontologías de dominio	106
8.2. Reutilización de recursos no ontológicos (NOR).....	107
8.2.1. Tarea 1. Búsqueda de recursos no ontológicos	107
8.2.2. Tarea 2. Valoración de recursos no ontológicos	108
8.2.3. Tarea 3. Selección de los recursos no ontológicos más apropiados	109
8.3. Reingeniería de recursos no ontológicos.....	109
8.3.1. Actividad 1. Ingeniería inversa de recursos no ontológicos	109
8.3.2. Actividad 2. Transformación de los recursos no ontológicos.....	111
8.3.3. Actividad 3. Ingeniería hacia adelante de la ontología	115
8.4. Formalización de la ontología	115
8.4.1. Diseño de las URIs	115
8.4.2. Vocabulario	115
8.4.3. Diagrama general de la ontología	118
8.5. Implementación	120
8.5.1. Diagrama de actividades	120
8.5.2. Implementación del tratamiento de los NOR	121
8.5.3. Implementación de la ontología.....	123
8.5.4. Implementación de los datos RDF y consultas SPARQL.....	126
8.6. Validación	127
9. Exploración y análisis de los datos RDF	132
9.1. Mapeado de las medidas en JSON	132
9.2. Transformación a RDF	133
9.3. Análisis de los datos: consultas SPARQL y gráficas.....	134
10. Seguimiento del proyecto.....	138
10.1. Introducción	138
10.2. Fase preliminar de documentación y estudio	138
10.3. Plan de proyecto: fase de iniciación (19/01/2021-15/02/2021)	139
10.4. Fase de reutilización (16/02/2021-23/02/2021)	139
10.5. Fase de reingeniería (24/02/2021- 23/03/2021)	140
10.6. Fase de diseño (24/01/2022-08/02/2022)	140
10.7. Fase de implementación (04/04/2022-04/05/2022)	141

10.8. Resumen.....	141
11. Conclusiones y líneas de trabajo futuras	142
11.1. Conclusiones	142
11.2. Líneas de trabajo futuras.....	143
Bibliografía	144
Apéndice A: contenido de la memoria	149
Apéndice B: reutilización de elementos.....	150
Apéndice C: ontología AHU101 Alice Perry.....	153
Apéndice D: instalación del entorno	162

Lista de figuras

Ilustración 1. Ciclo de vida de un proyecto con BIM	4
Ilustración 2. Estado de la adopción de BIM en el mundo	5
Ilustración 3. Estructura de capas de la arquitectura IFC	7
Ilustración 4. “ifcAlarm”: esquema de definiciones heredadas para supertipos	8
Ilustración 5. “ifcAlarm”: extracto del esquema de definición de recursos	8
Ilustración 6. Ejemplo simple en EXPRESS.....	9
Ilustración 7. Ejemplo de SCHEMA con restricciones aritméticas, lógicas y funciones	9
Ilustración 8. Representaciones Formales para <i>IfcAlarm</i>	10
Ilustración 9. Esquema para la Web 1.0, 2.0 y 3.0	11
Ilustración 10. Modelo de capas de la Web Semántica	12
Ilustración 11. Evolución de documentos RDF en la <i>Linked Open Data Cloud</i>	14
Ilustración 12. Imagen General del <i>Linked Open Data Cloud, 2022</i>	14
Ilustración 13. Taxonomía linneana, árbol filogenético animal	15
Ilustración 14. Razonamiento con LPO	20
Ilustración 15. Arquitectura de un SBDL.....	21
Ilustración 16. Razonamiento con Reglas.....	21
Ilustración 17. Fases en la metodología On-To-Knowledge	24
Ilustración 18. Fases en la metodología <i>Methontology</i>	25
Ilustración 19. Esquema de una URI.....	26
Ilustración 20. Ejemplo documento XML	27
Ilustración 21. Grafo de la tripleta RDF	28
Ilustración 22. Ejemplo grafo RDF	28
Ilustración 23. Serialización XML de un recurso en RDFS.....	29
Ilustración 24. Grado de representabilidad de los perfiles de OWL2	31
Ilustración 25. Estructura básica de un archivo OWL / OWL2	31
Ilustración 26. Representación de elementos básicos en OWL/OWL2.....	33
Ilustración 27. Consulta simple en SPARQL	36
Ilustración 28. Ejemplo de regla en SWRL.....	37
Ilustración 29. Consumos de energía en una edificación	39
Ilustración 30. Unidad exterior AHU	40
Ilustración 31. Esquema básico de unidad AHU	40
Ilustración 32. Jerarquía de definición sistema HVAC	42
Ilustración 33. Tipos de datos necesarios para el <i>ifcHvacDomain</i>	43

Ilustración 34. Ejemplo de especificación ifcOWL de la clase ifcBSplineCurve	45
Ilustración 35. Estructura de SAREF	46
Ilustración 36. Vista general de la ontología SAREF	46
Ilustración 37. SAREF: tipos de dispositivos	47
Ilustración 38. SAREF: clase Function y subclases	47
Ilustración 39. SAREF: clase Profile	48
Ilustración 40. SAREF modelo para Measurement y restricciones	48
Ilustración 41. SAREF: clase y propiedades de Measurement	49
Ilustración 42. Vista general de la extensión SAREF4BLDG	50
Ilustración 43. SAREF4BLDG: jerarquía para Device (Part.1)	51
Ilustración 44. SAREF4BLDG: jerarquía para Device (Part.2)	52
Ilustración 45. Escenarios para la metodología NeOn	54
Ilustración 46. NeOn: procesos y actividades incluidas en el glosario	55
Ilustración 47. NeOn: extracto del Glosario de Procesos y Actividades	56
Ilustración 48. NeOn: tareas para la especificación de requisitos	60
Ilustración 49. NeOn: modelo para la Plantilla ORSD	61
Ilustración 50. NeOn: tareas para el proceso de planificación	62
Ilustración 51. NeOn: tareas para la reutilización de recursos no ontológicos	63
Ilustración 52. NeOn: tareas para la búsqueda de ontologías	64
Ilustración 53. NeOn: reutilización de ontologías de dominio completas	65
Ilustración 54. NeOn: reutilización de declaraciones ontológicas	67
Ilustración 55. NeOn: evaluación de ontologías	69
Ilustración 56. NeOn: tareas para la alineación entre ontologías	70
Ilustración 57. NeOn: reingeniería de recursos no ontológicos	72
Ilustración 58. NeOn: modularización de ontologías	73
Ilustración 59. NeOn: Ejemplo de patrón conceptual de participación n-aria	75
Ilustración 60. NeOn: proceso de evolución de la ontología	76
Ilustración 61. NeOn: adaptación lingüística y cultural ontológica	77
Ilustración 62. NeOn, transformaciones NOR en ontologías	79
Ilustración 63. NeOn: diagrama de Gantt con la planificación estimada	91
Ilustración 64. Integración de recursos ontológicos	107
Ilustración 65. Ejemplo de CSV en bruto del edificio Alice Perry	107
Ilustración 66. Segmento del excel con anotaciones sobre los sensores	108
Ilustración 67. Metadatos de los CSV en JSON-LD	112
Ilustración 68. Transformación de población para un NOR en la ontología	114
Ilustración 69. Detalle del excel con la columna de tipo de dispositivo	116

Ilustración 70. Detalle del excel con el esquema gráfico AHU101	116
Ilustración 71. Detalle del diagrama general para la ontología, elaborado con WebVOWL	119
Ilustración 72. Diagrama de actividades	120
Ilustración 73. Protégé: creación de URIs, anotaciones generales e importaciones.....	124
Ilustración 74. Protégé: creación de prefijos.....	124
Ilustración 75. Protégé: pestaña “Entities” y árbol de clases	125
Ilustración 76. Protégé: instanciar un individuo (Parte 1).....	125
Ilustración 77. Protégé: instanciar un individuo (Parte 2).....	126
Ilustración 78. Resultado de la evaluación de la ontología en OOPS!.....	128
Ilustración 79. Fila con datos en CSV.....	132
Ilustración 80. Fila del CSV en formato JSON	133
Ilustración 81. Esquema OWL-RDF para cada medida.....	133
Ilustración 82. Ejemplo de medidas en OWL-RDF	134
Ilustración 83. Grafica sobre datos RDF asociada a la PQ PC06	136
Ilustración 84. Histograma sobre datos RDF asociados a temperaturas en sala.....	137
Ilustración 85. Diagrama de Caja sobre datos RDF asociados a temperaturas en sala	137
Ilustración 86. Pantalla inicial de Protégé.....	162
Ilustración 87. Pasos para la instalación de Anaconda.....	163
Ilustración 88. Interfaz Jupyter Notebook.....	164
Ilustración 89. Jupyter Notebook: crear directorio y notebook.....	164

Lista de tablas

Tabla 1. <i>Namespaces</i> en cabeceras OWL / OWL2	32
Tabla 2. Elementos de información en cabeceras OWL/OWL2	32
Tabla 3. Entidades principales para ifcHvacDomain	43
Tabla 4. Conversión de términos adoptados entre EXPRESS y OWL	44
Tabla 5. Metodología NeOn, tabla de fases y procesos	57
Tabla 6. Metodología NeOn, familia de modelo en cascada	58
Tabla 7. Metodología NeOn, preguntas y versión de modelo	59
Tabla 8. Actor Usuario General	81
Tabla 9. Actor Usuario Técnico	81
Tabla 10. Requisitos no funcionales antes de su validación	82
Tabla 11. Requisitos funcionales antes de la validación	85
Tabla 12. ORSD de la ontología	90
Tabla 13. Resumen de planificación estimada por horas	92
Tabla 14. Listado de riesgos	94
Tabla 15. Listado de riesgos priorizado	94
Tabla 16. Risk-01. Retrasos en la planificación	95
Tabla 17. Risk-02. Desconocimiento de las tecnologías a utilizar	95
Tabla 18. Risk-03. Cambios en la especificación de requisitos	95
Tabla 19. Risk-04. Problemas de salud	95
Tabla 20. Risk-05. Ausencia del tutor	96
Tabla 21. Risk-06. Falta de tiempo	96
Tabla 22. Risk-07. Problemas de hardware	96
Tabla 23. Risk-08. Falta de experiencia	97
Tabla 24. Risk-09. Llegar a un punto de bloqueo en la documentación y revisión	97
Tabla 25. Risk-10. Mala Planificación	97
Tabla 26. Presupuesto simulado	98
Tabla 27. Presupuesto real	98
Tabla 28. NeOn: selección de ontologías candidatas	104
Tabla 29. NeOn, tabla para la valoración de ontologías de dominio	104
Tabla 30. NeOn, tabla para la selección de ontologías de dominio	105
Tabla 31. Muestra de reutilización de elementos	106
Tabla 32. Ontologías importadas por SAREF y SAREF4BLDG	106

Tabla 33. Componentes y características de la AHU101	109
Tabla 34. Campos asociados al CSV de entrada.....	111
Tabla 35. Esquema y ejemplos de URIs	115
Tabla 36. Instanciación de elementos de la ontología (Parte 1)	117
Tabla 37. Instanciación de elementos de la ontología (Parte 2)	117
Tabla 38. Descripción y solución a escollos localizados en OOPS!	131
Tabla 39. CQ validadas sobre los datos RDF generados	135
Tabla 40. Resultado parcial de la consulta SPARQL de la MD_PC06	136
Tabla 41. Tareas de la Fase 1	139
Tabla 42. Tareas de la Fase 2	140
Tabla 43. Tareas de la Fase 3	140
Tabla 44. Tareas de la Fase 4	140
Tabla 45. Tareas de la Fase 5	141
Tabla 46. Resumen de planificación real por horas.....	141

Capítulo 1

1. Introducción

La sociedad de la información en la que nos encontramos inmersos se caracteriza por el uso intensivo de las tecnologías de la información y comunicación (TIC) y su capacidad para generar datos y nuevas fuentes de información disponibles, siendo las máquinas las que acaparan el mayor porcentaje de intercambio de esta información. Desde hace más de una década instituciones y gobiernos intentan dirigir este volumen de datos hacia un uso responsable y sostenible en lo que se ha dado en denominar *Smart Cities* y *Smart Building*, debido, entre otras cosas, a su gran potencial en ámbitos tanto científicos como económicos si se aplican tratamientos adecuados a los mismos.

Dentro de una edificación, el sistema de climatización es uno de los elementos que realiza mayor consumo energético, pudiendo alcanzar cuotas de más del 40% sobre el gasto total. Los sistemas de climatización comprenden multitud de variables heterogéneas y datos dinámicos, específicos de cada edificación, que complican la creación de sistemas de la gestión eficiente por su complejidad dinámica. El desarrollo de la Inteligencia Artificial y las técnicas de Minería de Datos hacen posible abordar este problema desde múltiples perspectivas, siendo el enfoque ontológico y los lenguajes de gestión de conocimiento OWL y RDF una de ellas. Estos lenguajes no solo brindan la oportunidad de integrar y almacenar información de estas fuentes heterogéneas, sino que además permiten generar y extraer conocimiento nuevo de dichos datos por lo que su aprovechamiento se hace todavía más imprescindible.

1.1. Contexto

La idea de este proyecto surgió como colofón en un trabajo de la asignatura de Ingeniería del Conocimiento impartida por el profesor D. Carlos Alonso. Dicho trabajo abordaba las ontologías y su implantación como método de representación de conocimiento avanzado, al incluir en su representación contenido semántico y sintáctico que permite inferir conocimiento nuevo mediante técnicas adecuadas.

En paralelo el Departamento de Informática (ATC, CCIA, LSI) realiza estudios de investigación con *Data Mining* sobre *IoT* en *Smart Building*. Para ello utiliza la información generada por sensores de un edificio. Desde dicho departamento se llevan realizando labores de estudio y proyectos Big Data, pero partiendo de los datos relacionales obtenidos de los sensores, por lo que, de manera natural, surge el interés por realizar una revisión sobre el alcance y validez del tratamiento de dichos datos desde un punto de vista ontológico.

1.2. Objetivos

El objetivo general de este trabajo es explorar la adaptación de un entorno de procesamiento de datos IoT¹ basado en datos tabulados, a las tecnologías de procesamiento de información basadas en el uso de ontologías. Se partirá de la información sobre el sistema de climatización AHU² o HVAC³ de un edificio inteligente y se modelizará dicho sistema mediante la ontología estándar SAREF⁴.

Posteriormente se diseñará y desarrollará un entorno básico de pruebas que confirme el posible aprovechamiento de esta tecnología, transformando mediciones obtenidas en forma relacional a ontológica, generando para ello un mapeado de los datos como Datos Enlazados. Se debe validar la información almacenada en el nuevo sistema mediante la aplicación de búsquedas y métricas de estudio estadístico como en el sistema actual, con algunos ejemplos básicos. Quedan fuera del ámbito de este estudio las capacidades de inferencia de conocimiento asociadas al nuevo modelo ontológico.

1.2.1. Objetivos de desarrollo

Los objetivos generales que se han definido para este Trabajo Fin de Grado son los siguientes:

- Llevar a cabo una revisión sobre el estado actual de las ontologías y la web semántica, de cara a sus aplicaciones en inteligencia artificial y concretamente aplicada al ámbito de los sistemas de climatización de edificaciones, como alternativa al sistema actual basado en tablas de datos relacionales
- Modelizar con SAREF y SAREF4BLDG el sistema de climatización AHU-101 del *Alice Perry Engineering Building* de la NUI en Galway, Irlanda
- Trasladar la ontología modelizada a un entorno de desarrollo informático sólido para poder trabajar con ella. Facilitar la realización de desarrollos posteriores, que permitan estudiar el potencial ontológico como sistema para extraer información útil y generar nuevo conocimiento
- Implementar un entorno de estudio que permita el procesado de los datos tabulados actuales al modelo de Datos Enlazados con la ontología desarrollada, que, en la medida de lo posible, permita la entrada de datos en otros formatos, incluyendo *streaming* (tiempo real) y garantizando flexibilidad de acceso al flujo de datos en todas las fases de su transformación
- Implementar un entorno gráfico sencillo que permita visualizar los datos ontológicos de una manera cómoda e intuitiva, para comprobar su validez mediante la realización de varias consultas y gráficas, incluyendo estadísticas descriptivas básicas

1.2.2. Objetivos personales

El Trabajo Fin de Grado se corresponde con una asignatura dentro del plan de estudios, y como en todas las asignaturas, se transmite conocimiento. Mis objetivos personales al comenzar el proyecto han sido:

- Mejorar la comprensión y funcionamiento de un proyecto planificado
- Reforzar y ampliar mis conocimientos sobre ontologías, Datos Enlazados y sus aplicaciones prácticas
- Familiarizarme con conceptos de IoT y estudiar su potencial como sistema de extracción de información útil

¹ Internet Of Things, Internet de las Cosas

² AHU: Air Handling Unit, unidad de tratamiento del aire

³ HVAC: Heating Ventilation Air Conditioning, calefacción ventilación y aire acondicionado

⁴ The Smart Applications REFERENCE (SAREF) ontology <https://saref.etsi.org/>

1.3. Estructura de la memoria

Este documento se estructura en los siguientes capítulos:

- **Capítulo 1. Introducción:** se describe el origen y la motivación que impulsan el proyecto, así como los objetivos generales que se pretenden alcanzar.
- **Capítulo 2. Contexto general:** se define el marco contextual general del proyecto, detallando la situación actual de todas las tecnologías y actores implicados en el mismo.
- **Capítulo 3. Contexto específico:** se detallan los elementos específicos que implican los requisitos del desarrollo del TFG.
- **Capítulo 4. Metodología:** describe y define la metodología aplicada.
- **Capítulo 5. Plan de proyecto:** se establece el propósito de la ontología y el programa a desarrollar, su alcance y la calendarización estimada del proyecto.
- **Capítulo 6. Plan de riesgos y presupuestos:** plan de posibles riesgos adaptado al contexto de un TFG y estimaciones de costes.
- **Capítulo 7. Tecnologías y software empleados:** resumen de las principales tecnologías y software utilizados para la realización del proyecto
- **Capítulo 8. Desarrollo de la ontología:** contiene la aplicación de la metodología que obtiene como resultado final la ontología y el programa objetivo del proyecto.
- **Capítulo 9. Exploración y análisis de los datos RDF:** se realiza una revisión más detallada de cómo se transforman los datos al formato RDF y se analiza su utilidad mediante consultas, gráficas y estadísticas descriptivas básicas sobre el nuevo formato.
- **Capítulo 10. Seguimiento del proyecto:** se describe la evolución del proyecto cronológicamente a través de las fases realizadas.
- **Capítulo 11. Conclusiones y líneas de trabajo futuras:** describe los resultados del proyecto y las incidencias registradas en su ejecución. Presenta, además, líneas de trabajo futuras que podrían desarrollarse.
- **[Bibliografía](#):** contiene las referencias bibliográficas empleadas en el TFG.

Para finalizar se añaden cuatro apéndices:

- **[Apéndice A. Contenido de la memoria:](#)** se detalla la estructura y contenido de los documentos almacenados en el GitHub del proyecto.
- **[Apéndice B. Reutilización de elementos:](#)** se describen los elementos (clases, relaciones y propiedades) de las ontologías estándar utilizados para el desarrollo del proyecto.
- **[Apéndice C. Ontología AHU10 Alice Perry Building:](#)** se presentan las clases, relaciones, propiedades e instancias de la ontología desarrollada.
- **[Apéndice D. Instalación del entorno:](#)** se describen las instrucciones para instalar el entorno de desarrollo de la aplicación y la ontología.

Capítulo 2

2. Contexto General

En este capítulo se resume y expone la información recopilada y analizada durante la etapa de estudio y documentación, que servirá como base imprescindible a la hora de entender el contexto que rodea al objetivo del TFG. Se presentarán los medios y tecnologías involucrados en el contexto de los Edificios Inteligentes, la Web Semántica y las ontologías como sistema de representación del conocimiento.

2.1. Metodología BIM (Building Information Modeling)

Se trata de una metodología de trabajo colaborativa para la creación y gestión de un proyecto de construcción en el ámbito de la Ingeniería Civil y Arquitectura. Su objetivo es centralizar toda la información del proyecto en un modelo de información digital creado por todos los agentes que intervienen en el ciclo de vida de la construcción: desde la ideación del proyecto, aprobación, construcción, uso, hasta su posible demolición y reciclaje. Durante todo el ciclo de vida del proyecto se utilizará un software dinámico de modelado en tres dimensiones y en tiempo real y costes (considerados en el modelo como 4ª y 5ª dimensión), que hará disminuir la pérdida de tiempo y recursos al permitir una permanente actualización y comunicación entre todos los agentes del proceso.

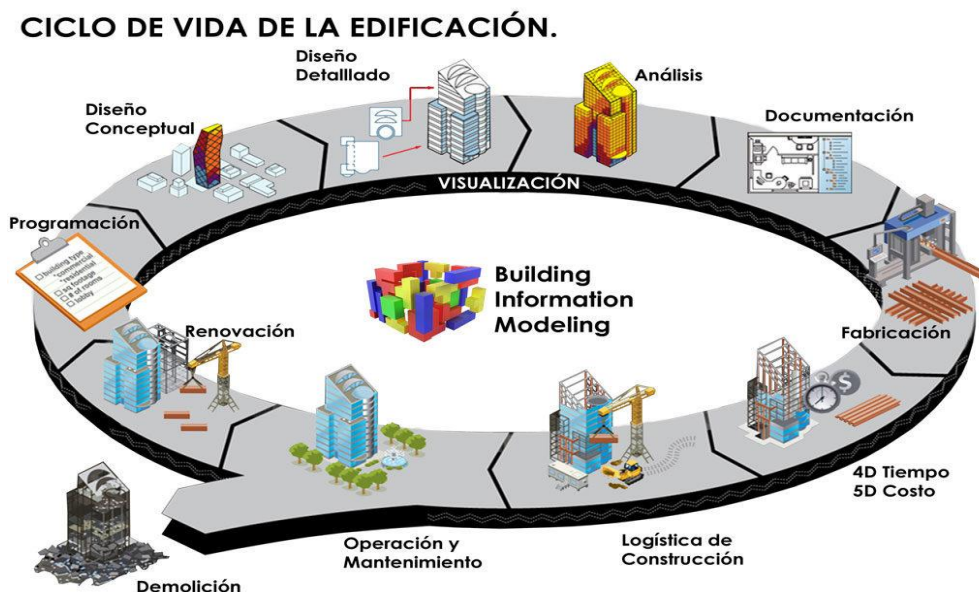


Ilustración 1. Ciclo de vida de un proyecto con BIM [1]

La tecnología parte de estudios realizados en los años 70 del siglo pasado [2], pero no comenzó a desarrollarse por limitaciones tecnológicas hasta principios de este siglo, siendo Autodesk en 2004 quien

posicionó su producto e impulsó los estándares actuales para la industria (IFC, Open BIM, IPD). En 2002 se crea el primer proyecto puramente BIM en Finlandia.

Mediante BIM se modela una construcción civil, por ejemplo, un edificio, unificando y validando los estándares de cada actor en la obra: arquitecto, contratista, electricista, seguridad y riesgos, interiorista, expertos en salud, etc. Es capaz de centralizar sus especificaciones y requisitos, visualizarlas, probarlas y analizar colisiones o simular el funcionamiento para observar desgastes, fallos, consumo de energía, etc., en tiempo real o simulado, con los correspondientes costes asociados en cada momento, hasta su ciclo final de demolición, renovación o reciclaje. Todo el soporte del proceso se incluye en la ISO 19650.

Se encuentra fuertemente implantada en el mundo anglosajón (en EE.UU. se operan proyectos desde 2006 y R.U. lo fija como estándar en 2010 y obligatorio en 2016) y en muchas zonas de Sudamérica. La Unión Europea en la Directiva 2014/24/UE establece el marco de adopción para la metodología BIM y su necesidad de implantación en 2018. Paralelamente en España se abre el estudio de implantación con base a las normas dictadas en 2012 por AENOR en el Comité de Normalización AEN/CTN 41/SC13. Se estandariza en 2018 y es de obligatoriedad para licitaciones públicas desde 2020. Podemos hacernos una idea de su implantación en la Ilustración 2:



Ilustración 2. Estado de la adopción de BIM en el mundo [3]

2.1.1. Open BIM y el formato IFC (Industry Foundation Classes)

Según la definición del propio organismo internacional de cooperación encargado del seguimiento de los estándares BIM desde 1994, *Building SMART International*⁵, el término Open BIM referencia “un enfoque colaborativo para el diseño y la construcción de los edificios basados en estándares y flujos de trabajo abiertos”. Su objetivo fundamental es agilizar el intercambio de datos entre todos los actores

⁵ *Building SMART International*, <https://www.buildingsmart.org/>

involucrados en la creación de un modelo BIM que cubra todos los campos de aplicación posibles. Desde el diseño a la construcción, desde el funcionamiento del edificio hasta su demolición y al reciclado de componentes y materiales, al finalizar el ciclo de vida de la estructura. El *buildingSMART International* además de coordinar el estándar open BIM, desarrolla el diccionario de datos (*buildingSMART Data Dictionar*) y el modelo de datos neutral IFC (*buildingSMART Data Model*) [4].

Un requisito fundamental para el Open BIM es el uso de formatos de datos abiertos y neutrales. La metodología BIM se implementa en varios formatos de archivo escritos en un lenguaje propio según los diferentes estándares mencionados con anterioridad. De los cinco estándares abiertos a nivel mundial, IFC es la solución más elegida para el Open BIM y se considera la principal. IFC es un formato de datos neutral y gratuito, no controlado por los desarrolladores de software, que favorece la interoperabilidad entre programas de distintas compañías (ArchiCAD, Revit, Allplan, etc.) que implementan BIM sin pérdidas ni distorsiones en los datos. Está registrado como estándar oficial ISO 16739:2013.

En el formato de datos se define cada clase (objetos, procesos, personas, conceptos abstractos) y se guardan las diferentes relaciones entre ellas. Siempre teniendo en cuenta que todos los elementos y sus relaciones han de ser legibles e intercambiables entre diferentes aplicaciones de software, optimizando también el sistema de recuperación y guardado. Por tanto:

- un modelo IFC contiene tanto entidades geométricas, como no geométricas
- un modelo IFC refleja la geometría del objeto y los datos asociados a sus elementos
- un archivo IFC facilita la transferencia de datos entre aplicaciones mediante la exportación de datos
- IFC es un formato correctamente documentado, libre y abierto con una interfaz para la exportación y la importación. Siguiendo al estándar IFC los proveedores de software garantizan la interoperabilidad entre todas y cada una de las herramientas y aplicaciones BIM

Cada clase modelada contiene un esquema de datos y una estructura de formato para archivo de intercambio. Asociados a una clase encontraremos los siguientes estándares dentro de IFC:

- Lenguaje de especificación de datos EXPRESS, definido en ISO 10303-11,
- Lenguaje de definición de esquema XML (XSD), definido en ISO 10303-28

La modelización de clases se realiza mediante el paradigma de la programación orientada a objetos, basado en el lenguaje de programación C++, al que se le añade una semántica pre-post implementada desde el ámbito de las Ingenierías Industriales. A lo largo de los últimos 20 años existe un vasto desarrollo de todos los elementos que forman parte de un proceso constructivo en IFC, debidamente estandarizado, aunque estas adaptaciones comparten un sesgo marcado por la industria de Ingeniería Civil, que complican la paridad con los enfoques imperantes en Ciencias de la Computación y dificultan su total compatibilidad.

Los propósitos de diseño de IFC garantizan una interoperabilidad dentro del dominio de Ingeniería Civil o Industrial y siempre supeditados al esquema definido EXPRESS y la validez de sus definiciones garantizada mediante los desarrollos en XSD. La tendencia actual del sector consiste en la creación de aplicaciones de gestión e intercambio de información basada en tecnologías web semánticas que garanticen la interoperabilidad de datos, el intercambio de datos flexible, la gestión de datos distribuidos y el desarrollo de herramientas reutilizables. Objetivos que incluso se priorizan sobre los iniciales del IFC, extendiendo ya las correspondientes normas ISO que regulan, sobre las tecnologías semánticas (ifcOWL) y que se comentarán en el apartado de ontologías.

2.1.2. Arquitectura IFC, estándar EXPRESS

El modelo de arquitectura de IFC se estructura en cuatro capas: capa de recursos, capa central, capa de interoperabilidad y capa de dominio/aplicación. En IFC podemos definir clases nuevas como subclases de una clase ya existente, de la cual heredan propiedades. La Ilustración 3 muestra un extracto de la jerarquía IFC. Todas las clases IFC (excepto las clases de recursos) se derivan de la superclase *IfcRoot*. Esta clase permite establecer atributos básicos del modelo como identidad, descripción y propiedad. *IfcRoot* arroja tres clases fundamentales para las categorías definidas: objetos (*IfcObjectDefinition*), relaciones (*IfcRelationship*) y propiedades (*IfcPropertyDefinition*). Cada una de estas tres clases tiene su respectivo árbol de herencia de subclases. Se puede leer un análisis más profundo en el artículo [5] y en la documentación técnica de la última versión [6].

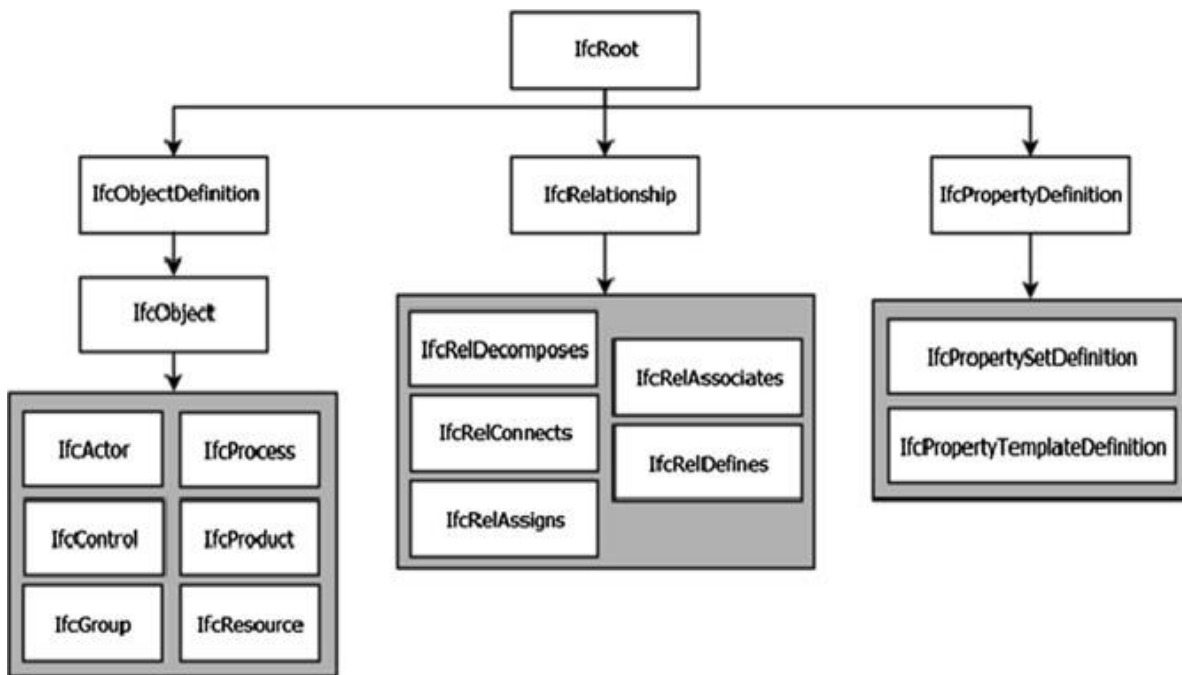


Ilustración 3. Estructura de capas de la arquitectura IFC [5]

La especificación IFC es por tanto un gran diccionario que incluye términos, conceptos y elementos de especificación de datos que se originan a partir del uso de las disciplinas, oficios y profesiones de la industria de la construcción y la gestión de instalaciones. La nomenclatura para la inclusión de términos suele incluir un prefijo común (“*ifc*” para reglas o funciones, “*Pset*” para propiedades, etc.) seguidos de una descripción con palabras en inglés en *Camelcase* sin prefijo.

Cada clase del estándar IFC genera un gran árbol de relaciones siguiendo la estructura detallada en la Ilustración 3. La línea principal de este árbol se corresponde con la que describe los objetos, pero también existen definiciones (“SCHEMA” en el estándar) que definen propiedades, geometrías, materiales, unidades de medida, o cualquier consideración que cubra las necesidades y acciones en el entorno de la arquitectura y construcción. Un ejemplo de estos árboles lo podemos observar en la Ilustración 4 y la Ilustración 5 con el elemento “*ifcAlarm*” obtenida de la documentación general en la última revisión del estándar. La definición que da IFC para esta clase es [7]:

“Una alarma es un dispositivo que señala la existencia de una condición o situación que está fuera de los límites de la expectativa normal o que activa dicho dispositivo.

Las alarmas incluyen la provisión de botones de rotura de vidrio y cajas manuales que se utilizan para activar alarmas.”.

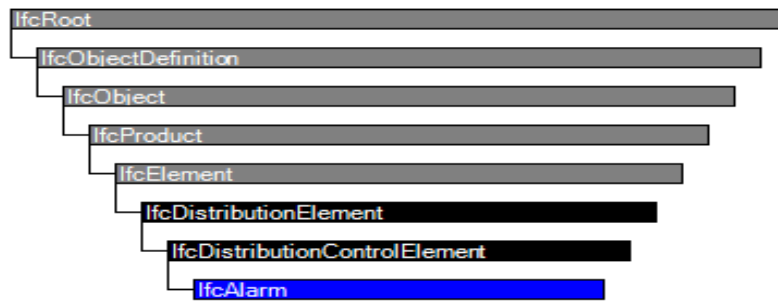


Ilustración 4. “ifcAlarm”: esquema de definiciones heredadas para supertipos (clases padres) [7]

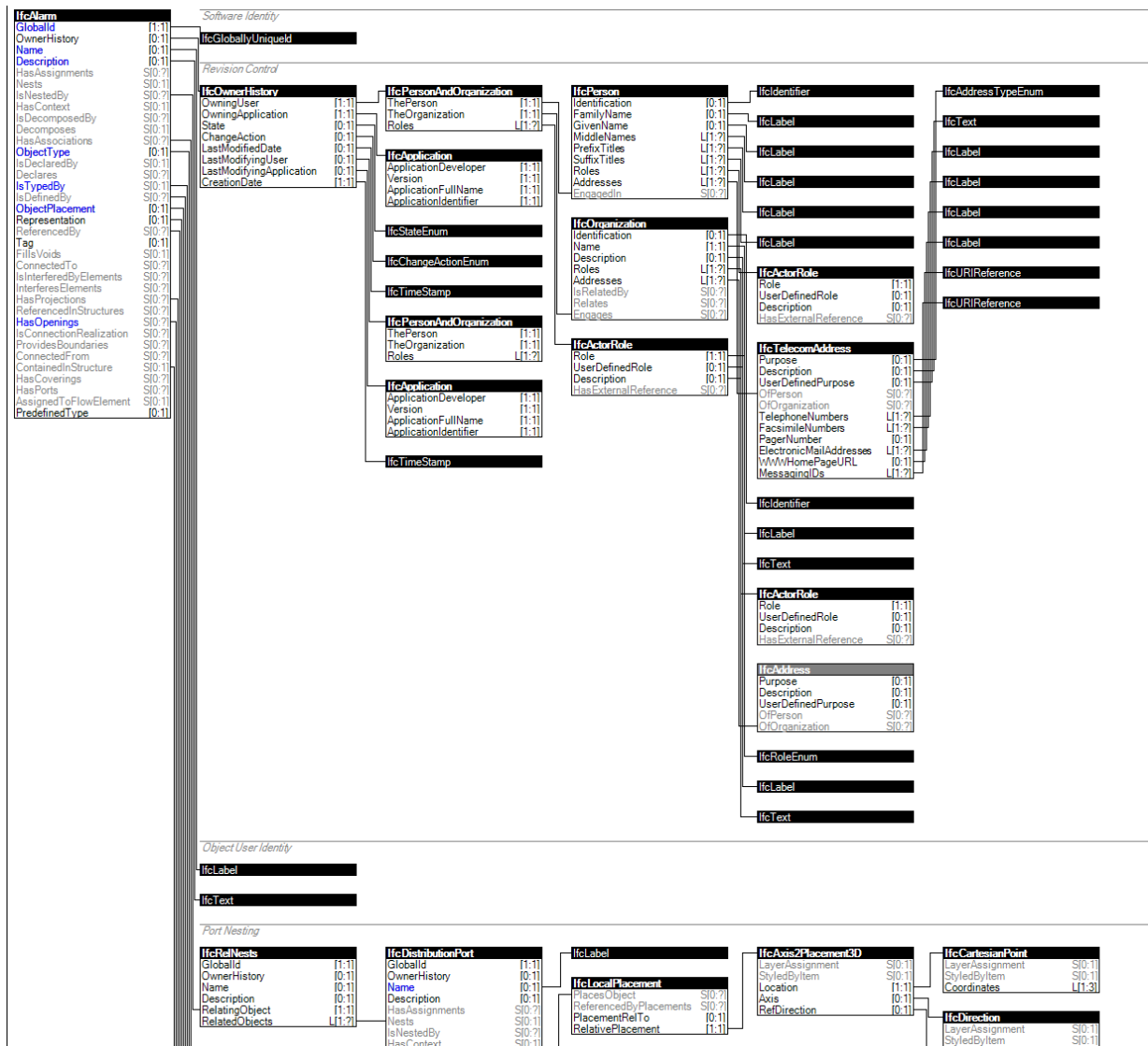


Ilustración 5. “ifcAlarm”: extracto del esquema de definición de recursos (clase y sus relaciones) [6]

EXPRESS (ISO 10303-11) es el lenguaje de programación del estándar en el que se implementa una definición o SCHEMA (esquema). A la hora de implementarlo se puede definir dos maneras: textual y gráfica. Para la verificación formal mediante herramientas estandarizadas es mejor la representación textual en formato de archivo ASCII. La representación gráfica, denominada EXPRESS-G, suele ser más conveniente en el uso humano para facilitar explicaciones y tutoriales, aunque no es capaz de representar todos los detalles que se alcanzan en forma textual.

Dentro de un SCHEMA se pueden definir varios tipos de datos junto con restricciones estructurales y reglas algorítmicas, incluyendo la posibilidad de validar formalmente una población de tipos de datos, para verificar todas las reglas estructurales y algorítmicas. Un ejemplo muy sencillo con algunas entidades y relaciones humanas de parentesco lo tenemos en la Ilustración 6 obtenido de la información que aporta la Wikipedia para EXPRESS [8] :

- familia sea un EXPRESS SCHEMA
- persona sea un ENTITY SUPERTYPE Hombre/Mujer sean los ENTITY SUBTYPE.
- persona fue definida como una variable ABSTRACT (abstracta) con una condición ONEOF (escoge un SUBTYPE).
- el nombre sea una característica mandante de cada ocurrencia (persona)
- padre/madre sean unas características opcionales de cada ocurrencia (persona)
- Hombre puede tener el papel (opcional) de padre, pero padre es obligatoriamente Hombre
- Mujer puede tener el papel (opcional) de madre, pero madre es obligatoriamente Mujer

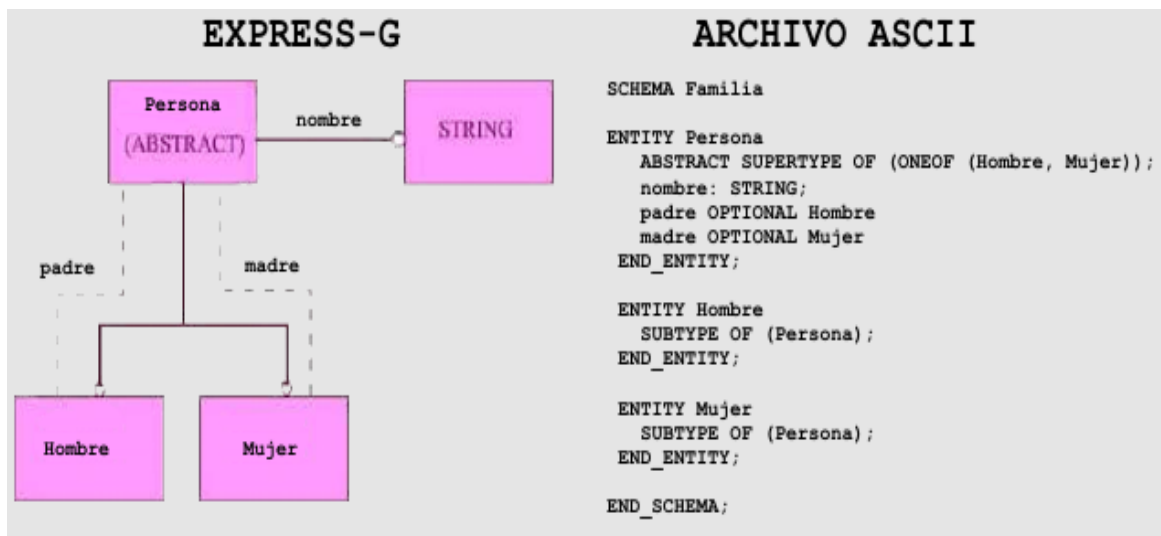


Ilustración 6. Ejemplo simple en EXPRESS [8]

Un ejemplo real más complejo que incluye restricciones aritméticas, lógicas y funciones Ilustración 7:

```

SCHEMA shape_tolerance_schema;
REFERENCE FROM measure_schema -- ISO 10303-41
(derive_dimensional_exponents, dimensional_exponents,
length_measure_with_unit, measure_with_unit,
measure_value, plane_angle_measure_with_unit);
TYPE area_unit_type = EXTENSIBLE ENUMERATION OF
(circular,
square,
rectangular);
END_TYPE;
ENTITY geometric_tolerance_relationship;
name : label;
description : text;
relating_geometric_tolerance : geometric_tolerance;
related_geometric_tolerance : geometric_tolerance;
END_ENTITY;
RULE subtype_exclusiveness_geometric_tolerance FOR
(geometric_tolerance);
WHERE
WR1: SIZEOF(QUERY (gt < * geometric_tolerance | NOT
(type_check_function(gt,
['SHAPE_TOLERANCE_SCHEMA.ANGULARITY_TOLERANCE'],
2 ))) = 0;
END_RULE;
FUNCTION sts_get_product_definition_shape
(input : geometric_tolerance_target) : product_definition_shape;
CASE TRUE OF
('SHAPE_DIMENSION_SCHEMA.DIMENSIONAL_LOCATION' IN
TYPEOF(input)) :
RETURN(input$shape_aspect_relationship.relatiing_shape_aspects
hape_aspect.of_shape);
('SHAPE_DIMENSION_SCHEMA.DIMENSIONAL_SIZE' IN
TYPEOF(input)) :
RETURN(input$dimensional_size.applies_to$shape_aspect.of_shap
e);
('PRODUCT_PROPERTY_DEFINITION_SCHEMA.PRODUCT_DE
FINITION_SHAPE' IN TYPEOF(input)) :
RETURN(input);
('PRODUCT_PROPERTY_DEFINITION_SCHEMA.SHAPE_ASPEC
T' IN TYPEOF(input)) :
RETURN(input$shape_aspect.of_shape);
OTHERWISE : RETURN(?);
END_CASE;
END_FUNCTION;
END_SCHEMA; -- shape_tolerance_schema
    
```

Ilustración 7. Ejemplo de SCHEMA con restricciones aritméticas, lógicas y funciones [9]

2.1.3. Arquitectura IFC, estándar XML (XSD)

IFC XML es una definición de esquema XML⁶ XSD⁷, derivada del modelo IFC EXPRESS. El estándar ISO 10303-28 establece cómo realizar la transcripción desde EXPRESS a XML. El esquema IFC XML consta de dos partes:

- archivos XSD: son dos, uno más pequeño de configuración que contine el esquema común para todos los modelos EXPRESS, incluye las definiciones para la sección de encabezado y los tipos de datos generales equivalentes implementados en EXPRESS. Un archivo de tamaño mayor que incluye el número de la versión en su nombre y contiene las definiciones XSD de todas las clases, relaciones, atributos y tipos de datos específicos de IFC.
- Archivo XML: es el archivo que debe usarse para la validación en línea. Contiene los espacios de nombres y las ubicaciones de los archivos XSD con la definición del esquema

Para finalizar este apartado dedicado a IFC y continuando con el ejemplo de “*ifcAlarm*”, extraemos su representación para los estándares EXPRESS y XML, Ilustración 8.

XML Specification

```
<xs:element name="IfcAlarm" type="ifc:IfcAlarm" substitutionGroup="ifc:IfcDistributionControlElement" nillable="true"/>
<xs:complexType name="IfcAlarm">
  <xs:complexContent>
    <xs:extension base="ifc:IfcDistributionControlElement">
      <xs:attribute name="PredefinedType" type="ifc:IfcAlarmTypeEnum" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

EXPRESS Specification

```
ENTITY IfcAlarm
SUBTYPE OF (IfcDistributionControlElement);
  PredefinedType : OPTIONAL IfcAlarmTypeEnum;
WHERE
  CorrectPredefinedType : NOT(EXISTS(PredefinedType)) OR
  (PredefinedType <> IfcAlarmTypeEnum.USERDEFINED) OR
  ((PredefinedType = IfcAlarmTypeEnum.USERDEFINED) AND EXISTS (SELF\IfcObject.ObjectType));
  CorrectTypeAssigned : (SIZEOF(IsTypedBy) = 0) OR
  ('IFCBUILDINGCONTROLSDOMAIN.IfcAlarmType' IN TYPEOF(SELF\IfcObject.IsTypedBy[1].RelatingType));
END_ENTITY;
EXPRESS-G diagram
```

Ilustración 8. Representaciones Formales para *IfcAlarm* [7]

2.2. Evolución de la World Wide Web

En 1989 Tim Berners-Lee inventó la World Wide Web [10] describiendo sus principios [11], pero esta web ha ido evolucionando desde su primera implementación en la década del 90 con la Web 1.0 concebida como una web estática. Las páginas web se consideraban como documentos de hipertexto (texto e imágenes) estáticos de lectura alojadas en un servidor y presentadas a los usuarios que eran meros consumidores de contenidos mediante consultas, pero no productores. Los documentos generados acumulaban grandes cantidades de información, pero los usuarios no podían interactuar o contribuir con dicho contenido ya que las páginas eran creadas, mantenidas y modificadas por un grupo muy restringido de usuarios.

⁶ XML: *Extensible Markup Language*, <https://www.w3.org/TR/xml/>

⁷ XSD: *XML Schema Definition Language*, <https://www.w3.org/TR/xmlschema11-1/>

No existe un momento concreto en el que la web pasó a denominarse Web 2.0, pero múltiples fuentes [12] [13] señalan que este término empezó a utilizarse en una conferencia de Tim O'Reilly en el año 2004 en la que colaboraba en esta discusión con Dale Dougherty. Esta nueva evolución de la web trae dinamismo, permitiendo a los usuarios colaborar para añadir contenido (redes sociales, plataformas para compartir contenido multimedia, foros, consultas online, etc.). Esta transición no implicó un cambio drástico en las tecnologías, fue más bien una transición desde la web estática hasta la actual, colaborativa y dinámica.

La web 2.0 sigue teniendo carencias de interpretación de sus contenidos tanto para humanos como para máquinas por lo que surge la Web 3.0 como una extensión de la web 2.0 dotada de significado o Web Semántica. Se define en el año 2000 buscando la introducción explícita de un significado en los recursos y servicios alojados en la Web, así como la descripción de las relaciones existentes entre los propios recursos, de tal manera que tanto el significado como las descripciones puedan procesarse a nivel computacional [14].

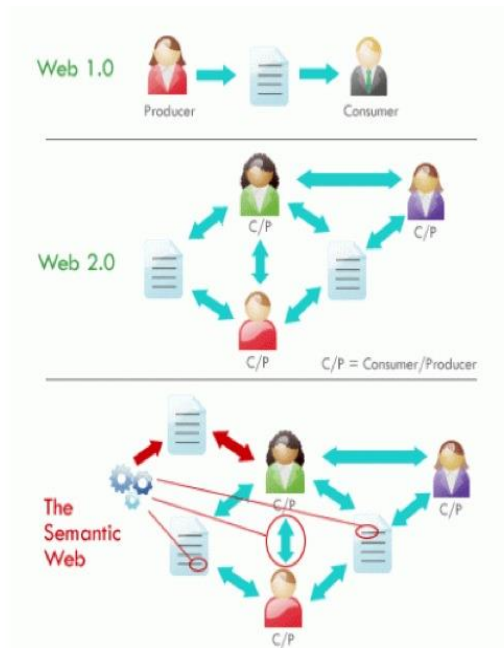


Ilustración 9. Esquema para la Web 1.0, 2.0 y 3.0 [15]

Existe un debate abierto sobre la cuestión de haber realizado ya la transición hacia la Web 3.0. acrecentado por la ausencia de una definición clara sobre qué debe ser la web 3.0, pero sí existe consenso sobre las tecnologías que deben ser utilizadas y como deben manejarse los datos. Estas tecnologías sí que están siendo implantadas cada vez con mayor uso, sobre todo por sus conexiones con Data Mining y la Inteligencia Artificial.

Actualmente ya existen desarrollos para la Web 4.0 o web simbiótica que incluyen razonamiento y gestión por parte de las máquinas. El objetivo es alcanzar una web ubicua donde los humanos y máquinas se comuniquen sin ninguna barrera, generando decisiones inteligentes.

2.3. La Web Semántica

T. Berners-Lee creador de la WWW define la Web Semántica como “una extensión de la actual Web con mayor significado que facilita que los computadores y las personas trabajen en cooperación” [16] o como “una red de datos que pueden ser procesados directa o indirectamente por las máquinas” [17] . Se crea además el

proyecto *World Wide Web Consortium*⁸, nombrado actualmente como W3C, como parte de la iniciativa de la Web Semántica, con el objetivo de normalizar o regular los cambios que se vayan produciendo durante la evolución de la WWW. El W3C propone la siguiente definición oficial para la Web Semántica [18]:

“La Web Semántica proporciona un marco común que permite que los datos sean compartidos y reutilizados a través de aplicaciones, empresas y fronteras comunitarias. Es un esfuerzo colaborativo liderado por el W3C con la participación de un gran número de investigadores y socios industriales. Está basado en *Resource Description Framework* (RDF)”

Esta definición hace referencia a una de las tecnologías utilizadas para desarrollar la Web Semántica: RDF que junto con el lenguaje XML (*eXtensible Markup Language*) forman la base para expresar la información e interpretar su significado y que se analizarán más adelante.

Conforme a la definición, como objetivo fundamental perseguido en su desarrollo, está el definir y estructurar los datos de manera que se puedan crear relaciones entre ellos de una forma eficiente, siendo posible la publicación, consumo o integración de los mismos mediante una automatización de procesos con aplicaciones. Al estructurar los datos de esta nueva manera, se abordan inconvenientes ya conocidos (desambiguar conceptos, sobrecarga de información, etc.) y otros nuevos (técnicas de búsqueda y extracción de información o interoperabilidad entre diferentes sistemas de información, capacidad de realizar razonamientos y obtener conclusiones, etc.). En cualquier caso, aunque desde la primera publicación sobre Web Semántica, se han desarrollado estándares, recomendaciones e implementado multitud de proyectos, hasta el momento no se puede decir que la idea de la Web Semántica esté totalmente implementada en la práctica.

El consorcio W3C estructura la Web Semántica en capas, con apenas variaciones desde su definición inicial en 2001 [19] [20] :

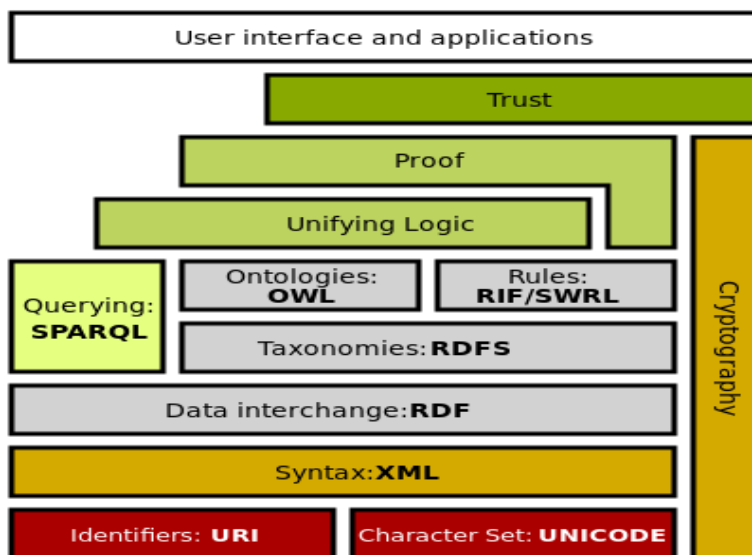


Ilustración 10. Modelo de capas de la Web Semántica [21]

“La primera capa se compone de los estándares que permiten la ubicación de los recursos en Internet: URI⁹ y el formato UNICODE¹⁰. Sobre esta primera capa se dispone el XML como base sintáctica que estructura el contenido de Internet y XML-*namespaces* para relacionar elementos y propiedades. La tercera

⁸ W3C Consortium: <http://www.w3.org/>

⁹ URI: *Uniform Resource Identifier*, https://en.wikipedia.org/wiki/Uniform_Resource_Identifier

¹⁰ UNICODE o Unicode Estándar [ISO/IEC 10646](https://www.iso.org/standard/52093.html) revisado por [Unicode Consortium](http://www.unicode.org/)

capa emplea RDF como modelo para describir propiedades de los recursos, permitiendo la descripción de clases, objetos y propiedades, entre otros elementos, mediante taxonomías. Del establecimiento de un marco común para estas descripciones se encarga el nivel superior: RDFS, que define dichos conceptos con un vocabulario consensuado y lo representa mediante redes de nodos interconectados organizados como una ontología. Esta ontología tiene además la posibilidad de aplicar reglas a dichas redes y descripciones. A continuación, aparece una capa lógica que permite realizar consultas y aserciones sobre los modelos que suele estar enlazada con una capa de fiabilidad o prueba. Por último, se despliega una capa de seguridad basada en niveles de confianza.” [21]

Algunas de las tecnologías y herramientas que definen la Web Semántica comenzaron su desarrollo mucho antes de la publicación de la misma en 2001. La especificación sobre RDF fue publicada por el consorcio W3C en 1997. RDF proporciona un lenguaje de descripción de recursos potente y simple basado en tripletes Sujeto-Predicado-Objeto y la especificación de URI. En 1999 RDF pasó a ser una recomendación. La especificación RDF se basa en los siguientes estándares:

- **Unicode**, utilizado para representar los caracteres en los alfabetos de múltiples idiomas
- **URI** define identificadores de recursos únicos
- **XML y XML-Schema**, utilizados para estructurar e intercambiar información y como formato de almacenamiento de archivos RDF (**XMLRDF-Syntax**)

Junto con RDF se desarrolló un lenguaje que proporciona herramientas para especificar ontologías: **RDFS**, y que en 2004 alcanzó el estado de recomendación por el W3C. Las ontologías representables mediante estos estándares son sencillas y básicamente organizan el conocimiento de forma jerárquica. En este mismo año también alcanza el estado de recomendación OWL (*Ontology Web Language*). OWL se puede entender como un complemento de RDF/RDFS ya que reutiliza y extiende muchas de las definiciones de dichos estándares, pero realmente es un lenguaje nuevo que aumenta esas capacidades basándose en los avances de las lógicas descriptivas. Igualmente permite representar ontologías en términos de clases y propiedades, pero incluye controles lógicos simples que garantizan la integridad de la ontología y también permite vincular diferentes ontologías entre sí mediante la importación de definiciones. Está articulado en 3 unidades diferentes (**OWL-Lite**, **OWL-DL** y **OWL-Full**) en función de la potencia expresiva requerida. La gran mayoría de las ontologías actuales están creadas o traducidas en OWL.

Con el objetivo de combinar en un solo estándar las descripciones de reglas que puedan utilizarse para inferencias lógicas no estándar: lógica de orden superior, reglas de producción, cláusulas de Horn, etc., surge SWRL (*Semantic Web Rule Language*) para OWL-DL y OWL-Lite. Está establecido como especificación por el Consorcio W3C en 2004, posteriormente y con el mismo objetivo, en 2005, se inician los trabajos sobre RIF (*Rule Interchange Format*) convertido en recomendación en 2010. SPARQL es el lenguaje de consulta de repositorios RDF establecido como recomendación oficial por el Consorcio W3C en 2008 con una sintaxis similar al SQL¹¹.

El resto de capas señaladas en el esquema carecen de desarrollo o estándar específico. La capa de criptografía que permitiría garantizar y verificar que las declaraciones de la web semántica provienen de una fuente de confianza, se puede lograr mediante el uso de una firma digital adecuada para la documentación RDF. El nivel de prueba puede derivarse de la confianza en la propia lógica de la ontología a la hora de obtener nueva información. El nivel de confianza estaría respaldado por el criptográfico y comprobaría que las órdenes provienen de una fuente confiable. La interfaz de usuario es la capa final que permite a los humanos utilizar aplicaciones de la web semántica.

En el 2006 surge la iniciativa Linked Data (Datos Enlazados) o Linked Open Data (Datos Abiertos) [22] consistente en documentos (gráficos) RDF de disponibilidad abierta y pública bajo licencias gratuitas que ha

¹¹ SQL: Structured Query Language, <https://en.wikipedia.org/wiki/SQL>

sido el gran impulsor de la Web Semántica. La cantidad de documentos RDF vinculados disponibles públicamente se ha incrementado significativamente hasta nuestros días, pese a sus principios lentos como puede observarse en la Ilustración 11. El seguimiento de la mayor parte de estos datos puede realizarse en el sitio web de Linked Open Data Cloud. Los conjuntos de datos en Linked Open Data Cloud [23] cubren una amplia variedad de temas, que incluyen geografía, gobierno, ciencias de la vida, lingüística, medios, publicaciones científicas y redes sociales, como se muestra en la Ilustración 12.



Ilustración 11. Evolución de documentos RDF en la *Linked Open Data Cloud* [24]

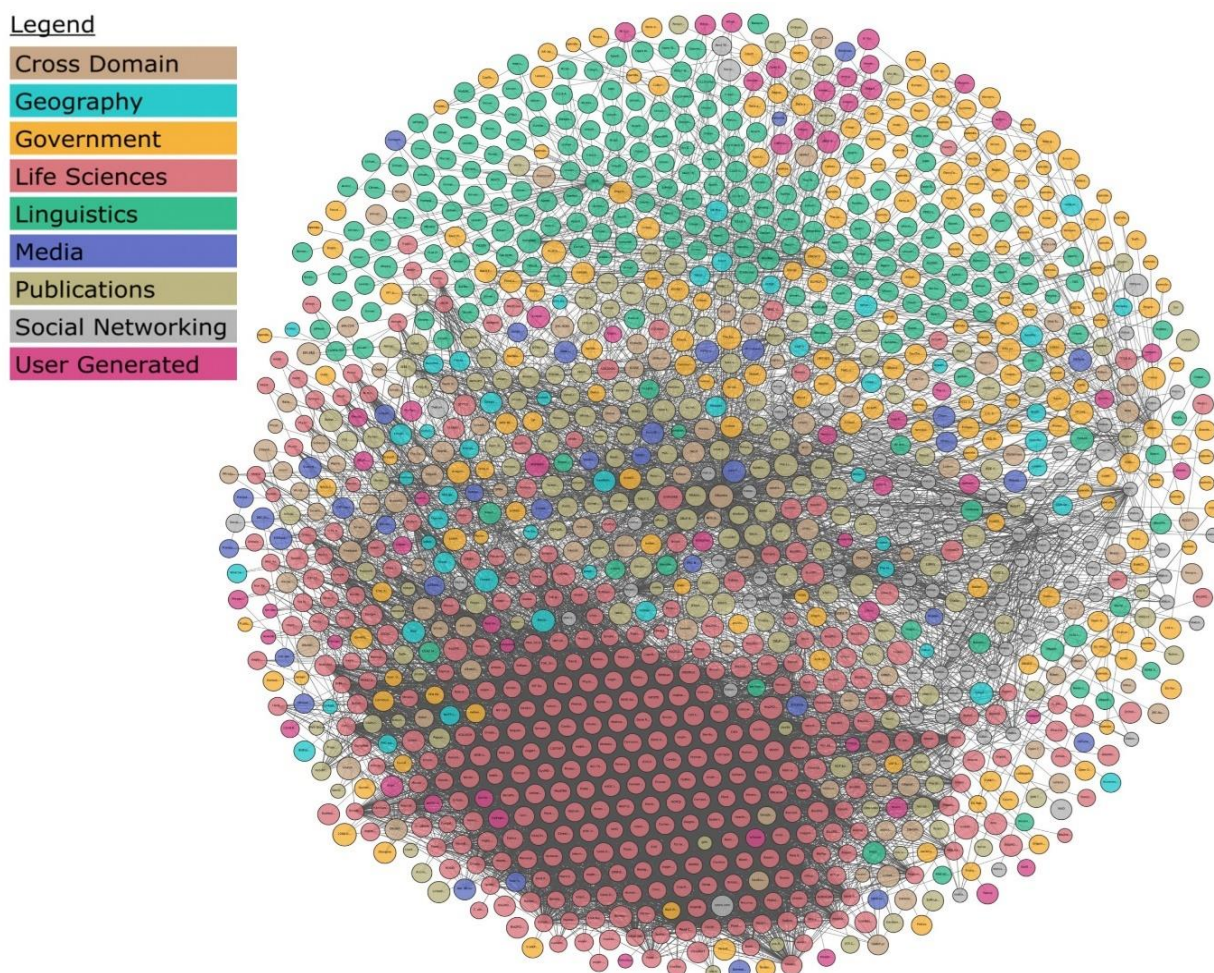


Ilustración 12. Imagen General del *Linked Open Data Cloud*, 2022 [23]

Desde sus inicios, dos de las principales tareas a las que se enfrenta la implantación de la Web Semántica son la creación de nuevas ontologías y la coordinación de las existentes. Dentro de los principales obstáculos podemos subrayar la falta de apoyo para la verificación automática sobre la autenticidad de la información,

así como la falta de agentes inteligentes que funcionen, ya que existen programas que utilizan RDF como mero formato de intercambio de datos dentro de la iniciativa *Linked Open Data*, aunque sin desarrollo ontológico.

2.4. Esquemas de Representación del Conocimiento

Dado un problema y su correspondiente dominio siempre dispondremos de un conjunto de datos básicos, no interpretados, asociados al mismo y que forman la entrada en un sistema encargado de su resolución. Este conjunto de datos es la información. La interpretación de dichos datos o su modelización conforme a una metodología y estructura se considera conocimiento. Un esquema de conocimiento es una herramienta para representar la realidad en un computador. Dentro de la Web Semántica se crean varios estándares para alcanzar la representación del conocimiento en diferentes niveles o capas, cada uno de los cuales se utiliza para modelizar o realizar consultas sobre los modelos semánticos desarrollados para cada área del conocimiento o dominio.

2.4.1. Taxonomías y Tesauros

Una correcta organización del conocimiento orientado a unos objetivos puede facilitar las tareas de mantenimiento y explotación del mismo. Las taxonomías, tesauros y ontologías son herramientas de organización sujetas a una formalización de reglas, siendo la primera menos compleja que la última.

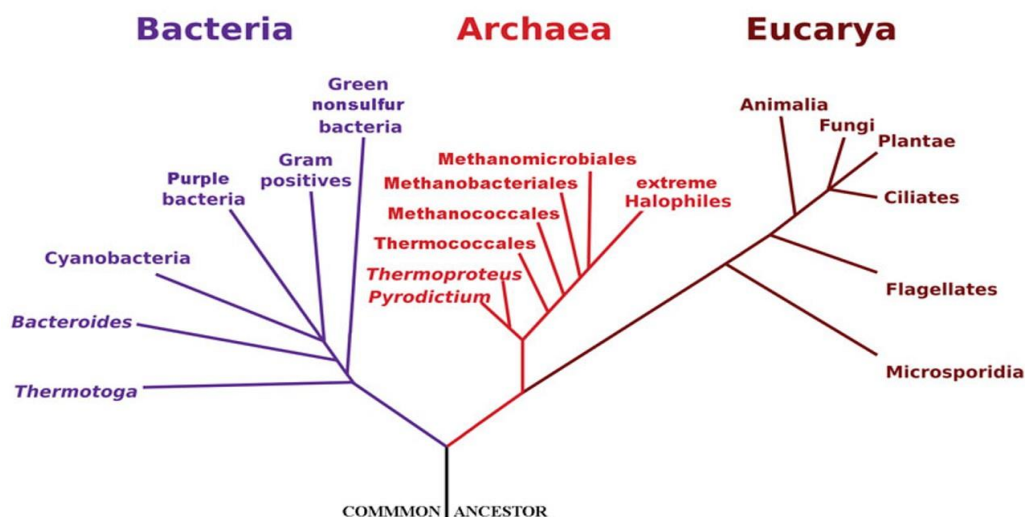


Ilustración 13. Taxonomía linneana, árbol filogenético animal [25]

Una taxonomía es una lista de términos organizados de forma jerárquica (estructura arbórea) desde los términos generales hacia los más específicos, incluyendo los relacionados, siendo quizá una de las más conocidas la clasificación animal linneana¹², uno de cuyos ejemplos se puede ver en la Ilustración 13. Un tesauro es una taxonomía a la que se le añaden las interrelaciones entre los objetos que pueden ser: de afinidad, de preferencia o jerárquicas. Las relaciones de afinidad expresan términos relacionados conceptualmente, que no están jerárquica ni preferencialmente relacionados. Las relaciones jerárquicas expresan términos más amplios o más específicos de cada concepto. Las relaciones preferenciales establecen el descriptor (término preferido) entre un grupo de sinónimos, así como los términos homónimos para diferenciar su significado, marcando el significado preferido para cada uno. La ISO 2788:1986 [26] formaliza estas relaciones de la siguiente manera:

¹²Taxonomía linneana: https://en.wikipedia.org/wiki/Linnaean_taxonomy

- Relaciones de sinonimia o preferencia: entre el término preferido (TP) o descriptor y el término no preferido (TNP).
- Relaciones jerárquicas de tipo todo-parte o clase-subclase: es decir, entre los términos más amplios (TA) y los términos más específicos (TE)
- Relaciones asociativas: entre términos relacionados (TR) de forma pragmática, es decir, no de forma jerárquica ni de sinonimia

Un tesoro se constituye como instrumento que traduce los diferentes términos utilizados en lenguaje natural por los usuarios a un lenguaje controlado y dinámico con el que se puede encontrar información confiable.

2.4.2. Ontologías

El concepto de ontología tiene sus orígenes en la filosofía: desde Platón, pasando por Kant, hasta nuestros días con Husserl, diferentes filósofos han expuesto sus puntos de vista sobre la definición del 'ser' y de cómo el hombre construye conceptos estableciendo categorías fundamentales al 'ser' de las cosas en el mundo real. En los sistemas basados en conocimiento la existencia es equivalente a que pueda ser representado, por lo que el término ontología se toma prestado para denotar dentro de un dominio determinado, una representación declarativa y consensuada del vocabulario utilizado.

Podemos decir que una ontología es una definición formal de un tesoro, que amplía y codifica el conocimiento basándose en lógica formal y estructurándola de forma que sea procesable por máquinas (computadores) y llegar a realizar inferencias de nuevo conocimiento, aunque sin duda, la definición formal más extendida para ciencias de la computación es la realizada por el ingeniero Gruber [27]:

“Una ontología es una especificación formal explícita de una conceptualización compartida. La conceptualización se refiere a un modelo abstracto de algún fenómeno en el mundo. Explícitamente quiere decir que el tipo de conceptos usados, y las constantes sobre su uso están explícitamente definidos. Formal se refiere a que la ontología debería ser leída por una máquina. Compartido refleja la noción de que una ontología captura el conocimiento consensual, que no es privado o individual, y que es aceptada por un grupo.”

Siguiendo con las explicaciones de T. Gruber, una ontología debe contener términos que formalicen conceptos, relaciones, funciones, instancias, restricciones y axiomas. Se entiende por concepto a una entidad, ideas o clases que se intenta formalizar para representar el dominio. Las interacciones entre las diferentes clases del dominio representan las relaciones. Las dos relaciones más comunes son *is-a* (es un) y *part-of* (parte de). La primera permite establecer taxonomías y la segunda permite definir entidades complejas de manera incremental. Una función es un tipo de relación concreto que obtiene un elemento de la ontología a partir de unos datos de entrada. Las instancias representan objetos específicos de una clase. Los axiomas o reglas de restricción son teoremas que se aplican sobre las relaciones que deben cumplir los diferentes elementos de la ontología, esto limita la interpretación y uso correcto de los términos y permite inferir nuevo conocimiento, que no aparece indicado explícitamente en la definición inicial. Por último, los nombres de los diferentes términos suelen venir acompañados de etiquetas con descripciones de texto para facilitar su comprensión a los usuarios.

2.4.2.1. Ingeniería ontológica

La ingeniería ontológica es un campo de las ciencias de la computación que estudia las metodologías para la construcción de ontologías utilizadas en diferentes áreas como puede ser la inteligencia artificial, la Web Semántica, gestión de conocimiento para sistemas expertos o interoperabilidad entre plataformas.

En general podemos decir que cubre la construcción del conocimiento de dominio de un problema de manera interpretable por computadoras [28].

Este campo de estudio ha hecho que aparezcan nuevos métodos y metodologías para el desarrollo de ontologías, estudios sobre los ciclos de vida de ontologías junto con herramientas y lenguajes para su creación, gestión y soporte [29]. Revisaremos los conceptos principales que fundamentan la Web Semántica y la gestión de la ontología generada por este TFG en diferentes apartados.

2.4.2.2. Clasificación

El proceso de clasificación de ontologías se puede aplicar a varios criterios para identificar los diferentes tipos de ontologías. Siguiendo la clasificación propuesta por Heijst [47] podemos clasificarlas usando dos dimensiones, primera: la cantidad y tipo de estructura de la conceptualización; y segunda: el tipo de conocimiento representado. La primera dimensión abarca 3 categorías:

- Ontologías de información:** especifican la estructura de documentos o conjuntos de datos para su integración en bases de datos.
- Ontologías terminológicas,** tipo lexicón: son descripciones de términos que representan el conocimiento en un dominio de discurso concreto, son un típico ejemplo las del campo de la medicina o la biología en la Ilustración 13.
- Ontologías de modelado del conocimiento:** superan la representación de las ontologías de la información añadiendo conceptualizaciones internas más ricas y reglas para el uso correcto de los términos. Suelen tener en cuenta el uso que se les va a dar a la hora de representar el conocimiento.

Desde la segunda dimensión, el tipo de conocimiento representado, tendríamos cuatro categorías:

- Ontologías de dominio:** su objetivo es capturar el conocimiento de un área concreta de manera reutilizable. Describen las relaciones y restricciones sobre la estructura y el vocabulario del dominio.
- Ontologías genéricas o generales:** similares a las de dominio, pero definen conceptos abstractos comunes en muchas áreas de dominio, su objetivo es establecer un marco común y uniforme de integración del conocimiento. Suelen emplearse como base común para diseñar otras ontologías al incluir abstracciones comunes a distintos dominios. Son ontologías de nivel alto, pero a menudo es difícil trazar la línea divisoria entre los términos frontera para distinguir las genéricas y las de dominio.
- Ontologías de aplicación:** solamente incluyen las definiciones necesarias para representar el conocimiento requerido para una aplicación específica. Son una mezcla entre las de dominio y las genéricas, pero no son reutilizables ya que las adaptan a un uso concreto. Algunas veces se incluyen extensiones para representar tareas y métodos de resolución.
- Ontologías de representación:** proporcionan la expresión de las primitivas necesarias para representar cualquier ontología de manera neutra, sin contar con el conocimiento a representar. Presentan gran cantidad de formalismos utilizando el paradigma KR (*Knowledge Reasoning*) o Representación del Conocimiento, es decir técnicas de razonamiento automatizado pertenecientes a las ciencias de la computación.

2.4.2.3. Características Generales

Según las definiciones dadas hasta ahora, y siguiendo las indicaciones de varios autores algunos ya vistos como Gruber [27] y Corcho [28] o Flores [30], pasamos a nombrar algunos de los elementos y características más representativas de las ontologías:

- Facilitan un vocabulario común, no ambiguo, para el dominio del problema, con la capacidad de poderse reutilizar o compartir en diferentes aplicaciones que utilicen la misma ontología.
- Contienen una taxonomía que clasifica o categoriza las diferentes entidades del dominio. La taxonomía debe separar dichas entidades de forma mutuamente excluyente, definiendo los grupos y subgrupos que aparezcan sin ambigüedades.
- Incluyen una completa generalización/especificación de las clases y subclases formalmente especificadas junto con sus relaciones e instancias, de manera que se asegure la consistencia en los posibles procesos deductivos a los que se someta la ontología.
- El vocabulario más la taxonomía generan un *Marco de Trabajo Conceptual* para la consulta, análisis o discusión de información sobre el dominio.
- Se implementan en un lenguaje para la representación ontológica (*OL*) de manera que la especificación de las clases, sus relaciones, instancias y restricciones dependerán de las características del lenguaje elegido.

2.4.3. Modelado ontológico

Se pueden emplear diferentes técnicas de modelado de conocimiento, así como implementarlas en diferentes lenguajes ontológicos [29]. El carácter de esta modelización puede ser totalmente informal expresada en lenguaje natural; semiinformal haciendo uso del lenguaje natural, pero de manera estructurada y restringida; semiformal donde ya emplearíamos un lenguaje artificial formalmente definido; y por último rigurosamente formal donde aparte de tener los términos definidos con una semántica artificial estricta, encontraremos propiedades, teoremas y reglas que la completan.

En un repaso histórico [29] [30] a principios de los años 90 del siglo pasado, las ontologías fueron construidas utilizando las técnicas de representación de conocimiento en Inteligencia Artificial del momento como los Marcos y la Lógica de Primer Orden. En la actualidad se emplean otras técnicas para la representación del conocimiento basadas en lógica descriptiva para modelarlas y se implementan utilizando distintos lenguajes desarrollados con este fin. También se han empleado para su modelado el Lenguaje de Modelado Unificado (UML) para entornos de Ingeniería de Software o los Diagramas Entidad/Relación (E/R) de Base de Datos, todos ellos capaces de representar conceptos y relaciones entre ellos. A continuación, se mostrará un breve resumen de las técnicas mencionadas:

- **Marcos (*Frames*) y lógica de primer orden (LPO).** Fue Gruber [27] quien propuso en 1993 modelizar ontologías con esta tecnología, identificando cinco clases de componentes:
 - Clases: representan los conceptos del dominio, siendo la base de la descripción del conocimiento a formalizar. Es habitual organizarlas en taxonomías pudiendo establecer mecanismos de herencia para su gestión
 - Relaciones: muestran las interacciones entre las clases del dominio. Generalmente son de tipo binario; el primer argumento se identifica como dominio y el segundo como rango
 - Funciones: son un tipo concreto de relación, identifican una asociación única entre una tupla del dominio y un elemento de rango.
 - Instancias: representaciones concretas de un objeto o clase
 - Axiomas: modelizan sentencias que han de ser ciertas. Permiten junto con los mecanismos de herencia, inferir conocimiento no explicitado en la taxonomía inicial o conocimiento nuevo. También expresan reglas que se utilizan para verificar la consistencia de la ontología

- **Lógica Descriptiva (DL).** Es un formalismo lógico de representación de conocimiento que describe los conceptos relevantes de un dominio y los relaciona para especificar propiedades. La base de construcción sintáctica son los conceptos atómicos (predicados unarios), los roles atómicos (predicados binarios) y los individuos (constantes). La descripción de los elementos los divide en dos partes:
 - Tbox: contienen la terminología describiendo propiedades generales de los conceptos, jerarquías e interrelaciones entre conceptos. Son las definiciones de conceptos y roles, lo que hemos visto como clases
 - Abox: contienen conocimiento extendido especificando los individuos y a donde pertenecen en la jerarquía. Son las definiciones de las instancias.

El conocimiento implícito sobre los conceptos e individuos puede inferirse automáticamente con la ayuda de procedimientos de inferencia lógica desarrollados para estos formalismos[31] [29], para ello se construye un sistema basado en lógica descriptiva (SBLD) compuesto por una base de conocimiento (KB) formada por los componentes Tbox y el Abox con las aserciones [30].

- **UML.** Es uno de los estándares más utilizados en el diseño de software, capaz de representar clases y jerarquías de clases, atributos y relaciones, pero que no incluye la posibilidad de añadir reglas. Dentro del diseño de software la necesidad de incluir reglas o variantes en UML se resuelve habitualmente con el lenguaje formal OCL (*Object Constraint Language*) basado en la lógica de primer orden. De esta forma los Diagramas de Clases representarían las clases, sus atributos y las relaciones, incluyendo los axiomas mediante OCL. Los Diagramas de Objetos, por último, representarían las instancias [29].
- **Diagramas E/R.** Los diagramas E/R Extendidos son de uso habitual en el modelado de ontologías. Una entidad es un objeto del mundo real distinguible de otros. Las clases se representan mediante Entidades E-R que a su vez pueden organizarse en taxonomías mediante la relación de generalización entre elementos. Los atributos de cada clase se representan mediante los Atributos-ER y mediante Relaciones-ER se conectan entre las Entidades-ER para definir las relaciones entre clases. Los axiomas se pueden representar mediante restricciones de integridad o incorporando notación complementaria como la LPO [29].

2.4.4. Lenguajes ontológicos

Son lenguajes formales utilizados para la construcción de ontologías, a menudo dependientes de la técnica de modelado elegida. Muchos de ellos surgieron por motivos completamente diferentes a las ontologías, como hemos podido observar en el caso del UML descrito en el apartado anterior, pero han acabado desarrollando puntos de coincidencia y retroalimentación que los validan para su uso.

Todos ellos, por tanto, permiten la codificación del conocimiento sobre un dominio específico incluyendo reglas de razonamiento que respaldan y validan dicho conocimiento. Los lenguajes de ontología suelen ser declarativos, casi siempre son generalizaciones de lenguajes sobre Marcos con reglas de LPO o DL [32]. En general se puede realizar una clasificación de los lenguajes [32]

- atendiendo a su sintaxis:
 - **Sintaxis tradicional:** [Common Logic](#), [Cycl](#), [DOGMA](#), [F-Logic](#), [FO-dot](#), [KIF](#) y Ontolingua, [KL-ONE](#), [LOOM](#), [OCML](#), [OKBC](#), [PLIB](#), [RACER](#)
 - **Lenguajes de marcado (habitualmente XML):** [DAML+OIL](#), [OIL](#), [OWL](#), [OWL2 RDF](#), [RDFS](#), [SHOE](#)
 - **Lenguajes naturales controlados:** [Attempto Controlled English](#)

- **Lenguajes naturales de vocabulario abierto:** Executable English

- atendiendo a la estructura lógica:
 - **Basados en Marcos:** [F-Logic](#), [OKBC](#), KM
 - **Basados en DL:** [KL-ONE](#), RACER, [OWL](#), [OWL2](#), [Gellish](#)
 - **Basados en LPO:** [Common Logic](#), [Cycl](#), [FO-dot](#), [KIF](#)

Para poder codificar una ontología deben contener una sintaxis bien definida, semántica claramente especificada y una buena expresividad que facilite la eficiencia para realizar razonamientos y la compatibilidad o traducción entre diferentes lenguajes ontológicos. Entre los lenguajes más utilizados destacan RDF, RDFS, OIL, DAM+OIL y OWL/OWL2. La W3C recomienda para su uso en la web semántica RDF/RDFS y OWL/OWL2 que se verán con detalle en el Apartado 2.5.3 y 2.5.4 respectivamente. De hecho, la utilización de OWL ha alcanzado grandes niveles de representabilidad semántica y eficiencia computacional, esto, junto con el apoyo y las especificaciones de la W3C lo han convertido en uno de los lenguajes más utilizados actualmente.

2.4.5. Razonamiento ontológico

Una de las características atribuidas a la inteligencia humana es la capacidad de razonar, entendiendo como tal a la capacidad de obtener nueva información partiendo de la ya disponible. En Inteligencia Artificial existen diversas técnicas para realizar razonamiento aplicando diversas estrategias de inferencia, como reglas o lógica, sobre el conocimiento disponible (hechos).

Dependiendo del formalismo de representación del conocimiento elegido para la ontología dispondremos de diferentes técnicas de extracción de conocimiento. Estas técnicas no solo dependerán de la naturaleza de representación de la ontología sino también del tipo de manipulación a realizar y los objetivos de conocimiento que se persigan. Las técnicas de razonamiento más utilizadas en ontologías son tres [30]:

- **Razonamiento con LPO.** La lógica de primer orden se vale de la representación de los objetos, relaciones y la teoría del cálculo de predicados para realizar inferencias sobre el dominio. Razonar consiste en determinar nuevas sentencias válidas consecuencia lógica de las ya existentes. Para ello se utiliza un conjunto de reglas que indican como, dada una hipótesis (una sentencia o un conjunto de ellas) se puede obtener otra u otras como conclusión de aplicar dicha regla o reglas a la hipótesis

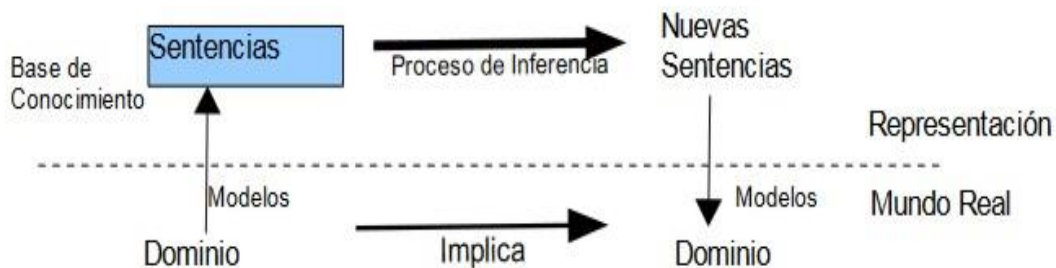


Ilustración 14. Razonamiento con LPO [30]

- **Razonamiento con DL.** Mediante un Sistema basado en lógica descriptiva (SBLD) que se compone por una base de conocimiento (KB) formada por los Tbox y Abox. La inferencia se realiza por medio de algoritmos que comprueban si las aserciones se satisfacen o no en la KB. Existen varios tipos de

razonamiento aplicables basados en la inferencia lógica, generalmente se basan en la creación de nuevos conceptos comprobando si son consistentes o contradictorios con la KB existente (Satisfactibilidad) o saber si un concepto es igual (Equivalencia), más general (Subsunción) o disjunto (Disyunción) con respecto de otro.

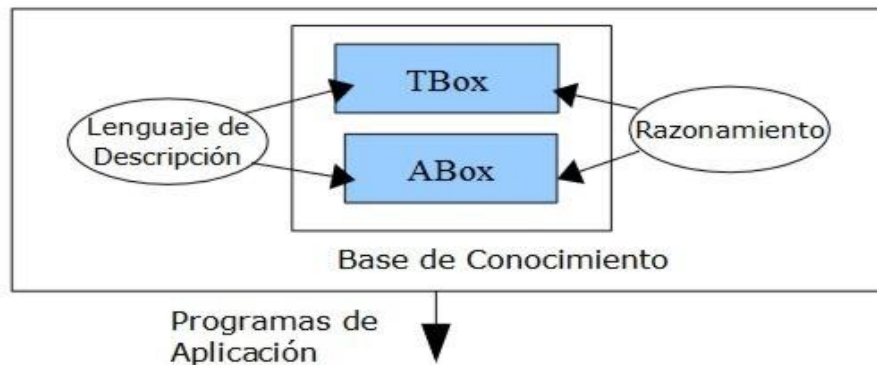


Ilustración 15. Arquitectura de un SBDL [30]

- **Razonamiento con Reglas.** Este tipo de representación de conocimiento define una arquitectura con una base de hechos, una base de reglas y un motor de inferencia. La base de hechos se modifica desde su estado inicial según se aplican los procesos de inferencia que añaden o quitan hechos, hasta sus estados finales que representan las situaciones objetivas a alcanzar. En la base de reglas se describen los elementos de deducción básicos que utilizará el sistema. El motor de inferencia actúa sobre las bases y en seleccionar alguna regla que pueda aplicarse a una situación actual, hasta que los hechos de la base de hechos satisfagan una condición de terminación o se ejecute una regla de parada.

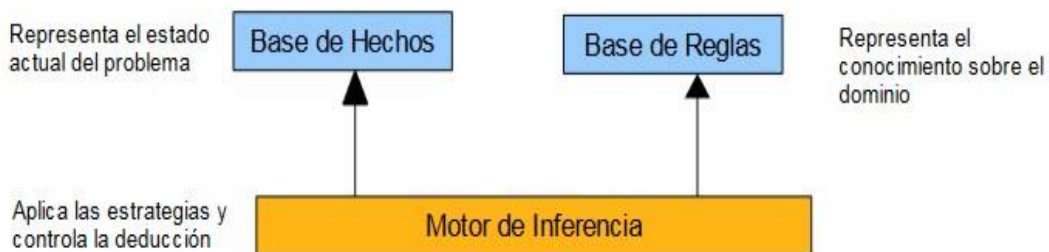


Ilustración 16. Razonamiento con Reglas [30]

La fase de selección de regla es la más compleja y comprende las etapas de: restricción, que acota el conjunto de reglas a examinar; filtrado, evalúa qué reglas de la fase anterior pueden ser utilizadas sobre la base de hechos; y resolución del conjunto conflicto, mediante una estrategia de decisión selecciona una regla de la fase de filtrado como pueden ser: orden lineal explícito, regla más específica, la más general, hechos más nuevos o más antiguos, principio de refracción, etc.

Por último, indicar que la estrategia de deducción tiene dos alternativas: encadenamiento hacia adelante o encadenamiento hacia atrás. En la primera, el motor de inferencia tiene en cuenta aquellas reglas cuyo antecedente satisfaga la base de hechos actual. En cambio, en la segunda, se trata de probar una premisa partiendo de un hecho que se debe llegar a demostrar, si coincide con la base de hechos se termina la búsqueda, si no, se siguen buscando reglas.

2.4.6. Herramientas de gestión de ontologías

Existen multitud de herramientas relacionadas con la gestión de ontologías, aunque algunas de ellas parten de desarrollos en investigación desde entornos universitarios y carecen de continuidad. Una de las

fuentes de información más actualizadas la encontramos en el VLDB Endowment¹³ [33] con conferencias anuales sobre gestión de datos a nivel internacional, artículos del IJCSIS¹⁴ [34], grupos de desarrollo universitarios como el de la Universidad de Berlín [35] o la propia Wikipedia. A continuación, pasamos a nombrar algunos de los más relevantes que mantienen su actividad:

- **Protégé** (Stanford Medical Informatics): es una herramienta de libre distribución basada en Java y equipada con una arquitectura de plugin extensibles, que permite un rápido desarrollo de aplicaciones y prototipos. De entre todas las herramientas de libre distribución disponibles para el diseño e implementación de ontologías, Protégé destaca por ser una de las más versátiles y completas, con amplio apoyo institucional y continuas actualizaciones. Su entorno gráfico y los plugin facilitan en todo momento la comprensión y visualización de los conceptos por lo que será una de las herramientas utilizadas para el desarrollo de este TFG.
- **OWL API** (University of Manchester): es una API de libre distribución desarrollada en Java que sirve como referencia de implementación para la creación, manipulación y serialización de las ontologías OWL/OWL2.
- **Apache Jena** (The Apache Software Foundation): es un *framework* de licencia abierta para la interacción y procesado de RDF desarrollado en Java.
- **OWLREADY** (Université Sorbonne Paris Nord): biblioteca de licencia libre para la programación orientada a ontologías en Python. Puede gestionar ontologías, gráficos de conocimiento para RDF/OWL. En su versión actualizada para Python3 se denomina **OWLREADY2**, esta última versión será la empleada en el desarrollo del entorno de programación de este TFG frente a la otra alternativa disponible con entorno Java (OWL API, Apache Jena). En las pruebas realizadas, la gramática de clases JAVA y el empleo de POJOS¹⁵ añadían una complejidad innecesaria al programa que con el empleo de Python desaparecía, permitiendo entender el proceso de manera más natural.
- **OntoMaven** (Freie Universität Berlin): herramienta de libre distribución basada en plugin Maven que encapsula varios repositorios de desarrollo ontológico en Java.
- **Java4C** (Freie Universität Berlin): es una implementación de referencia del estándar OMG API4KB (API para bases de conocimiento) en Java, de libre disposición.
- **Apollo** (Open University of United Kingdom Knowledge Media Institute): también bajo licencia libre, permite modelar ontologías con primitivas básicas. El modelo implementado en Java se basa en el protocolo de conectividad de base de conocimiento abierto (OKBC).
- **Virtuoso** (OpenLink Software): existe con versión libre limitada, proporciona almacenamiento y gestión para sistemas de bases de datos RDF/OWL
- **GraphDB** (Ontotext Software): existe con versión limitada libre y es otra base de datos para sistemas de bases de datos RDF/OWL

2.4.7. Metodologías para el desarrollo y diseño de ontologías

Desde el principio de las ontologías ha existido gran variedad de enfoques para su desarrollo, ya que la idoneidad de una ontología depende en gran medida de su propósito concreto y el enfoque empresarial o particular que sea necesario en ese momento, así como la estructura de la información (documentos) que se procese. El objetivo siempre ha sido la consecución de un proceso formal que involucra una planificación de actividades y tareas, especificación de requisitos, reutilización de recursos de conocimiento y conceptualización del conocimiento. Tonkin, Pfeiffer y Hewson [36] realizan una clasificación distinguiendo tres tipos de metodologías: metodologías autorreflexivas, enfoques

¹³ Very Large Data Base Endowment Inc. organization

¹⁴ International Journal of Computer Science and Information Security

¹⁵ POJOS: Plain Old Java Objects, clases simples de java independientes del framework

colaborativos y metodologías empíricas. La metodología autorreflexiva es la más sencilla: una única persona es responsable de su desarrollo y se basa en el conocimiento que ya posee, son útiles para estudios en áreas altamente especializadas con escaso intercambio de datos. Si el dominio es grande, abarca varias perspectivas, o el intercambio de datos es mayor y es necesaria la participación de varias personas, el enfoque colaborativo es el más adecuado, buscando siempre el consenso basado en el conocimiento experto de varias personas, muy útil en el ámbito empresarial. Las empíricas toman como datos de partida la documentación de interés en función del dominio elegido y la modelizan de una manera más práctica y directa.

En el artículo de Barber y Pisano [37] podemos encontrar una reseña histórica de múltiples métodos como pueden ser: Uschold y King (1995); Grüninger y Fox (1995) Metodología TOVE; Fernández-López, Gómez-Pérez y Juristo (1997) Methontology; Noy y McGuinness (2001) Simple knowledge-engineering methodology; o Stuart (2016). A continuación, procederemos a describir tres metodologías para el desarrollo y diseño, algunas son las ya citadas y otras parten de estudios más actualizados que involucran a varias de las mismas.

2.4.7.1. Stuart (2016)

La metodología propuesta por Stuart [38] destaca el carácter iterativo de la construcción, las distintas partes fruto del desarrollo deben someterse a un proceso de evaluación continua en su implementación. También hace hincapié en una buena documentación que garantice el uso y reutilización de la ontología. Stuart propone las siguientes fases, o ciclo de vida:

- **Alcance.** Se define su propósito, contenido y los usuarios destino.
- **Reutilización.** Este apartado tiene bastante importancia ya que permite la reducción de costes y la interoperabilidad con otras ontologías existentes, recomienda para ello incluir anotaciones en las propiedades de la ontología (*annotation properties*).
- **Identificación del software adecuado.** Es importante saber si todos los requerimientos necesarios son satisfechos por un único software o por varios, y evaluar la disponibilidad de dichas aplicaciones, su extensibilidad, facilidad de manejo, etc. ya que condicionarán el desarrollo de la ontología.
- **Adquisición de conocimiento.** Se produce por elicitación u obtención a través del lenguaje natural de personas involucradas en los conceptos o por medio del descubrimiento, mediante su extracción de los documentos existentes.
- **Identificación de términos importantes.** Deben estar en correspondencia con el alcance asignado a la ontología e incluir solo aquellos términos para los que exista constatación de instancias.
- **Identificación de términos adicionales, atributos y relaciones.** En consonancia con el apartado anterior y dado que la ontología no trata de crear un vocabulario sino un modelo de datos hay que asociar los atributos y relaciones a todo término de manera que se enriquezca dicho modelo.
- **Especificación de las definiciones.** Se deben revisar y precisar correctamente todos los términos obtenidos y garantizar los principios ontológicos filosóficos sobre las clases, las instancias, las jerarquías, solapamientos y dependencias. Existen desarrollos de software que garantizan la resolución a nivel lógico de estos procesos.
- **Integración con ontologías existentes.** Es la clave de la Web Semántica aprovechando la expresión de términos y propiedades de uso constatado y generalizado.
- **Implementación.** Momento en el que se integran las distintas partes del desarrollo y se codifican en el entorno de software seleccionado.
- **Evaluación.** Una vez finalizada la anterior fase se deberá hacer una batería de pruebas e informes sobre toda una serie de criterios tales como la exactitud, comprensión, completitud, claridad, eficiencia computacional, consistencia, etc.

- **Documentación.** En paralelo a la implementación y las anotaciones sobre ella, es importante complementar la información con textos narrativos.
- **Sostenibilidad.** Se comprueba que los recursos necesarios para el uso, mantenimiento, actualización y utilidad de la ontología estarán disponibles durante el tiempo de vigencia de la misma.

2.4.7.2. On-To-Knowledge

On-To-Knowledge [39] es una metodología que propone la mejora de los sistemas de conocimiento existentes en grandes organizaciones y que habitualmente está almacenada electrónicamente y de manera distribuida, sin descuidar el factor humano. Las principales fases de esta metodología son:

- **Estudio de Viabilidad** (*Feasibility study*). Se identifican los problemas y las opciones de mejora, teniendo en cuenta las problemáticas de los usuarios para determinar la viabilidad de bordar un desarrollo ontológico.
- **Arranque de la ontología** (*Kickoff*). Se crea la especificación de requisitos para la ontología, generando una descripción técnica informal de la misma.
- **Refinamiento** (*Refinement*). Se genera un prototipo a partir del refinamiento y formalización de los términos de la fase anterior.
- **Evaluación** (*Evaluation*). Se verifica si la ontología cumple con los requisitos establecidos en la especificación, tanto a nivel humano como ontológico, pudiendo establecerse un ciclo de refinamiento-evaluación hasta su logro.
- **Implantación y mantenimiento** (*Application & Evolution*). Se codifica la ontología en un lenguaje formal y una vez terminada se pone en funcionamiento sobre el entorno real, y se realiza el seguimiento de forma que si se detectan cambios se puedan realizar ajustes mediante nuevos ciclos desde la fase de refinamiento.

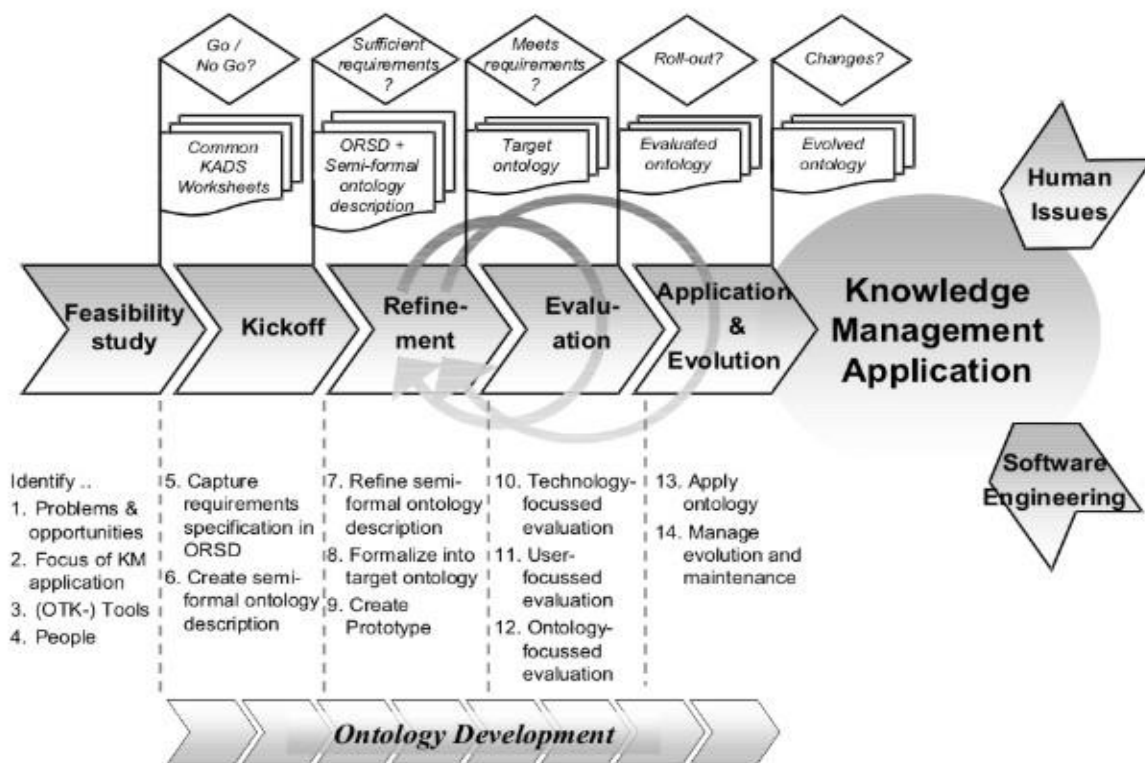


Ilustración 17. Fases en la metodología On-To-Knowledge [39]

2.4.7.3. Methontology

Methontology [40] [41] es una propuesta metodológica desarrollada en la Universidad Politécnica de Madrid que aúna en su proceso conceptos de las metodologías de Grüniger y Fox, la metodología TOVE, las propuestas de Fernández-López, Gómez-Pérez y Juristo (1997), entre otras metodologías anteriores [37]. El ciclo de vida propuesto es el siguiente:

- **Especificación.** Define el propósito y alcance para el diseño de la ontología, identificando aspectos como el nivel de formalidad, características, granularidad, escenarios.
- **Adquisición de conocimiento.** Es una fase transversal a todas las fases de la ontología, su objetivo es obtener y clarificar la mayor cantidad de conocimiento sobre el dominio. Algunos de los recursos para conseguir este objetivo son las consultas a expertos en el dominio, las consultas en fuentes bibliográficas, o la revisión de trabajos relacionados.
- **Conceptualización.** Se construye el modelo ontológico con las entidades, relaciones y propiedades en base al conocimiento del dominio adquirido.
- **Integración.** Con el objetivo de la reutilización y la integración con otros proyectos, en esta fase se buscan términos entre otras ontologías que se ajusten al dominio, evitando redundancia y fomentando compatibilidad.
- **Implementación.** Consiste en la construcción de la ontología en un lenguaje formal.
- **Evaluación.** Esta fase también tiene carácter transversal cuyo objetivo es la continua validación de la ontología que garantice su utilidad y correcto funcionamiento.
- **Documentación.** Se integra en una fase toda la documentación generada por cada fase anterior, como pueden ser las especificaciones de requisitos, la adquisición de conocimiento, el proceso de conceptualización, las ontologías relacionadas o las diferentes pruebas de validación realizadas.

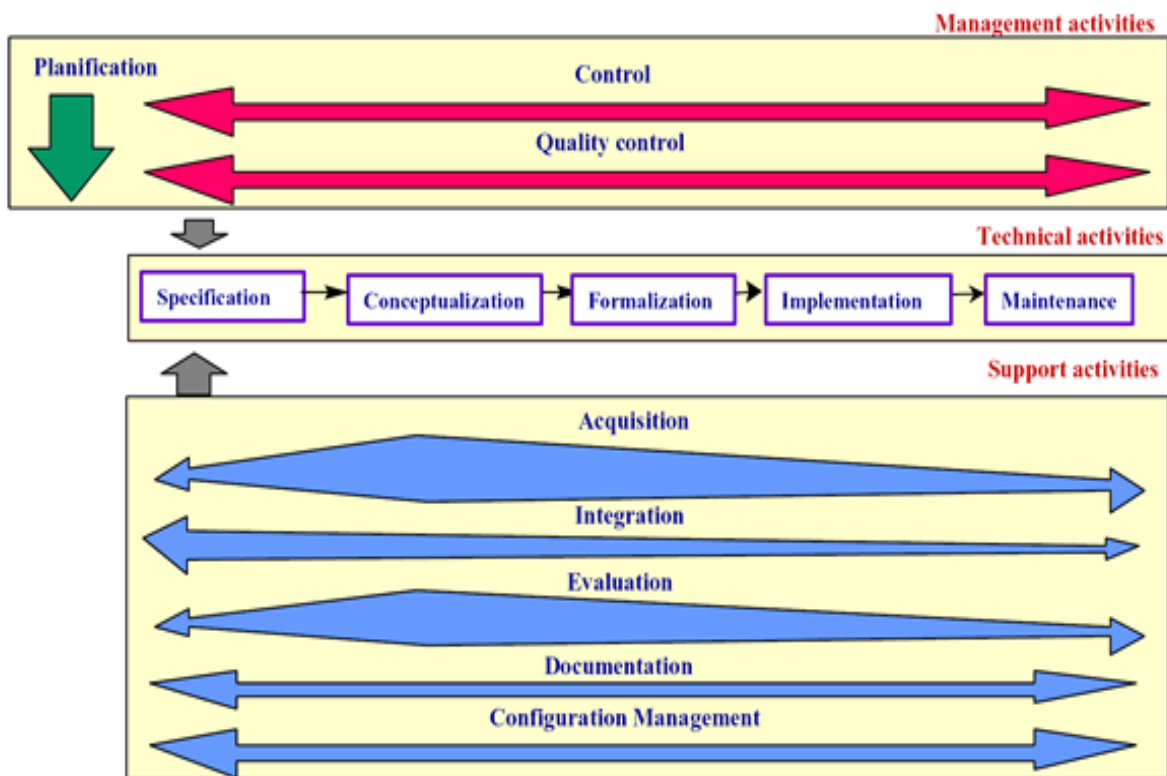


Ilustración 18. Fases en la metodología *Methontology* [40]

2.4.7.4. NeOn

La metodología NeOn [42] surge en 2009 con la necesidad de aportar un enfoque práctico a los diseños ontológicos, y diferenciarse de otros enfoques más metodológicos imperantes. Para abordar esta necesidad se propone una metodología basada en escenarios, que brinda orientación para todos los aspectos clave del proceso de ingeniería de ontologías, es decir, el desarrollo de ontologías colaborativas, la reutilización de recursos ontológicos y no ontológicos, y la evolución y mantenimiento de ontologías en red.

Los componentes principales que se definen en esta metodología son:

- Un Glosario, identifica y define las actividades y procesos a seguir para la construcción de ontologías o redes de ontologías.
- Nueve escenarios, solos o combinados permiten ajustar la construcción de ontologías o redes de ontologías en base a los requisitos iniciales que se ajusten a ellos.
- Dos modelos de ciclo de vida para la construcción de ontologías y redes de ontologías, especifican el flujo que se debe seguir en su desarrollo.
- Un conjunto de pautas metodológicas definidas para desarrollar todas las actividades y los procesos para completar el desarrollo utilizando los componentes anteriores.

Esta metodología será la empleada para la realización del TFG, al contener gran cantidad de documentación y ejemplos. Se desarrollará completamente en el Capítulo 4.

2.5. Tecnologías y lenguajes estándar asociados a la Web Semántica

Partimos del modelo propuesto por el consorcio W3C que estructura la Web Semántica en capas ya tratado en la sección 2.3. En la Ilustración 10 de la misma sección se enumeran las tecnologías más significativas del modelo de capas de la Web Semántica. A continuación, se realizará una descripción más detallada de las principales tecnologías y lenguajes nombrados.

2.5.1. URI / IRI

A finales de 2004 el W3C desarrolló una especificación denominada *Architecture of the World Wide Web* [43] desde donde se define la WWW como un espacio de información y recursos que necesita una organización a la hora de localizarlos. Para lograr este objetivo se definen las URI (*Uniform Resource Identifier*) que tratan de identificar cualquier recurso mediante una secuencia de sintaxis controlada (cadenas de texto en alfabeto inglés) que sigue una serie de *schemes* o esquemas direccionados coincidentes con los protocolos de internet. Los IRI (*Internationalized Resource Identifier*) son similares a los URI, pero incluyen caracteres de otras lenguas diferentes a la inglesa. Con esta especificación W3C unifica los sistemas de localización existentes hasta el momento: URL (*Uniform Resource Locator*) y URN (*Uniform Resource Name*).

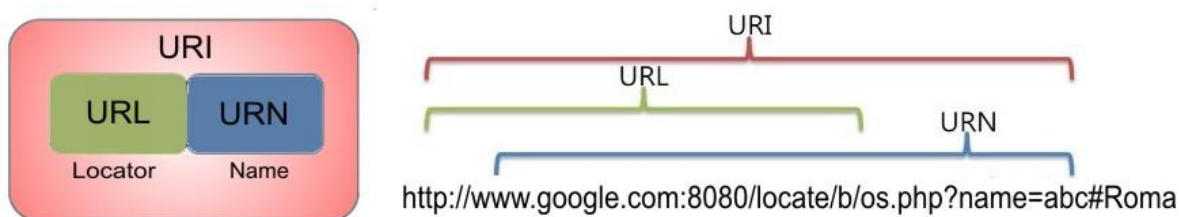


Ilustración 19. Esquema de una URI (elaboración propia)

Las URI engloban a las URL que son el sistema más utilizado para la web. Las URL se describen en la [RFC1738](#) como una cadena de caracteres ingleses, separados en bloques por barras inclinadas, que localizan por orden: el protocolo, el nombre de dominio internacional, directorio y/o subdirectorios, finalizando con el nombre del recurso a encontrar. Las URN, explicadas en la [RFC2141](#), funcionan en paralelo a las URL como sistema de categorización de recursos.

2.5.2. XML / XML Schema

XML (*Extensible Markup Language*) es un metalenguaje desarrollado bajo el W3C [44] que permite la creación de lenguajes de marcas personalizados ya que cada usuario puede definir sus propias etiquetas. Su uso habitual es el almacenamiento de datos de forma legible mediante la definición de una gramática de lenguajes específicos, al estilo de HTML, para estructurar la información. XML es una tecnología sencilla capaz de dar soporte a cualquier variedad de datos, incluyendo bases de datos, por lo que permite la intercomunicación entre diferentes sistemas facilitando su compatibilidad.

```
<?xml version="1.0" encoding="UTF-8"?>
<lista_de_compra>
  <articulo>
    <nombre>tomates</nombre>
    <precio>3.45</precio>
    <cantidad>1.0</cantidad>
    <undiades>kilogramos</undiades>
  </articulo>
  <articulo>
    <nombre>zumo de naranja</nombre>
    <precio>2.20</precio>
    <cantidad>2.0</cantidad>
    <undiades>litros</undiades>
  </articulo>
</lista_de_compra>
```

Ilustración 20. Ejemplo documento XML (elaboración propia)

Existen multitud de lenguajes basados en este metalenguaje para la representación de dominios, algunos de ellos estandarizados como pueden ser XHTML, RSS, MathML, EPUB, etc. [45], todos ellos definen sus propias etiquetas y semántica. Siguiendo la recomendación de la W3C los lenguajes de la web semántica como RDF o OWL utilizan el lenguaje de marcado XML para especificar sus formatos y serializar su sintaxis.

XML Schema, recomendación W3C [46] establece un control sobre los distintos tipos de documentos XML aplicando restricciones sobre la estructura y el contenido. Solo los documentos XML que cumplen dichas restricciones impuestas por el esquema se consideran válidos. Esto establece un control y seguridad que permite la estandarización a la hora de definir los tipos de datos estándar en un lenguaje tan versátil y libre como el XML.

2.5.3. RDF / RDFS

RDF (*Resource Description Framework*) [47] es un lenguaje estándar de propósito general utilizado para representar información e intercambio de datos en la web. Representa el significado de los datos mediante tripletas formadas por relaciones sujeto-predicado-objeto, lo que permite definir modelos de datos en forma de grafos dirigidos, donde cada tripleta contine un nodo origen (sujeto), una arista (predicado) y un nodo destino (objeto).

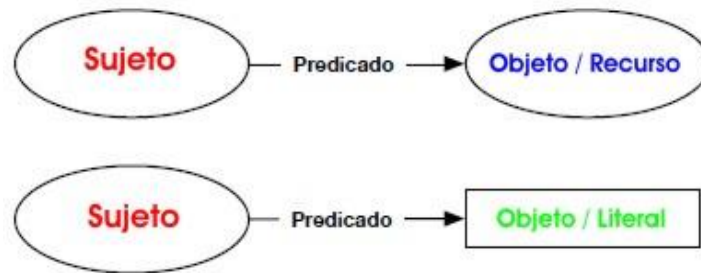


Ilustración 21. Grafo de la tripleta RDF (elaboración propia)

El modelo de datos RDF define tres elementos fundamentales que son: recursos, literales y declaraciones, considerando las propiedades como un tipo especial de recurso. Cada declaración consta de un par atributo-valor formados por propiedades clasificadas con un nombre y sus valores asociados.

- **Recursos:** son los elementos fundamentales del modelo y se identifican generalmente con un nombre utilizando para ello una URI denominada URlref.
- **Propiedades:** son un tipo específico de recurso utilizado como predicado en las declaraciones de descripción de recursos, estableciendo una relación binaria entre el sujeto y un valor. Las propiedades describen a los atributos de recursos y a las relaciones entre recursos.
- **Literales:** son cadenas de texto que pueden contener identificadores e identificadores de tipos de datos.
- **Declaraciones:** contienen un sujeto (un recurso), un predicado (una propiedad con nombre) y un objeto (un literal o un recurso con el valor de la propiedad que se asigna al sujeto).

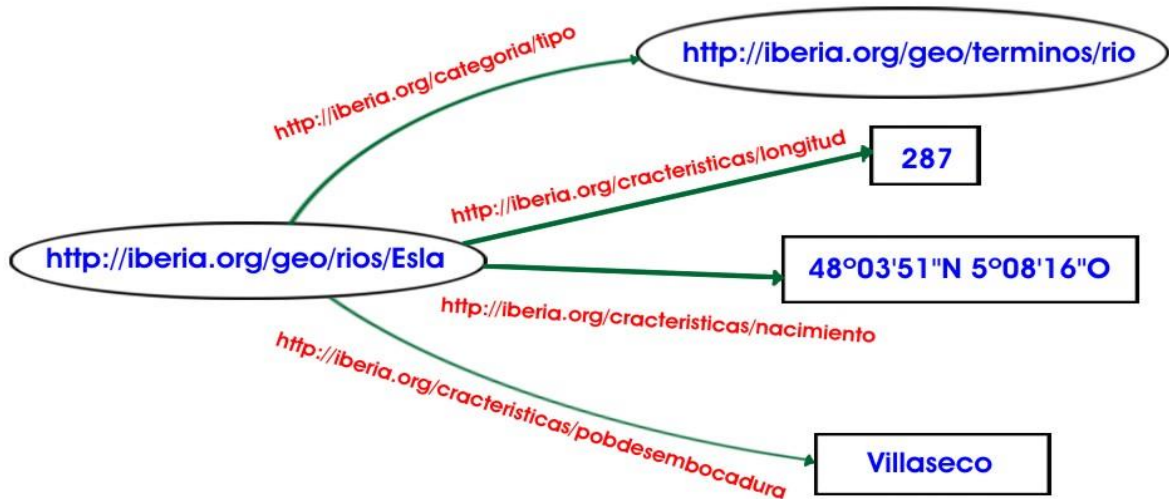


Ilustración 22. Ejemplo grafo RDF (elaboración propia)

En cada tripleta, un sujeto es un recurso y el objeto puede ser un literal u otro recurso. Cada nodo del grafo, por tanto, puede contener una URI que identifique una entidad concreta, un literal o lo que también se denomina nodo anónimo o en blanco si se desconoce el nombre que lo identifique. Las aristas se representan por medio de URIs que establecen las relaciones entre nodos o definen los pares atributo-valor.

RDF permite declaraciones sencillas, suficientes para desarrollar ontologías simples, pero carece de recursos para semánticas más complejas, es por este motivo que se incluye RDFS como estándar para el aumento de su expresividad en la descripción de ontologías.

RDF Schema (*Resource Description Framework Schema*), recomendación W3C [48], es una extensión de RDF que define primitivas para la creación de recursos. Incluye una definición de un vocabulario común para el desarrollo de ontologías de una manera estandarizada. Contiene estructuras básicas como clases y propiedades ya formalmente descritas. Con RDFS se pueden establecer jerarquía de clases, relaciones entre clases, herencia múltiple, descripción de propiedades e instancias, etc., aumentando la complejidad de las ontologías a describir. Algunas de estas definiciones son:

- **rdfs:Class**, declara recursos como clases para otros recursos. Tiene carácter recursivo.
- **rdfs:Resource**, la clase a la que pertenecen todos los recursos.
- **rdfs:Literal**, la clase de todos los valores literales, cadenas y números.
- **rdfs:Datatype**, abarca los tipos de datos definidos en el modelo RDF.
- **rdf:Property**, abarca las propiedades.
- **rdfs:subClassOf**, instancia de rdfs:Class, para definir jerarquías de clase con sus superclases.
- **rdfs:subPropertyOf**, instancia de rdf:Property para definir jerarquías de propiedades.
- **rdfs:domain**, instancia de rdf:Property que especifica el dominio de una propiedad.
- **rdfs:range**, instancia de rdf:Property que especifica el rango de una propiedad.
- **rdfs:ConstraintResource**, la clase que agrupa a todas las restricciones.
- **rdfs:seeAlso**, instancia de rdf:Property para relacionar un recurso con otro que proporciona información adicional sobre el primero.
- **rdfs:label**, instancia de rdf:Property para proporcionar una versión adicional sobre el nombre de un recurso.
- **rdfs:comment**, instancia de rdf:Property usada para proporcionar una descripción de un recurso

```

"El libro Dune en mi biblioteca, su autor es Frank Herbert y fue publicado en 1965"

Sujeto (recurso): "https://www.mibiblioteca/libro/Dune"
Predicado: mb:publicacion
Objeto (literal, número): 1965
Predicado(recurso): mb:autor
Objeto (literal, cadena de texto): "Frank Hervert"

<?xml version="1.0"?>

<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:mb="https://www.mibiblioteca.com/">

  <rdf:Description rdf:about="https://www.mibiblioteca.com/libro/Dunne">
    <mb:publicacion>1965</mb:publicacion>
    <mb:autor>Frank Herbert</mb:autor>
  </rdf:Description>

</rdf:RDF>

```

Ilustración 23. Serialización XML de un recurso en RDFS (elaboración propia)

La Ilustración 23 muestra una serialización de RDF/RDFS en XML. La primera línea del documento RDF es la declaración XML. La declaración XML continúa con el elemento raíz por los documentos RDF: **<rdf:RDF>**. El espacio de nombres **xmlns:rdf** especifica todos los elementos con prefijo “rdf” son del espacio de nombres **http://www.w3.org/1999/02/22-rdf-syntax-ns#** del estándar RDF. Lo mismo ocurre con **xmlns:mb** especifica que los elementos con el prefijo “mb” pero en este caso son del espacio de nombres **http://www.mibiblioteca.com/** definido para la ontología propia desarrollada “mibiblioteca.com”.

El elemento **<rdf:Description>** contiene la descripción del recurso identificado por el atributo **rdf:about**. Los elementos: **<mb:publicacion>** y **<mb:autor>** son propiedades del recurso propio definido para la ontología.

2.5.4. OWL / OWL2

OWL (*Ontology Web Language*), es la recomendación de la W3C [49] para la Web Semántica y uno de los lenguajes de ontologías más extendidos. OWL proporciona y extiende las definiciones de RDF/RDFS, aumentando los recursos para la descripción de ontologías, como pueden ser el aumento del grado en la descripción de clases usando combinaciones lógicas de inserción, unión y complemento; propiedades algebraicas en las funciones como la transitividad y reflexibilidad, o restricciones de cardinalidad sobre propiedades. Para garantizar la compatibilidad de los documentos dentro de la web semántica, admite el uso de definiciones con equivalente en RDF/RDFS en ambos formatos: RDFS o OWL, motivo por el cual a veces se considera a OWL erróneamente como un complemento de RDF/RDFS. De esta manera, un documento RDF/RDFS siempre se podrá interpretar en OWL pero lo contrario no siempre será posible.

En resumen, OWL incorpora nuevos recursos de construcción a los ya existentes en RDFS, que permiten definir ontologías incluyendo lógica descriptiva en el diseño de sus clases, relaciones y propiedades. Al poder describir ontologías con un vocabulario más complejo, aumenta también la capacidad para inferir información sobre las mismas sin necesitar recursos externos dedicados al procesamiento de reglas y lógica, como ocurre en otros lenguajes de ontologías.

En base a la complejidad de la ontología y a las necesidades de desarrollo, presenta tres sublenguajes que se van englobando al aumentar su nivel de libertad:

- **OWL Lite:** utilizada para representar ontologías muy sencillas como pueden ser clasificaciones jerárquicas y restricciones simples. Restringe el uso de la cardinalidad a 0 o 1. Es una buena opción para tesauros y taxonomías.
- **OWL DL:** para ontologías complejas, buscando un compromiso entre capacidad de representación y prestaciones computacionales, con tiempos de ejecución finitos. Permite el uso de todos los constructores disponibles, pero se basa en las lógicas descriptivas (DL) [50] con una semántica bien definida y propiedades formales correctamente estructuradas que garantizan la finitud de los cálculos.
- **OWL Full:** el mayor grado de libertad para la definición de ontologías, a cambio, no existen garantías computacionales sobre la finitud de los cálculos al no garantizarse un mantenimiento de la lógica necesaria para un razonamiento automatizado. Los razonadores de esta versión tienden a ser correctos pero incompletos.

OWL2 [51] es la versión actual de OWL e incorpora entre otras mejoras: cadenas de propiedad, claves, nuevos rangos de datos, tipos de datos enriquecidos, nuevas restricciones de cardinalidad, propiedades asimétricas, y disjuntas, anotaciones mejoradas, axiomas y especificaciones. Al igual que ocurre con OWL admite muchas sintaxis como Sintaxis Manchester [52] o Turtle [53], aunque la aceptada por todas las herramientas es la serialización con formato RDF/XML [54].

OWL2 también presenta tres perfiles que se unen a los ya existentes en OWL. La representabilidad de los nuevos perfiles con respecto a los anteriores se muestra en la Ilustración 24 . Los perfiles son:

- **OWL2 EL:** todas las tareas de razonamiento estándar dentro de la ontología tendrán un tiempo de ejecución polinomial. Pensado para aplicaciones que trabajen con grandes ontologías, y donde es posible intercambiar la expresividad del lenguaje para un mayor rendimiento en los algoritmos.

- **OWL2 QL:** utilizado para definir ontologías que requieren compatibilidad en el acceso a sus datos mediante el uso de lenguajes relacionales como SQL, garantizan un tiempo de ejecución logarítmico.
- **OWL2 RL:** empleado cuando se utilizan tecnologías de desarrollo de ontologías de bases de datos con reglas extendidas, operando directamente sobre los datos en tripletas. Garantizan un tiempo de ejecución polinomial.

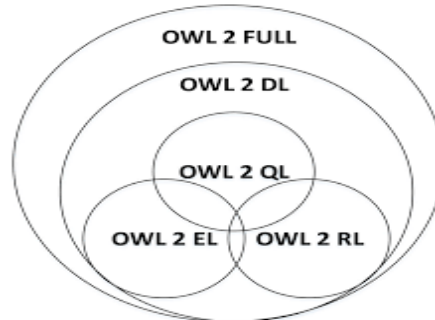


Ilustración 24. Grado de representabilidad de los perfiles de OWL2 [55]

2.5.4.1. Esquema general y validadores

Una ontología OWL / OWL2 se debe implementar y serializar en un único documento, o en documentos separados, pudiendo estar referenciados mediante URIs. Cada documento construye las ontologías utilizando la sintaxis y constructores OWL, además de los requisitos propios del formato como hemos podido ver en ejemplos anteriores sobre serialización en XML. A su vez también se permite, para las redes de ontologías, su serialización conjunta o separada. Las ontologías, a su vez pueden residir en un servidor o servidores de la organización o ser distribuidas por la Web utilizando diferentes localizaciones para los archivos, esto facilita la reutilización y extensión de las ontologías existentes. En general se recomienda la separación de los archivos que contienen los constructores de la ontología (modelo) de los que contienen las instancias (datos), y referenciarlos por URI diferentes.

ARCHIVO

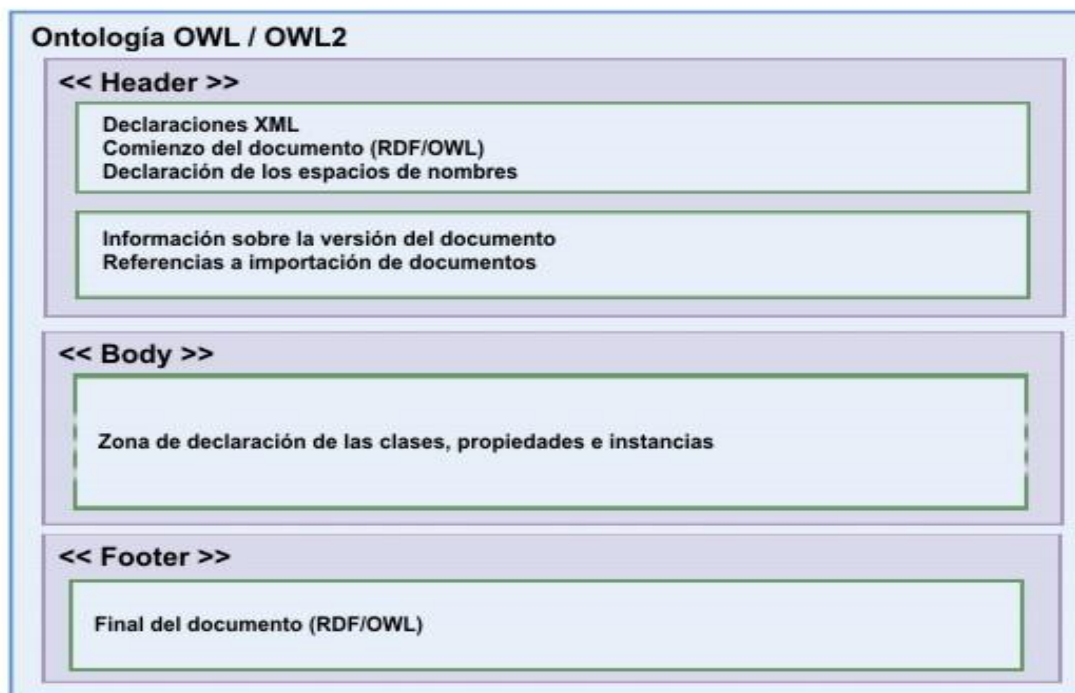


Ilustración 25. Estructura básica de un archivo OWL / OWL2 (elaboración propia)

Un archivo de una ontología OWL/OWL2 contiene tres apartados o zonas diferenciadas que constituyen los elementos necesarios para su creación. Estos apartados son el “**header**” o cabecera, “**body**” o cuerpo y “**footer**” o pie (final) como se puede apreciar en la Ilustración 25.

Es importante indicar que existen servicios de validación para los distintos tipos de OWL/OWL2 que también permiten mostrar las estructuras utilizadas o su forma abstracta, así como validar la estructura y consistencia de la ontología. Un ejemplo lo tenemos en la herramienta [OOPS!](#) (Ontology Pitfall Scanner!) un desarrollo del Ontological Engineering Group de la Universidad Politécnica de Madrid [56].

2.5.4.2. Elementos de la cabecera

Como figura en la Ilustración 25, y al igual que ocurría en el ejemplo de serialización de la Ilustración 23 del Apartado 2.5.3, en este apartado se inicializa la etiqueta RDF o OWL dependiendo del formato en XML, se declaran los espacios de nombres (*namespaces*) y elementos de la ontología como versiones e importaciones. En las siguientes tablas se comentan algunos de los más importantes:

Elementos del Namespace	
ELEMENTO	DESCRIPCIÓN
xmlns:xsd= “http://www.w3.org/2001/XMLSchema#”	especifica los espacios de nombres para XML Schema
xmlns:rdfs= “http://www.w3.org/2000/01/rdf-schema#”	identifica los espacios de nombres para RDFS
xmlns:rdf= “http://www.w3.org/1999/02/22-rdf-syntax-ns#”	identifica los espacios de nombres para RDF
xmlns:owl= “http://www.w3.org/2002/07/owl#”	identifica los espacios de nombres para OWL
xmlns:foaf= “http://xmlns.com/foaf/0.1/”	identifica los espacios de nombres para la ontología importada, en este caso Foaf ¹⁶
xmlns= “http://www.mibiblioteca.com/elementos#”	identifica los espacios de nombres para la ontología desarrollada

Tabla 1. *Namespaces* en cabeceras OWL / OWL2

Elementos de Información y Referencia	
ELEMENTO	DESCRIPCIÓN
owl:Ontology	se utiliza para especificar que se instancia una ontología OWL
owl:versionInfo	cadena de texto con información sobre la versión de la ontología
owl:priorVersion	indica mediante una URI la versión anterior de la ontología
owl:incompatibleWith	informa sobre la incompatibilidad con otras versiones de la ontología actual
owl:DeprecatedClass	Es un comentario sin validez, pero marca una determinada clase en obsolescencia “próxima”
owl:deprecatedProperty	Es un comentario sin validez, pero marca una determinada propiedad en obsolescencia “próxima”
owl:imports	permite referenciar a ontologías externas para su reutilización en la ontología actual

Tabla 2. Elementos de información en cabeceras OWL/OWL2

2.5.4.3. Características generales RDF Schema

Presentamos algunos de los principales elementos generales de OWL/OWL2, como se puede apreciar por el prefijo, muchos son recursos ya definidos en RDFS.

¹⁶ Ontología FOAF: Friend of a Friend, <http://xmlns.com/foaf/spec/>

- **owl:Class** y **rdfs:subClassOf**: OWL/OWL2 define una clase general llamada “**Thing**” como superclase de todas las instancias y clases de la ontología, a partir de esta superclase las clases se definen con este elemento. Las clases se pueden organizar utilizando jerarquías (especialización) con el elemento **rdfs:subClassOf**.
- **rdf:Property** y **rdfs:SubPropertyOf** : estos elementos representan la relación entre instancias o entre instancias y valores. También permiten crear jerarquías de propiedades. Dentro de estas propiedades nos encontramos con **owl:ObjectProperty** que define las relaciones entre instancias, y por otro lado la propiedad **owl:DatatypeProperty** que se utiliza en las relaciones con un valor numérico o cadena.

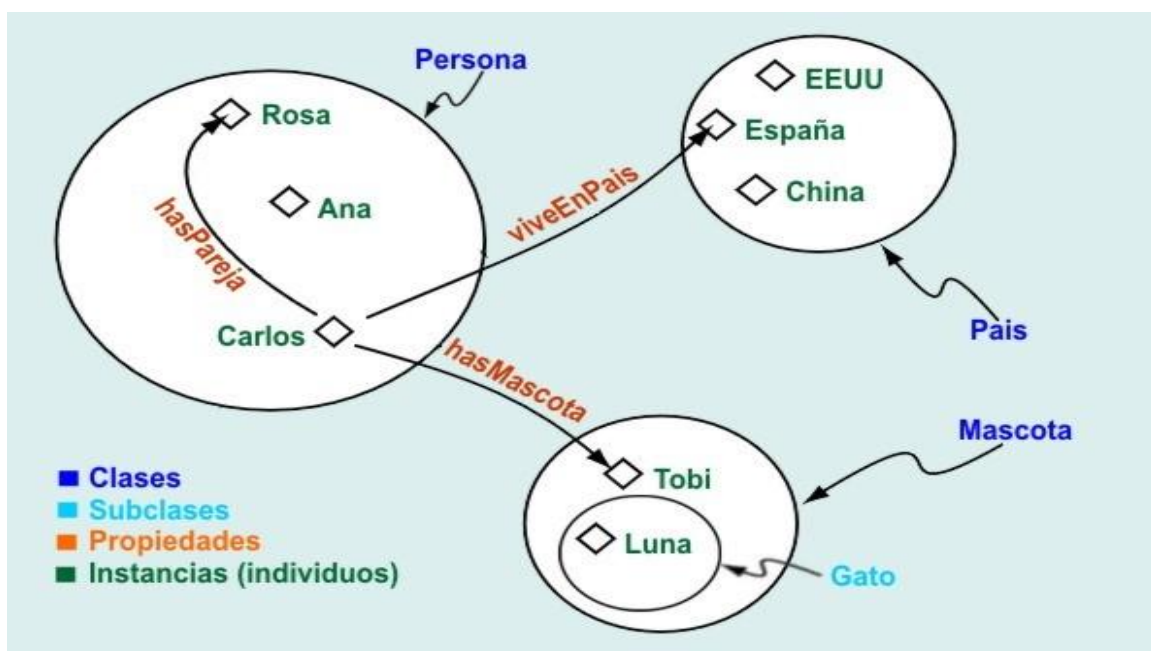


Ilustración 26. Representación de elementos básicos en OWL/OWL2 (elaboración propia)

- **rdfs:domain**: propiedad que limita las instancias sobre las que puede ser aplicada una relación. Cuando se emplea en OWL/OWL2 las relaciones (ObjectProperty, DatatypeProperty), tienen carácter de navegabilidad desde el dominio hacia el rango.
- **rdfs:range**: propiedad que limita las instancias que puede tener la propiedad como valor, sobre las que puede ser aplicada una relación. Cuando se emplea en OWL/OWL2 las relaciones (owl:ObjectProperty y owl:DatatypeProperty), tienen carácter de navegabilidad desde el dominio hacia el rango.

2.5.4.4. Características sobre la igualdad o desigualdad

Se listan algunos de los principales elementos generales de OWL/OWL2, usados sobre todo para OWL Lite.

- **owl:equivalentClass**: define dos clases como clases equivalentes, por lo que las instancias de una clase también lo son de la otra, es útil cuando se unen ontologías similares. No hay que confundirla con owl:sameAs.
- **owl:equivalentProperty**: lo mismo que la anterior, pero para propiedades.
- **owl:sameAs**: indica que dos instancias son exactamente iguales.
- **owl:allDifferent**: identifica como distintas entre sí a la lista de instancias facilitada.
- **owl:differentFrom**: establece que una instancia es distinta a otras proporcionadas. Teniendo en cuenta que RDF y OWL no asumen el principio de identificador único (UID), el uso de esta propiedad garantiza no llegar a contradicciones en los procesos de inferencia.

2.5.4.5. Características de las propiedades y valores

Se listan algunos identificadores especiales de OWL/OWL2 para facilitar información sobre las propiedades y sus valores. Es importante recordar que las relaciones de navegabilidad van desde el dominio hasta el rango.

- **owl:inverseOf**: especifica que una propiedad es inversa respecto de otra propiedad ya expresada.
- **owl:FunctionalProperty**: con esta propiedad indicamos que solo puede haber un único valor o instancia para un objeto (dominio), indirectamente establecemos su cardinalidad máxima en 1.
- **owl:InverseFunctionalProperty**: indica que la relación tiene la característica opuesta a la anterior. Es decir, que el dominio de la relación es funcional, o que solo puede haber un valor en el dominio para la relación.
- **owl:SymmetricProperty**: establece una relación entre propiedad e instancia de tipo conmutativo.
- **owl:TransitiveProperty**: establece una relación transitiva entre pares de relación donde el último miembro de la relación es el primero de la siguiente, por tanto, se establece indirectamente un nuevo par con el primero de la primera y el último de la segunda.

2.5.4.6. Restricciones en propiedades

Se muestran algunas de las restricciones sobre instancias que se pueden utilizar en una relación, estas restricciones solo pueden usarse dentro del nodo **owl:Restriction**.

- **owl:allValuesFrom**: se emplea sobre las propiedades (relaciones) con respecto a una clase. Define una restricción para la clase sobre el rango en la relación. A su vez, si una instancia de la clase está relacionada con otra instancia por esta propiedad, se infiere que al rango de esta relación se le aplica la misma restricción.
- **owl:someValuesFrom**: se utiliza en la restricción de una clase para enlazar con otra clase o valor como rango. Describe la clase en la que todas sus instancias tienen al menos un valor para la relación y el rango indicados.

2.5.4.7. Restricciones de cardinalidad

También se muestran algunas sobre la cardinalidad en las propiedades, hay que tener en cuenta que en OWL Lite la cardinalidad está restringida a 0 o 1, mientras que para OWL DL y OWL Full, no.

- **owl:minCardinality**: establece la cardinalidad mínima de una clase para una relación. Esta propiedad describe a una clase en la que todas sus instancias tienen al menos N valores semánticamente distintos para la propiedad de la relación, siendo N la cardinalidad mínima. El valor se especifica mediante *XML Schema datatype* [57] con valores no negativos.
- **owl:maxCardinality**: similar a la anterior, pero para expresar los máximos valores permitidos.
- **owl:Cardinality**: establece la cardinalidad de la relación a un valor concreto no negativo, útil por ejemplo para modelar relaciones como 'tieneFechaNacimiento' en una clase 'Persona'.

2.5.4.8. Algunos elementos adicionales de OWL DL y Full

Por último, se muestran algunos elementos utilizados para OWL DL y OWL Full que extienden algunos de los elementos anteriores empleados sobre todo en OWL Lite.

- **owl:oneOf**: describe las clases como una lista de instancias. La clase tendrá exactamente esas instancias, de forma invariable.

- **owl:hasValue**: define propiedades que tienen una instancia como valor.
- **owl:disjointWith**: especifica que algunas clases sean disjuntas entre sí, o lo que es lo mismo, que no tienen instancias en común.
- **owl:unionOf**, **owl:complementOf**, **owl:intersectionOf** : se liberan de las restricciones de Lite, permitiendo arbitrariedad de valores booleanos en sus descripciones.

2.5.4.9. Razonamiento básico en OWL

Como ya se avanzaba en la introducción del apartado, OWL incorpora aspectos de las lógicas descriptivas en la definición de sus elementos, esto permite efectuar inferencias básicas directamente sobre los elementos declarados (ABox y TBox), sin la necesidad de incluir reglas y herramientas externas (razonadores) ver Apartado 2.4.5. Estos elementos de razonamiento son el chequeo de consistencia y la subsunción.

Mediante un chequeo de consistencia continuo de los axiomas que se van añadiendo, aseguramos la consistencia del sistema y por tanto podremos obtener consecuencias lógicas sobre el mismo. Mediante la subsunción podemos determinar si una clase es subclase de otra (bien sea por definición directa o como resultado de aplicar los axiomas correspondientes). La subsunción también permite generar una jerarquía de clases inferida calculando todas las relaciones de subsunción entre las clases. Este proceso se denomina clasificación, y es de gran importancia, ya que en base a la clasificación podemos ampliar el conjunto de inferencias básicas con:

- Reconocimiento de instancias: se chequea si un individuo pertenece a una clase bajo todas las interpretaciones
- Recuperación: se recuperar todos los individuos de una clase
- Equivalencia: dos clases son equivalentes si se subsumen entre ellas
- Realización: se localizan las clases más específicas a las que pertenece un individuo
- Consultas conjuntivas booleanas: se trata de la capacidad de responder a preguntas lanzadas sobre la clasificación, comprobando si la respuesta a los términos de la consulta son consecuencia lógica del mismo (contiene la clase vacía) o no lo son (no contiene ninguna clase)

2.5.5. SPARQL

SPARQL (*SPARQL Protocol and RDF Query Language*) es un protocolo normalizado por la W3C [58] para la consulta de grafos RDF y OWL desde sus inicios en 2008. Se puede entender su utilización y sintaxis como una analogía con el lenguaje de consulta de datos relacionales SQL, pero adaptado a los datos en tripletas.

El lenguaje SPARQL incluye las URI de RDF, pero sin incorporar los espacios. Las referencias son siempre absolutas, aunque pueden definirse prefijos para simplificar las consultas. La mayoría de las formas de consulta en SPARQL se adaptan al patrón básico RDF/OWL de tripleta, salvo que cada sujeto, predicado u objeto puede ser una variable, identificada porque comienza con un carácter "?". Un patrón de grafo básico concuerda con un subgrafo de datos RDF/OWL almacenado cuando los términos de dicho subgrafo pueden ser sustituidos por las variables y el resultado es un grafo RDF equivalente al subgrafo en cuestión. Los estándares de tripletas de la cláusula SELECT (similar a SQL) tienen la misma forma que las tripletas normales, excepto que cualquiera de las tres partes de la tripleta puede ser substituida por una variable. La cláusula SELECT devuelve una tabla de variables con los valores que satisfacen la consulta. Un ejemplo sencillo de consulta con una variable lo podemos ver en la Ilustración 27.

```

Datos:
<http://www.mibiblioteca.com/libro/libro258> <http://purl.org/dc/elements/1.1/title> "Dune" .

Consulta:
SELECT ?title
WHERE
{
  <http://www.mibibliote.com/libro/libro258> <http://purl.org/dc/elements/1.1/title> ?title .
}

Resultado:
title
"Dune"

```

Ilustración 27. Consulta simple en SPARQL (elaboración propia)

SPARQL admite una serie de filtros y operadores que permiten hacer consultas complejas al conjunto de tripletas almacenadas mediante la cláusula FILTER u OPTIONAL en el WHERE, estableciendo de nuevo similitudes con SQL. De la misma forma encontramos cláusulas UNION, FROM, ORDER BY, etc., incluir nuevas tripletas en la base de datos con INSERT o eliminar con DROP y DELETE, podemos encontrar un interesante tutorial en [59].

Debido al auge del *Open Data* son múltiples los organismos públicos y privados que ofrecen puntos de acceso por vía Web y que aceptan el protocolo SPARQL de almacenamiento y como motor de consulta. Esos puntos de acceso reciben el nombre de **SPARQL endpoints**. Un **SPARQL endpoint** acepta consultas y devuelve los resultados vía HTTP, podemos encontrarlos de tipo genérico, para ejecutar consultas sobre cualquier dato RDF accesible por vía Web (especificado como parámetro) o de ámbito específico, limitados a las consultas sobre el conjunto de datos particulares determinados por la organización.

2.5.6. SWRL

SWRL (Semantic Web Rule Language) protocolo de la W3C [60] propuesto para la Web Semántica cuyo objetivo es poder expresar lógica y reglas combinando los sublenguajes de OWL (OWL DL y OWL Lite) con desarrollos previos como los del organismo RuleML¹⁷. Básicamente permite expresar reglas en términos de conceptos OWL y de esta forma potenciar sus capacidades deductivas. Incluye una sintaxis abstracta con reglas, limitándolas a cláusulas definidas (Horn¹⁸) y sin funciones, pero con capacidad de combinarlas con la base de conocimientos OWL/OWL2. El empleo de SWRL puede afectar a la decidibilidad de la ontología, por lo que es importante restringir la forma de las reglas admisibles imponiendo condiciones de seguridad.

Las reglas tienen forma de implicación con un antecedente (cuerpo) y un consecuente (cabeza). El significado asumido es el habitual: siempre que se cumplan las condiciones especificadas en el antecedente, entonces también deben cumplirse las condiciones especificadas en el consecuente. En la Ilustración 28 se puede ver un ejemplo de implementación de una regla y su serialización en XML extraído y adaptado de wikipedia.org. Es importante mencionar que existen plugin en Protégé que permiten la visualización y edición de reglas en el mismo formato, siendo de utilidad para futuros desarrollos sobre el proyecto y motivo añadido a su elección.

¹⁷ RuleML: https://wiki.ruleml.org/index.php/RuleML_Home

¹⁸ Cláusulas de Horn: https://en.wikipedia.org/wiki/Horn_clause

Sintaxis humana

$\text{hasParent}(?x1,?x2) \wedge \text{hasBrother}(?x2,?x3) \Rightarrow \text{hasUncle}(?x1,?x3)$

Sintaxis XML

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasBrother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom swrlx:property="hasUncle">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

Ilustración 28. Ejemplo de regla en SWRL [61]

Capítulo 3

3. Contexto específico

Este capítulo se centra en la exposición de la información recopilada y analizada durante la etapa de estudio y documentación, sobre los objetivos del proyecto. Se presentan los Sistemas de Climatización, la ontología general respaldada por BIM para los Edificios Inteligentes y el estándar ontológico europeo SAREF, requisito de este proyecto.

3.1. Sistemas de Climatización

Los sistemas de calefacción, ventilación y aire acondicionado (HVAC¹⁹) sirven para mantener las condiciones ambientales deseadas en un espacio. Está formado por varios subsistemas combinados [62]: por un lado, está la calefacción que mejora la calidad del aire de un interior en forma de calor. Para conseguir modificar la temperatura del aire se pueden utilizar dispositivos como resistencias eléctricas, bombas de calor o intercambiadores.

Otro elemento es la ventilación, cuya finalidad es la renovación del aire interior de un recinto y garantiza su calidad (eliminando partículas de contaminación, patógenos, etc.) mediante unidades de tratamiento. Este apartado es muy importante en las edificaciones modernas, más herméticas y aisladas que impiden la renovación natural del aire. Dispositivos como ventiladores y filtros realizan esta tarea.

Por último, tendríamos el aire acondicionado, que no hay que confundir con la ventilación, ya que se encarga de la refrigeración de un espacio con independencia del suministro y la calidad del aire. Los dispositivos más utilizados para bajar la temperatura del aire son compresores, ventilación de doble flujo o enfriamiento por evaporación.

El diseño y la gestión de un sistema HVAC tienen una fuerte dependencia de las características físicas del edificio y de los dispositivos elegidos para implementar sus funciones. Además, el control de un sistema en uso está sometido a constantes cambios: diferentes necesidades personales de confort para un mismo recinto y la gente que lo ocupa, diferentes actividades realizadas (por ejemplo, un auditorio en una conferencia o en una clase de danza), la constante entrada y salida de individuos, el cambio en las condiciones climáticas del exterior, etc.

Un equipo HVAC puede suponer alrededor del 40% del consumo del total de energía en una edificación según estudios generalizados en múltiples países industrializados [63] [64] [65] [66] [67]. El diseño y gestión de los sistemas HVAC, orientado a la eficiencia energética, está gestionado por estándares regulados por la unión de fabricantes siendo uno de los más importantes el ANSI/ASHRAE Standard 55 [68] o *European*

¹⁹ AHU: *Heating, Ventilation and Air Conditioning*, https://en.wikipedia.org/wiki/Heating,_ventilation,_and_air_conditioning

Ecodesign Regulation 1235/2014 [69]. Igualmente existen iniciativas gubernamentales de gestión medioambiental que incluyen estos equipos como la desarrollada por la Unión Europea en la Directiva (UE) 2018/844 [70].

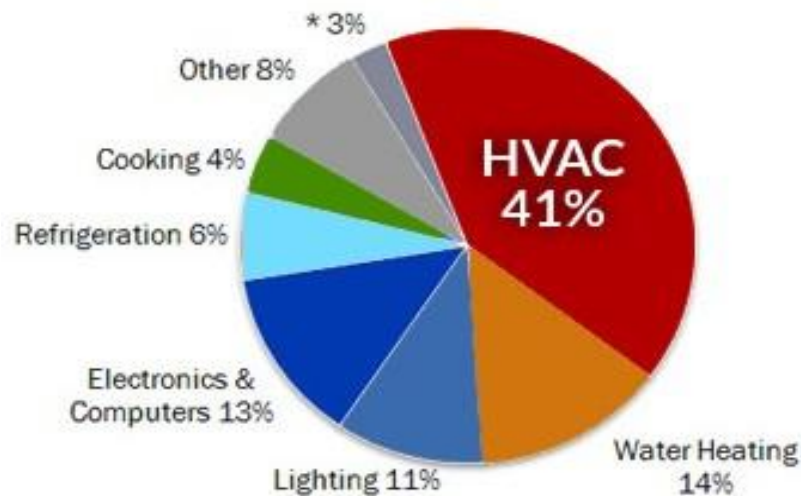


Ilustración 29. Consumos de energía en una edificación [65]

Los sistemas mal diseñados e ineficientes, aparte de no dar un nivel de servicio correcto, generan más gasto y contaminación. Igualmente sucede si el sistema está mal gestionado. Debido a la complejidad en su gestión, estos equipos funcionan bajo un control automatizado o sistema de gestión de edificios (BMS²⁰). Los sistemas de automatización tienen un funcionamiento fijo capaz de reaccionar al entorno interior o exterior según su lógica programada.

La capacidad para optimizar la energía de una manera eficiente y reaccionar a los cambios en tiempo real es vital para un funcionamiento óptimo de los sistemas. En los momentos de máxima demanda, donde la energía es más cara, tendrá un impacto grande en los costes de funcionamiento, lo mismo ocurre en los momentos de menor demanda si el sistema mantiene acciones innecesarias. Por estos motivos y frente al control adaptativo que ofrece la lógica programada, se hace necesario el desarrollo de un Sistema Inteligente, que aporte control predictivo, capaz de utilizar el aprendizaje automático para obtener los patrones de consumo de energía y poder ajustar el funcionamiento del sistema en previsión del entorno cambiante. En los últimos años, con el resurgimiento e implantación de técnicas de minería de datos y aprendizaje automático, son múltiples los estudios y proyectos encaminados en esta vía.

Introduciendo un software con Inteligencia Artificial se pueden optimizar las diferentes variables del sistema consiguiendo optimizar el flujo del aire, su calidad, humedad o temperatura, todo ello con un consumo de energía menor, y por tanto también bajando las emisiones de carbono. Las aplicaciones desarrolladas de esta forma para HVAC recogen datos de cada sensor, contador o máquina del sistema. Estos datos se analizan en tiempo real y junto con información externa como pueda ser un servicio de previsión meteorológica, y la información almacenada, envían las órdenes adecuadas par a los diferentes equipos del sistema [71].

3.1.1. Climatizador AHU (AIR HANDLING UNIT)

Una unidad de tratamiento de aire, o AHU por las siglas en inglés, es normalmente una caja de metal grande formada por la composición de varios elementos denominados módulos que contienen a su vez todos los elementos necesarios para la climatización de un edificio. Los módulos encargados del control del aire están conectados, generalmente, al sistema de ventilación del edificio, que distribuye el aire a

²⁰ BMS: *Building Management System*, https://es.wikipedia.org/wiki/Building_Management_System

través de conductos y lo devuelven de nuevo a la AHU, estos sistemas suelen estar colocados en las cubiertas del edificio o sus alrededores. Otros sistemas pueden descargar y recuperar el aire directamente del espacio aclimatado sin conductos, estos sistemas están integrados directamente en la instancia a aclimatar.



Ilustración 30. Unidad exterior AHU [72]

A continuación, se procederá a describir los principales tipos de componentes que forman parte de una AHU, así como los elementos que lo componen, intentando para ello seguir el recorrido que hace el aire desde su entrada en el sistema, hasta la salida por el suministro de la edificación [73] [62], se puede ver un esquema de los mismos en la Ilustración 31 :

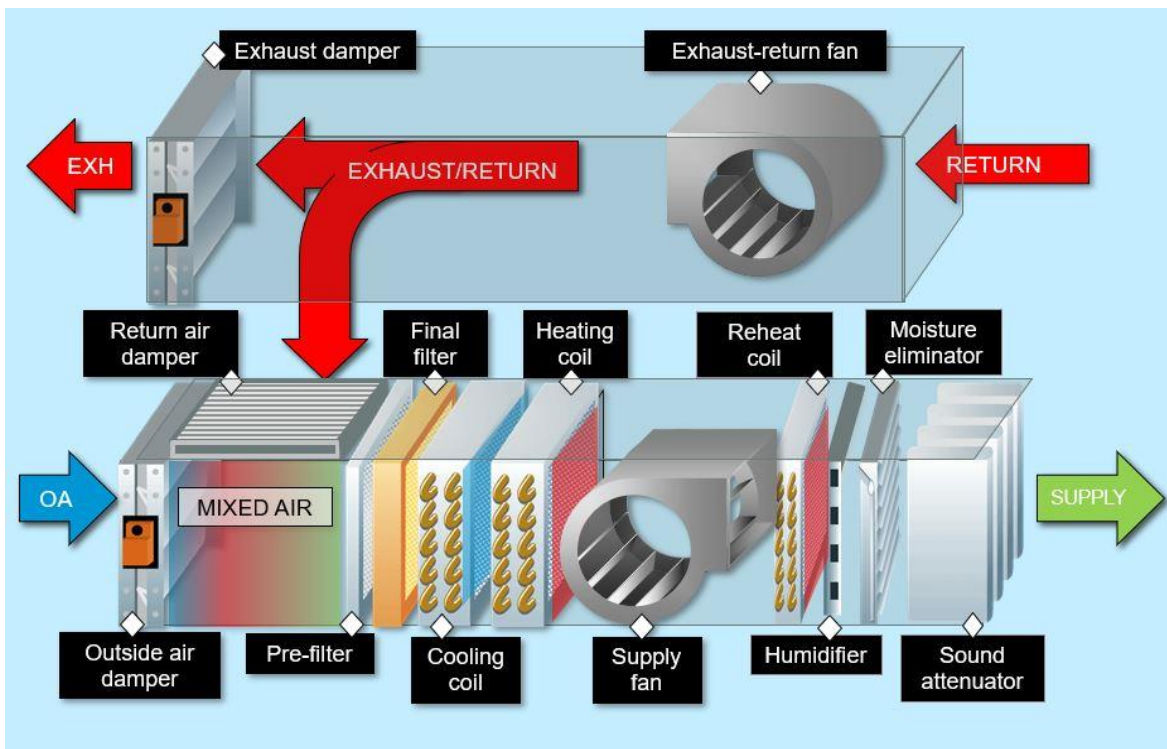


Ilustración 31. Esquema básico de unidad AHU [74]

- **Entrada de aire exterior (outside air damper).** Suelen tener unas palas o compuertas automatizadas que regulan el flujo de entrada, y por tanto controlan la presión del aire, además de prevenir la entrada de elementos externos de gran tamaño, como pueden ser hojas o plásticos.

- **Filtro de aire (*filters*).** Mantener el aire limpio y sin partículas es un requerimiento que se encuentra en casi todos los sistemas. Habitualmente se implementa mediante filtros tipo MERV, HEPA, electrostáticos o en combinaciones. Si hay requerimientos de filtrado de patógenos también pueden añadirse tratamientos de aire con luz ultravioleta o en fase gaseosa. Habitualmente los filtros se colocan en tandas de dos (o más) bancos sucesivos, con un filtro de panel de grado grueso colocado frente a otro filtro de bolsa de grado fino, u otro medio de filtración final.
- **Intercambiadores de calor y Serpentes (*heat exchanger and coils*).** Sirven para modificar la temperatura del aire suministrado al circuito, aportando calor y/o refrigeración. El acondicionamiento se produce mediante serpentines insertos en la corriente de aire dentro de la unidad de tratamiento del aire. Los serpentines se clasifican en directos o indirectos en relación al medio utilizado para el intercambio de temperatura. Los intercambiadores directos engloban los que queman combustible o gases colocados directamente en la corriente de aire, los calentadores de resistencia eléctrica y bombas de calor. En climas secos también incluirían los de enfriamiento por evaporación. Los indirectos emplean agua caliente o vapor para calentar y agua fría o glicol para refrigerar, proporcionados por otro módulo en el edificio. Es habitual incluir sensores de temperatura tanto dentro del serpentín como fuera, junto con placas de eliminación para drenar el condensado.

Si el sistema requiere deshumidificación se emplea un proceso capaz de reducir la humedad relativa del suministro de aire mediante el siguiente procedimiento: se obliga a que se produzca condensación mediante el sobreenfriamiento en el serpentín de enfriamiento. Un serpentín calentador colocado justo después del serpentín de enfriamiento, llamado serpentín de recalentamiento, recalienta el aire a la temperatura de suministro deseada. En climas fríos donde las temperaturas caen por debajo del punto de congelación, los serpentines de precalentamiento (también llamados de escarcha) se incluyen a menudo como una primera etapa para el tratamiento del aire que garantice la protección del sistema completo contra la congelación.

- **Humidificador (*humidifier*).** A parte de para aclimatar entornos que lo requieran, en los climas fríos, el calentamiento continuo del aire provoca su sequedad y electricidad estática, por lo que también se hace necesario. Existen varios métodos para humidificar el aire: por evaporación, el aire seco se proyecta sobre un depósito o sobre los deflectores de aire; por vaporizado directo o niebla de rociado sobre la corriente de aire; por neblina ultrasónica de agua o haciendo atravesar el aire un medio membranoso que se mantiene húmedo.
- **Cámara de mezclado (*mixing chamber*).** Es una zona del circuito que permite la introducción de aire exterior al sistema y mezclarlo con el aire del circuito, con el objetivo de mantener su calidad, así como la expulsión de aire interno para su renovación. La diferencia de temperaturas entre el aire existente y el incorporado permite realizar ajustes. La mezcla de aire se controla mediante reguladores (*dampers*).
- **Ventiladores (*fans*).** Pueden ser mecánicos o centrífugos y su objetivo es mover el aire. El caudal del aire puede controlarse mediante la modificación de la velocidad del ventilador o tener velocidad fija y situar unas palas mecánicas o amortiguadores a su salida. Los circuitos suelen tener varios ventiladores al final de la AHU y principio de los conductos de suministro llamados "ventiladores de suministro", que pueden complementarse con los "ventiladores de retorno" situados en el conducto de aire de retorno para empujar el aire de regreso hacia la AHU.
- **Dispositivo de recuperación de calor (*heat recovery device*).** Son intercambiadores de calor que se posicionan entre las corrientes de aire de suministro y extracción, y sirven para el ahorro de energía y aumento de la capacidad del sistema. Los principales intercambiadores utilizados son: recuperador de placas (*plate heat*), es un sándwich de placas de metal o plástico; rueda

térmica (*thermal Wheel*), consta de una matriz de metal corrugado en rotación lenta capaz con doble ciclo, de calefacción o enfriado; serpentín de circulación (*run around coil*) son dos serpentines conectados a una bomba de circulación por agua o salmuera como medio de transferencia de calor; tubería de calor (*heat pipe*) utiliza varios tubos en forma de serpentín, sellados con refrigerante como medio de transferencia de calor.

- **Aislantes, equilibradores y atenuadores (*isolators, balancing and attenuators*).** Forman parte de los componentes estructurales de la AHU. Para evitar que las vibraciones generen ruidos y movimientos que se trasladen al edificio, se insertan recubrimientos de goma entre los compartimentos que contienen los ventiladores. También se suelen colocar sobre resortes de suspensión, junto con los equilibradores, que son pesos convenientemente colocados para ajustar las oscilaciones del ventilador y mejorar su eficiencia. Las oscilaciones descontroladas afectan al desgaste de los elementos de la AHU. Para atenuar el ruido, suelen incluirse deflectores perforados aislados o colocados en línea, sustituyendo a los propios conductos, con forma de bocas acampanadas.
- **Control automatizado (*building automation*).** Son dispositivos que controlan y regulan todos los componentes de un sistema AHU. Pueden ser simples como un termostato de encendido/apagado, o complejos como los sistemas programables de automatización de edificios: BAS o BMS²¹. Los componentes de control comunes incluyen sensores de humedad, temperatura, CO₂, interruptores, controladores, actuadores o motores.

3.1.2. Representación BIM para HVAC-AHU

No existe un modelo base o unas características comunes para un HVAC o una AHU, dependerá de cada edificio, presupuesto, disponibilidad de materiales, etc. El modelo BIM ha garantizado la representabilidad de todos los elementos individuales constituyentes, evolucionando a partir de su versión IFC2x2 hasta la actual, la IFC4x4 (ISO 16739-1:2018) [75] para incluir en su definición esquemas de dominio propios de un HVAC/AHU y para facilitar la representabilidad y funciones de los sistemas BMS.

Estos esquemas se caracterizan por separar el modelado de componentes y sus interconexiones. El modelado de componentes separa los datos en tres capas: "Ocurrencia" relativa a la ubicación y conectividad; "Tipo" sobre su diseño, composición y geometría; e "historial de rendimiento", con un registro de los datos medidos y simulados por componente. En cuanto a las conexiones se establece un modelo de componentes conectados a puertos. Debido a la gran cantidad de componentes se establece una jerarquía de definiciones como se puede apreciar en la Ilustración 32.

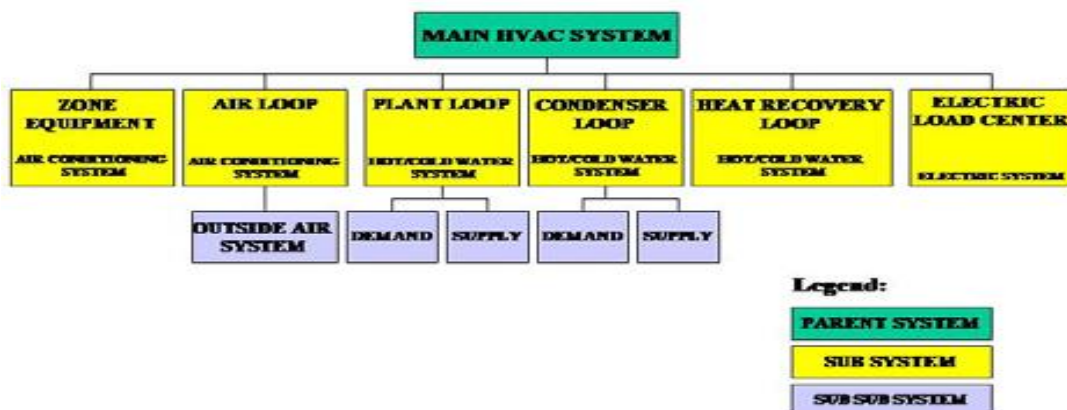


Ilustración 32. Jerarquía de definición sistema HVAC [76]

²¹ BAS / BMS: *Building Automation System / Building Management System*, https://en.wikipedia.org/wiki/Building_automation

Un ejemplo de la complejidad lo observamos en la definición del esquema que facilita IFC 4.0 para HVAC, dominio “*ifcHvacDomain*”. La Ilustración 33 define los diferentes tipos de datos necesarios, y la Tabla 3 describe las entidades principales que componen el sistema HVAC.

n.	Entities	n.	Entities	n.	Entities	n.	Entities
1	IfcAirTerminal	10	IfcCooledBeam	19	IfcFan	28	IfcPumpType
2	IfcAirTerminalBox	11	IfcCoolingTower	20	IfcFilter	29	IfcSpaceHeater
3	IfcAirToAirHeatRecovery	12	IfcDamper	21	IfcFlowMeter	30	IfcTank
4	IfcBoiler	13	IfcDuctFitting	22	IfcHeatExchanger	31	IfcTubeBundle
5	IfcBurner	14	IfcDuctSegment	23	IfcHumidifier	32	IfcUnitaryEquipment
6	IfcChiller	15	IfcDuctSilencer	24	IfcMedicalDevice	33	IfcValve
7	IfcCoil	16	IfcEngine	25	IfcPipeFitting	34	IfcVibrationIsolator
8	IfcCompressor	17	IfcEvaporativeCooler	26	IfcPipeSegment	35	IfcAirTerminalBoxType
9	IfcCondenser	18	IfcEvaporator	27	IfcPump	36	IfcAirTerminalType

Tabla 3. Entidades principales para ifcHvacDomain [77]

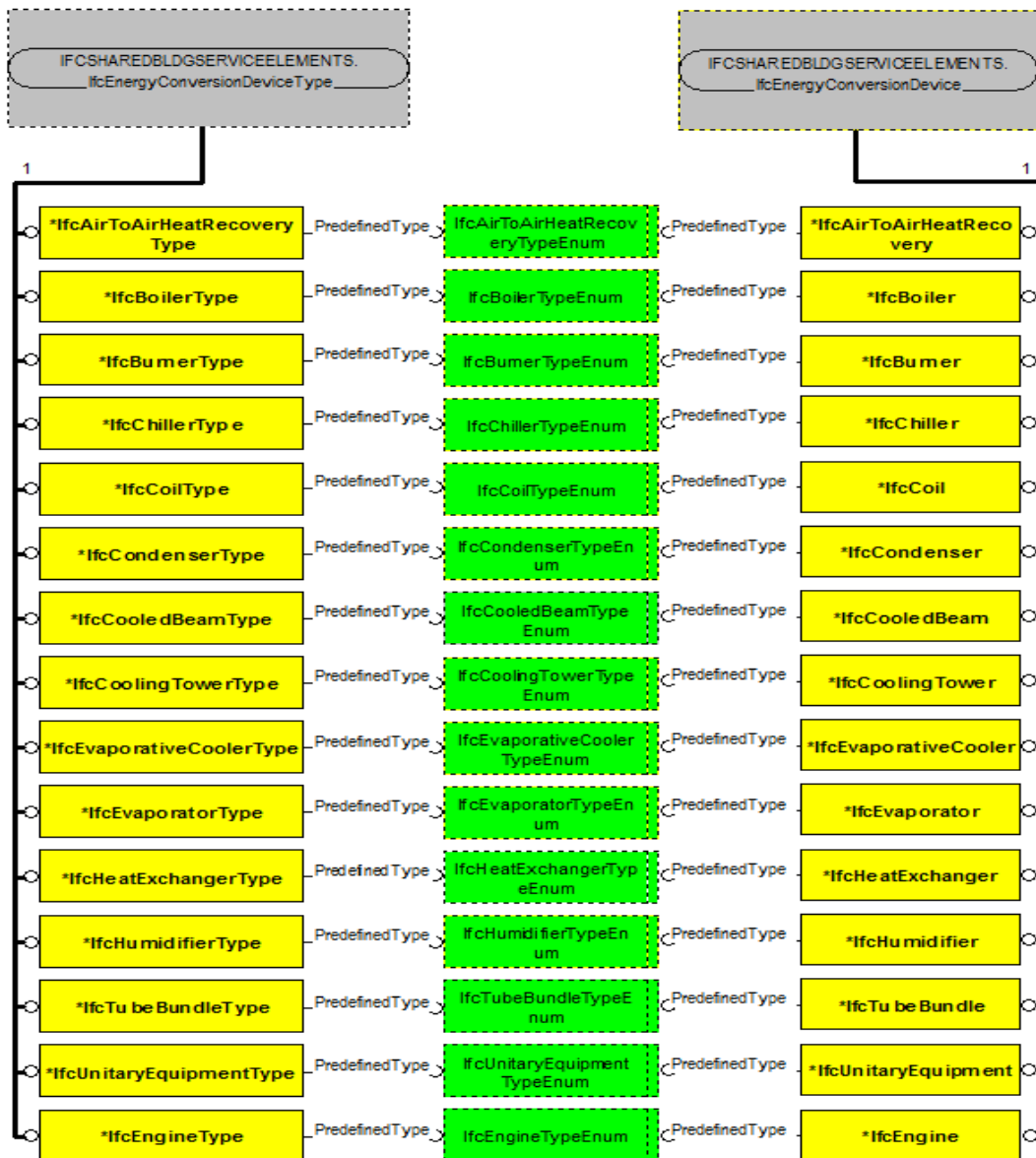


Ilustración 33. Tipos de datos necesarios para el *ifcHvacDomain* [77]

Toda herramienta de software BIM de uso común debe cumplir con el estándar IFC para el modelado geométrico y el intercambio de datos. El estándar IFC cubre, por tanto, la definición de los sistemas HVAC y además permite describir las funciones propias de los sistemas de control y automatización de edificios para su gestión, por ejemplo, a través de la clase “*IfcDistributionControlElement*” [78]. En cualquier caso, el modelo de datos IFC actual permite definir datos HVAC de diferentes maneras por lo que requiere una formulación de acuerdos sobre cómo determinados tipos de datos son definidos e interpretados por diferentes herramientas de software. Igualmente, algunas limitaciones en el software BIM actual, dificultan descripciones detalladas de las funciones de control asociadas a los sistemas BMS.

3.2. IFC OWL

IfcOWL [79] proporciona una representación web en OWL del esquema *Industry Foundation Classes* (IFC) visto en el Apartado 2.1. Establecido como estándar ontológico por la W3C [80], la ontología ifcOWL tiene el mismo estatus que los esquemas EXPRESS y XSD de IFC.

Todo el esquema IFC EXPRESS está representado en la ontología general ifcOWL, empleando las clases y propiedades de OWL lo que permite salvar la compatibilidad con los sistemas de Ingeniería de Software que lastraban a IFC. El proceso de conversión es totalmente abierto y bajo recomendación RDWG, siguiendo los principios siguientes:

- La ontología ifcOWL se mantiene en el perfil OWL-DL para permitir el razonamiento computable
- Se incluirán en ifcOWL axiomas para que coincida lo máximo con el esquema EXPRESS original
- Se establece como un objetivo principal la conversión de archivos de instancia IFC en archivos RDF equivalentes.

EXPRESS	OWL
Schema	Ontology
Simple data type	OWL class with a restriction on an owl:DatatypeProperty
Defined data type	OWL class
Constructed SELECT data type	OWL class with equivalence to a unionOf collection of OWL classes
Constructed ENUMERATION	OWL class with equivalence to a oneOf collection of owl:NamedIndividual items
Entity data type	OWL class with equivalence to a oneOf collection of owl:NamedIndividual items
Attribute of entity data type	Functional object property with specified domain and range; owl:AllValuesFrom restriction; owl:qualifiedCardinality or owl:maxQualifiedCardinality restriction
Attribute of entity data type as a SET	Non-functional object property with specified domain and range; owl:AllValuesFrom restriction; owl:minQualifiedCardinality and/or owl:maxQualifiedCardinality restriction or owl:qualifiedCardinality restriction
INVERSE attribute	object property with a specified owl:inverseOf
DERIVE attribute	N/A
WHERE rule	N/A
FUNCTION	N/A
RULE	N/A

Tabla 4. Conversión de términos adoptados entre EXPRESS y OWL

ifc b spline curve^c
[back to ToC](#) or [Class ToC](#)

IRI: https://standards.buildingsmart.org/IFC/DEV/IFC4/ADD2_TC1/OWL#IfcBSplineCurve

has super-classes

- [ifc bounded curve](#)^c
- [ifc b spline curve with knots](#)^c
- [ClosedCurve](#)^{op} **only** [ifc logical](#)^c
- [ControlPointsList](#)^{op} **only** [ifc cartesian point list](#)^c
- [ControlPointsList](#)^{op} **only** has next **some** has next **some** [ifc cartesian point list](#)^c
- [CurveForm](#)^{op} **only** [ifc b spline curve form](#)^c
- [Degree](#)^{op} **only** [ifc integer](#)^c
- [SelfIntersect](#)^{op} **only** [ifc logical](#)^c
- [ClosedCurve](#)^{op} **exactly** 1 [ifc logical](#)^c
- [ControlPointsList](#)^{op} **exactly** 1 [ifc cartesian point list](#)^c
- [CurveForm](#)^{op} **exactly** 1 [ifc b spline curve form](#)^c
- [Degree](#)^{op} **exactly** 1 [ifc integer](#)^c
- [SelfIntersect](#)^{op} **exactly** 1 [ifc logical](#)^c

has sub-classes

- [ifc b spline curve with knots](#)^c

is in domain of

- [ClosedCurve](#)^{op}, [ControlPointsList](#)^{op}, [CurveForm](#)^{op}, [Degree](#)^{op}, [SelfIntersect](#)^{op}

is disjoint with

- [ifc composite curve](#)^c, [ifc indexed poly curve](#)^c, [ifc polyline](#)^c, [ifc trimmed curve](#)^c

Ilustración 34. Ejemplo de especificación ifcOWL de la clase ifcBSplineCurve [80]

La creación de esta ontología con toda la base de conocimientos adquirida por IFC hace que su empleo completo, así como los procesos de razonamiento asociados, requieran de un hardware potente. Además, muchos de los estudios solamente abarcan áreas concretas o subdominios dentro de ifcOWL por lo que a lo largo de este tiempo han surgido adaptaciones simplificadas sobre ifcOWL dependiendo del subdominio de trabajo que sea requerido, al igual que ocurre con EXPRESS. Este es el caso de la familia de ontologías SAREF que veremos en el siguiente apartado.

3.3. SAREF

En 2013 la Comisión Europea publica un estudio sobre “activos semánticos disponibles para la interoperabilidad de dispositivos inteligentes” y crea la iniciativa para la estandarización SMART 2013/0077 para especificaciones Smart Building. La agencia europea de estandarización (ETSI) propone asumir la estructura semántica para dispositivos M2M²² y crear una ontología de referencia. En 2017 se publica la primera especificación ETSI SAREF como primera ontología estándar para el ecosistema IoT basada en el estándar IFC. Dada la gran diversidad de sectores y actividades implicadas por el IoT, se necesita una vertebración que garantice la interoperabilidad entre soluciones de diferentes proveedores y sectores de actividad. Se establece, por tanto, como sistema unificador de base oneM2M con ETSI como fundador. OneM2M proporciona el marco de comunicación y trabajo para compartir los datos entre los diferentes desarrollos y aplicaciones, y SAREF proporciona la interoperabilidad semántica necesaria para compartir la información aportada en los datos y por tanto facilitar la unificación de dispositivos [81].

La ontología SAREF, como puede verse en la Ilustración 35, proporciona los llamados **building blocks**, que permiten su separación y por tanto modificación en diferentes partes (ontologías) en función de las necesidades específicas del dominio.

²² M2M: Machine 2 Machine, máquina a máquina. Intercambio de información entre máquinas remotas.

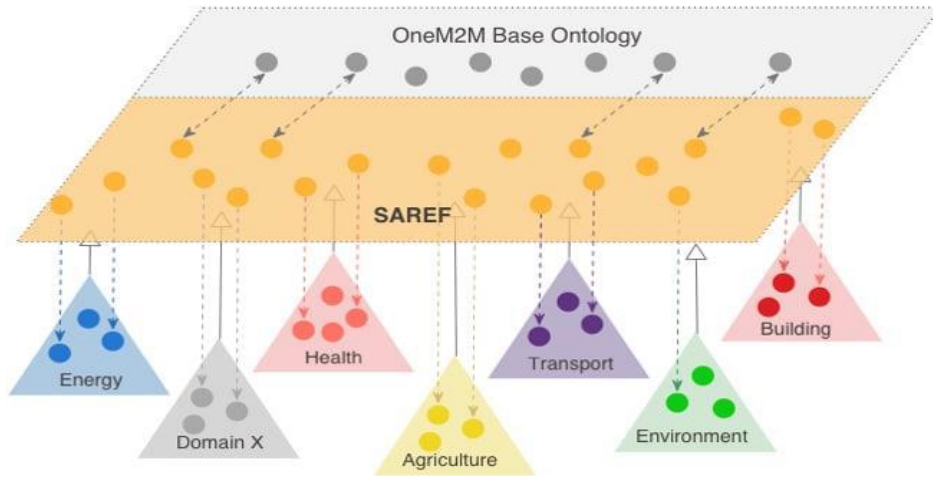


Ilustración 35. Estructura de SAREF [81]

El desarrollo de SAREF se centra en la interconectividad M2M y las relaciones humanas, teniendo en cuenta en la base fundamental ontologías como ifcOWL con la que guarda una concordancia de elementos como parte fundamental. Se parte del concepto de dispositivo como eje central de desarrollo. A continuación se amplían algunas de las clases fundamentales ya que algunas serán utilizadas en el desarrollo del TFG o serán útiles para posibles desarrollos futuros:

Dispositivo (*saref:Device*)

Un dispositivo es un objeto tangible diseñado para cumplir una tarea particular en el mundo real (edificios públicos, privados, lugares comunes, infraestructuras, etc.). Para realizar esta tarea, un dispositivo realiza una o más funciones. Ejemplos de dispositivos serían un interruptor de la luz, un sensor de humo, un secador de manos. El secador de manos tiene la tarea de secar y para realizarla tiene una función de inicio y otra de parada. A su vez un dispositivo puede tener algunas propiedades que lo caracterizan de manera única, como puede ser su fabricante (*saref:hasManufacturer*), modelo (*saref:hasModel*), etc. En la Ilustración 36 podemos observar una vista general de las principales clases y relaciones de SAREF, centrada en dispositivo.

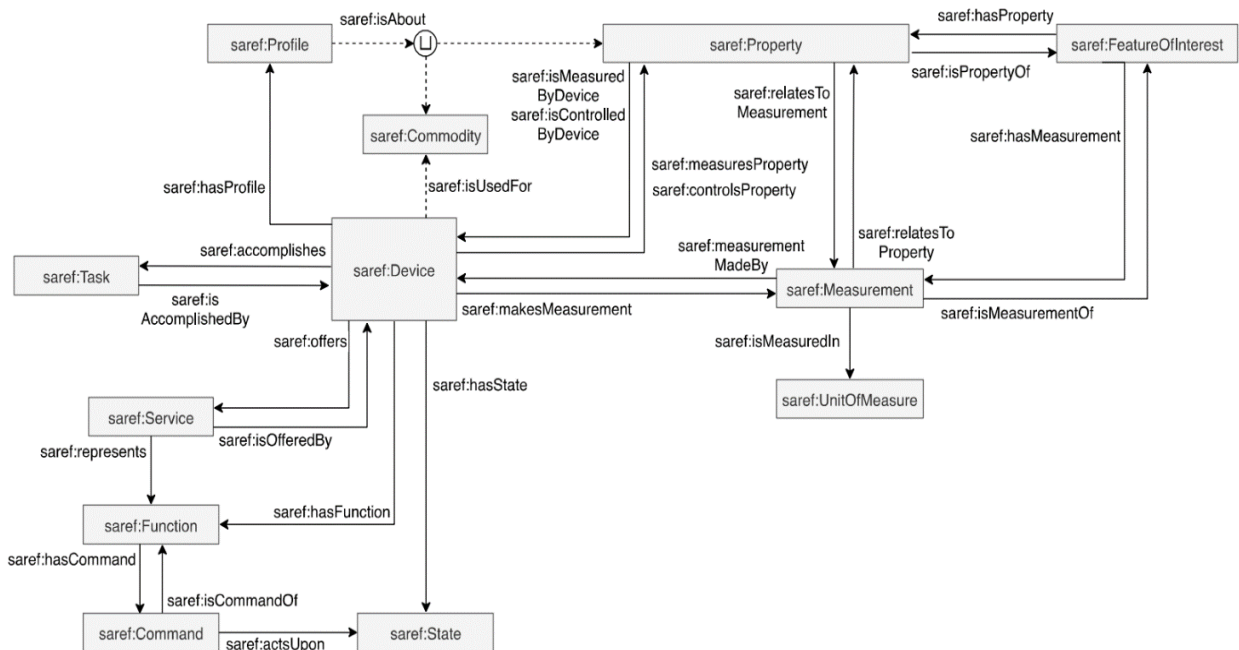


Ilustración 36. Vista general de la ontología SAREF [82]

SAREF está pensado de una forma modular que permite definir cualquier dispositivo a partir de bloques preestablecidos, teniendo en cuenta la función (o funciones) que realiza el dispositivo. Esto implica que un *saref:Device* tiene al menos una función (*saref:hasFunction* min 1 *saref:Function*). Igualmente, un *saref:Device* se puede utilizar para un propósito concreto (*saref:isUsedFor* property), este propósito es ofrecer un producto, como puede ser agua (*saref:Water*). También puede medir una propiedad, como la temperatura (*saref:Temperature*) o el consumo energético (*saref:Energy*), etc. Siguiendo el principio de modularidad, un dispositivo también puede estar formado por otros dispositivos, para lo que se emplea la propiedad *saref:consistsOf*.

En la Ilustración 37 se muestran los tipos de dispositivos representables, que son: *saref:Appliance*, como artefacto genérico; *saref:Actuator* como actuador (con el ejemplo ampliado de un interruptor, un *saref:Switch*); *saref:Sensor* para sensores, también con ejemplos; *saref:Meter*, medidores y un ejemplo de accesorio o maquinaria: *saref:HUVAC* predefinido que representa un sistema de climatización. Las sucesivas extensiones de SAREF, una de las cuales se ampliará en el siguiente apartado, incluyen y amplían dispositivos en función del dominio tratado.

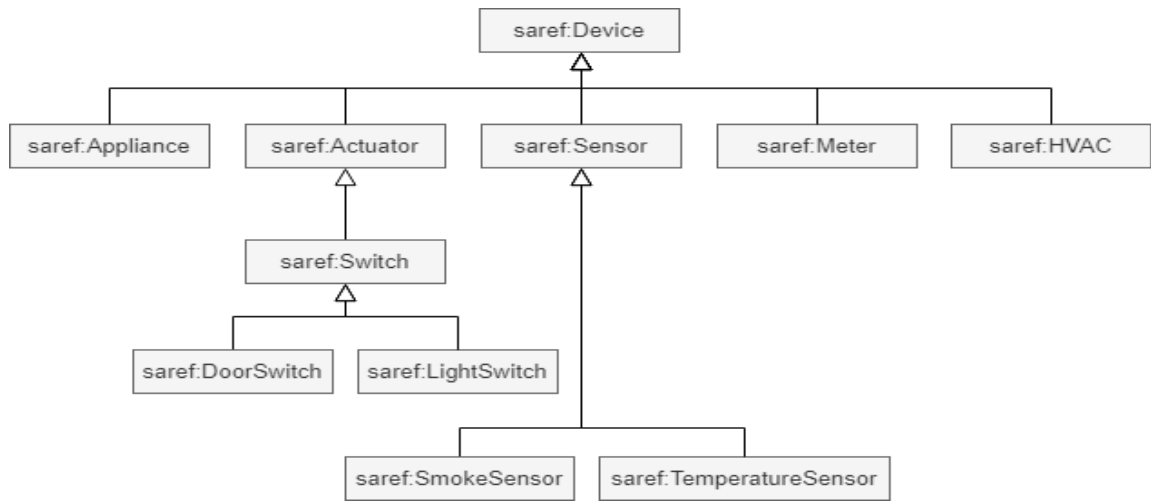


Ilustración 37. SAREF: tipos de dispositivos [82]

Dispositivo (*saref:Function*)

Una función se representa con la clase *saref:Function* y se define como "la funcionalidad necesaria para realizar la tarea para la que se ha diseñado un dispositivo". Por tanto dependen del dispositivo implementado. Un ejemplo lo tendríamos con un sensor de temperatura: es un dispositivo que consta de un sensor, de tipo *saref:Sensor*, que realiza la función de medir, *saref:SensingFunction* y, concretamente, mide una propiedad de tipo *saref:Temperature*. En la podemos ver las funciones definidas por SAREF, ampliadas en sus extensiones.

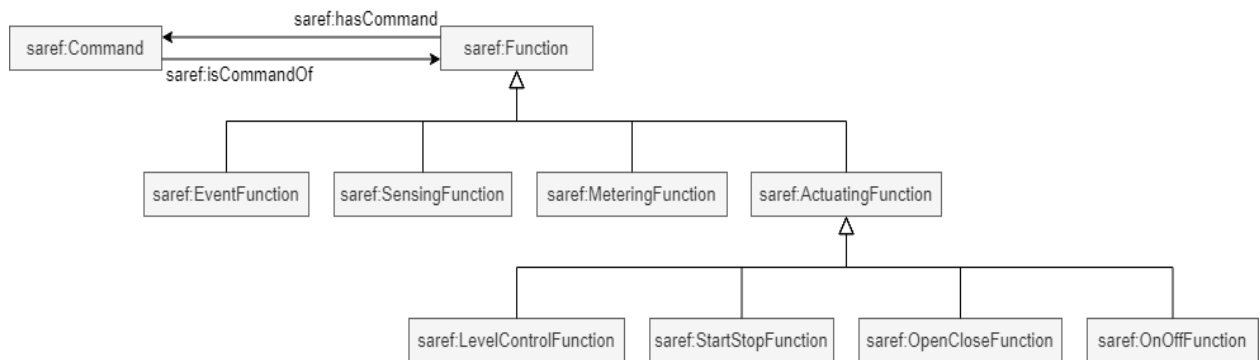


Ilustración 38. SAREF: clase Function y subclases [82]

Perfil (saref:Profile)

Un dispositivo puede tener asociado un perfil como se puede ver en la Ilustración 39. Un perfil es una especificación asociada a un dispositivo para recopilar información sobre una determinada propiedad (por ejemplo, consumo de energía o agua) para poder optimizar su uso, utilizando la propiedad *saref:isAbout*. Mediante la asignación de un espacio temporal y un coste con las propiedades *saref:hasTime* y *saref:hasPrice* respectivamente se pueden efectuar dichos cálculos.

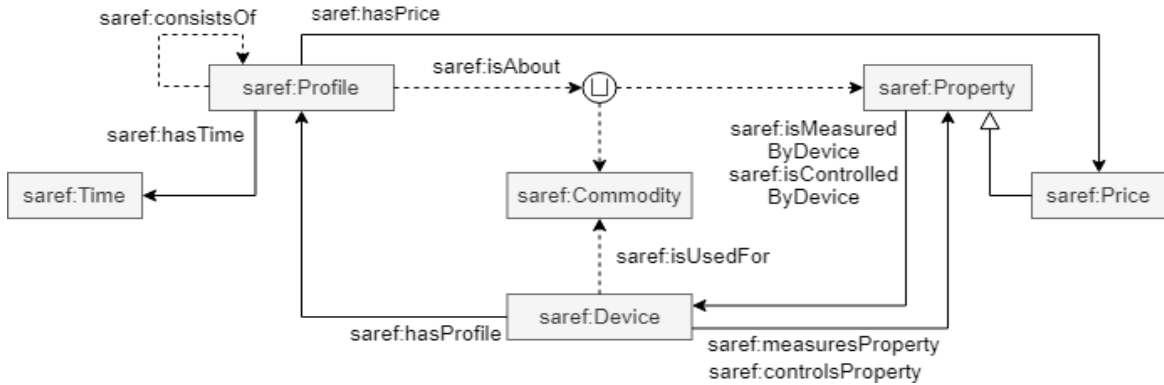


Ilustración 39. SAREF: clase Profile [82]

Medida, propiedad, unidad de medida (saref:Measurement, saref:Property, saref:UnitOfMeasure)

Estas clases permiten relacionar o mapear las diferentes mediciones que un dispositivo dado pueda realizar (usando la propiedad *saref:hasValue*) y asociarlas con las diferentes propiedades medidas, en sus correspondientes unidades (la clase *saref:Measurement* describe una medida de una cantidad física) para una propiedad (*saref:Property*) y de acuerdo con un *saref:UnitOfMeasure* dado. De esta forma, es posible diferenciar entre propiedades y las medidas realizadas para tales propiedades, y almacenar medidas para una propiedad concreta en diferentes unidades de medida. Se puede incluir una marca temporal (propiedad *saref:hasTimestamp*) para identificar cuándo la medida se aplica a la propiedad, se puede usar para medidas individuales o para series de medidas (por ejemplo, flujos de medidas). La clase *saref:Time* permite especificar el concepto del tiempo en términos de intervalos o instantes conforme a la ontología de tiempo de la W3C existente(<http://www.w3.org/2006/time#>).



Property	Definition
saref:hasTimeStamp only xsd:dateTime	The timestamp of a measurement is represented only by xsd:dateTime.
saref:hasValue exactly 1 xsd:float	A measurement should have exactly one value represented using xsd:float.
saref:hasValue only xsd:float	The value of a measurement is represented only by xsd:float.
saref:isMeasuredIn exactly 1 saref:UnitOfMeasure	A measurement should have exactly one unit of measurement which should be instance of saref:UnitOfMeasure.
saref:isMeasuredIn only saref:UnitOfMeasure	The unit of measurement of a measurement is represented only by instances of the class saref:UnitOfMeasure.
saref:relatesToProperty exactly 1 saref:Property	A measurement should be related exactly to one property which should be instance of saref:Property.
saref:relatesToProperty only saref:Property	The property to which a measurement is related to is represented only by instances of the class saref:Property.

Ilustración 40. SAREF modelo para Measurement y restricciones [83]

Es igualmente posible seguir una dirección inversa en las medidas: un *saref:Device* realiza una medición (*saref:MakesMeasurement*) en una determinada unidad de medida (*saref:UnitOfMeasure*), y esta medición puede relacionarse con una propiedad (*saref:Property*) utilizando *saref:relatesToProperty*. Un *saref:FeatureOfInterest* representa cualquier entidad del mundo real a partir de la cual se mide un *saref:Property*. Se puede seguir el esquema en la Ilustración 41.

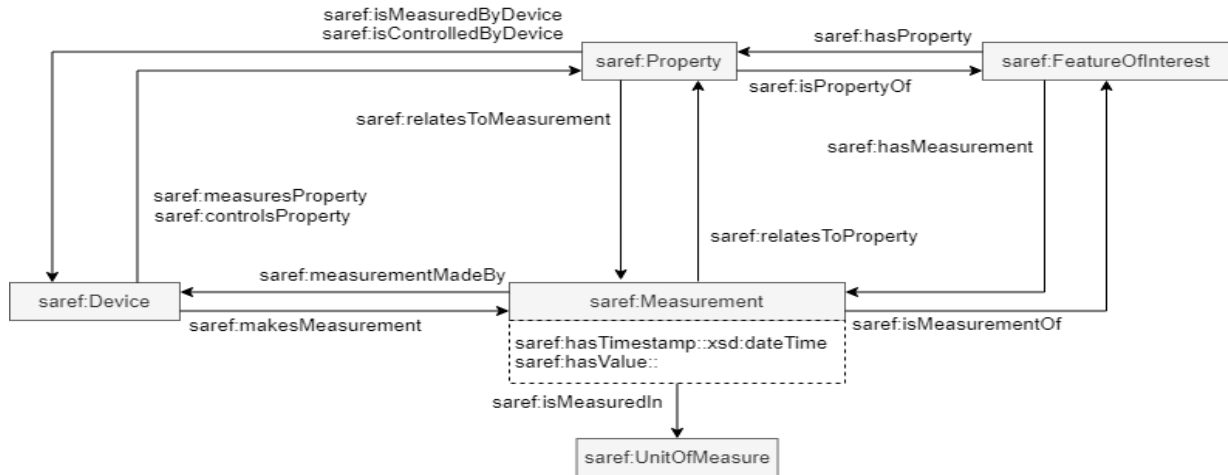


Ilustración 41. SAREF: clase y propiedades de Measurement [82]

Como ya se ha ido avanzado, a partir de 2017 se van creando las diferentes extensiones de SAREF con definiciones adicionales según el dominio tratado, algunas de ellas se muestran a continuación:

- [SAREF4ENER](#), especificación ETSI TS 103 410-1 V1.1.2 (2020-05). Extensión para el dominio energético creada en colaboración con las asociaciones [Energy@Home](#) y [EEBus](#)
- [SAREF4CITY](#), especificación ETSI TS 103 410-4 V1.1.2 (2020-05). Extensión para el dominio de las *Smart Cities*, los casos de uso incluyen *e-Salud* y estacionamiento inteligente, monitoreo de la calidad del aire, movilidad y alumbrado público, etc.
- [SAREF4INMA](#), especificación ETSI TS 103 410-5 V1.1.2 (2020-05). Extensión para el dominio de la Industria y Manufacturación, contempla la falta de interoperabilidad entre equipos de producción, la cadena de valor para rastrear el producto desde su origen hasta el consumidor de manera única, lotes, etc.
- [SAREF4AGRI](#), especificación ETSI TS 103 410-6 V1.1.2 (2020-05). Extensión para el dominio de la agricultura inteligente y la cadena alimentaria. Las fuentes de interés incluyen GPS, datos meteorológicos, observación remota vía satélite y observación local usando sensores cercanos o proximales que miden parámetros relevantes para la agricultura, incluidos el movimiento y la temperatura del ganado, la humedad en el suelo, el valor de PH, la salinidad y el color de las plantas (NDVI).
- [SAREF4WATR](#), especificación ETSI TS 103 410-10 V1.1.1 (2020-07). Extensión para el dominio del agua. Se centra en el modelado de dispositivos contadores de agua y sus características de interés. Clasifica los diferentes tipos de agua, sus usos y propiedades así como la infraestructura que conlleva su gestión.
- [SAREF4EHAW](#), especificación ETSI TS 103 410-8 V1.1.1 (2020-07). Extensión para el dominio de *e-Salud*, teleasistencia y gestión de productos, recursos y servicios de la salud.
- [SAREF4BLDG](#), especificación ETSI TS 103 410-3 V1.1.2 (2020-05). Extensión para el dominio de Edificios Inteligentes que se ampliará en el siguiente apartado.

- [SAREF4LIFT](#), especificación ETSI TS 103 410-11 V1.1.1 (2021-07). Extensión para el dominio de los dispositivos de carga.
- [SAREF4WEAR](#), especificación ETSI TS 103 410-9 V1.1.1 (2020-07). Extensión para el dominio de los dispositivos “wereable” o ropa inteligente.

Para el desarrollo de este TFG es requisito fundamental el uso de la ontología SAREF, aunque existen diferentes tecnologías para la integración semántica de datos con objeto de mejorar la eficiencia en los sistemas de IoT, y en el dominio de los sistemas de climatización. Podemos ver una comparativa en el artículo de W. Li y otros (2019) [84] donde figuran ontologías como oneM2M, WoT TD, SNS & SOSA, CIM Information Model y SAREF junto con un breve estudio sobre sus distintas coberturas y alguna referencia a desarrollos ontológicos de iniciativa privada.

3.4. SAREF4BLDG

La especificación SAREF4BLDNG (ETSI TS 103 410-3) [83] es una extensión de la ontología SAREF creada en base al estándar IFC, concretamente la IFC4, para la información sobre la construcción, limitándose a dispositivos o accesorios dentro del dominio de un edificio. El uso de SAREF4BLDG está pensado para que los dispositivos inteligentes de los fabricantes que admiten el modelo de datos IFC se comunicarán fácilmente entre sí. Esta compatibilidad obliga a generar descripciones de dispositivos neutrales que faciliten el intercambio de datos entre los distintos actores implicados en el dominio.

SAREF4BLDG se centra en ampliar la ontología SAREF para incluir los dispositivos definidos por IFC versión 4 y permitir la representación de dichos dispositivos y otros objetos físicos en espacios de construcción. Concretamente amplía 67 clases, 177 propiedades de objeto y 82 propiedades de tipo de datos, así como se reutilizan algunas de las clases de SAREF. La Ilustración 42 presenta una descripción general de las clases de nivel superior de la ontología y las propiedades. Aparece la clase *s4bldg:Building* para representar edificios y se redefine *geo:SpatialThing* de SAREF para incluir espacios de construcción dentro del edificio (*s4bldg:BuildingSpace*). Además se ha implementado una generalización (*s4bldg:PhysicalObject*) que permite que los espacios de construcción contengan tanto dispositivos (*saref:Device*) como objetos de construcción (*s4bldg:BuildingObject*).

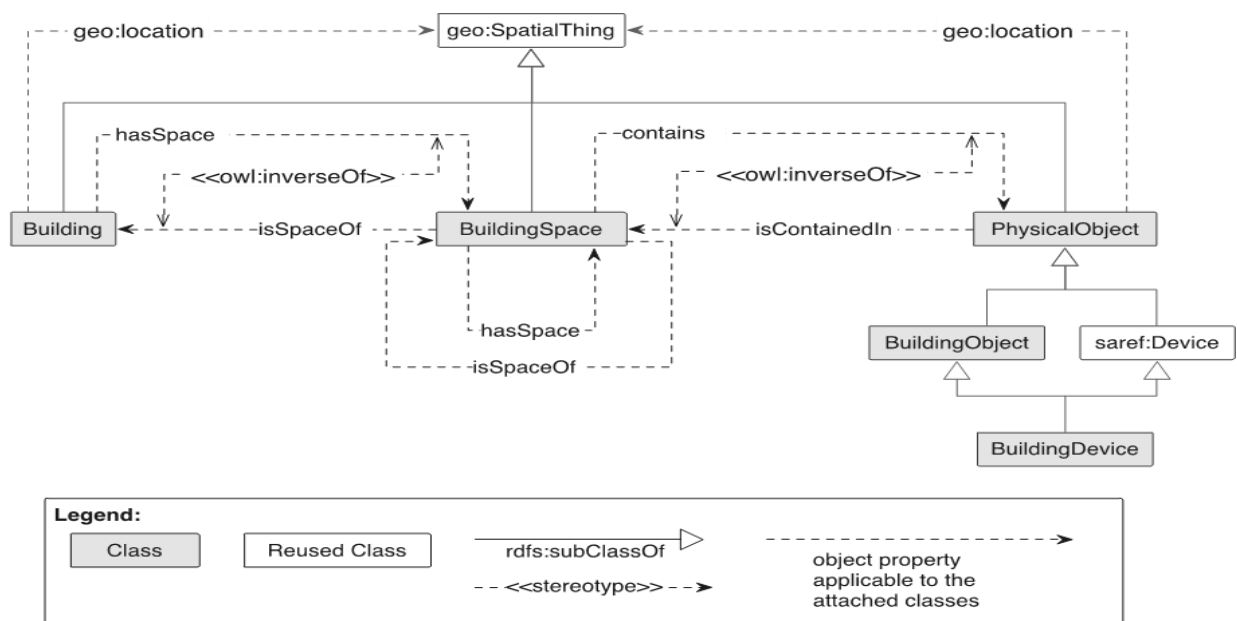


Ilustración 42. Vista general de la extensión SAREF4BLDG [83]

La representación de los dispositivos definidos en el estándar IFC y sus conexiones con SAREF es la principal razón para generar esta extensión. Para ello crea una jerarquía con 62 clases teniendo en cuenta el subconjunto de la jerarquía IFC relacionada con los dispositivos relacionados con *Smart Building* junto con varias clases que facilitan su categorización. Las clases para los dispositivos se organizan en 6 niveles jerárquicos, en la Ilustración 43 aparecen los cinco primeros niveles y el resto en la Ilustración 44. Las clasificaciones entre elementos superiores como los de transporte (*s4bldg:TransportElement*) y aislamientos de vibraciones (*s4bldg:VibrationIsolation*) y *s4bldg:Device* corresponden a alineaciones con IFC. De hecho, la mayoría de los tipos de dispositivos incluidos en IFC pertenecen a la categoría de dispositivos de distribución de ahí su mayor extensión (Ilustración 44), además, las clases creadas en esta extensión se han relacionado con la ontología ifcOWL siempre que ha sido posible. Esta relación ha sido declarada mediante la propiedad de anotación *rdfs:seeAlso*.

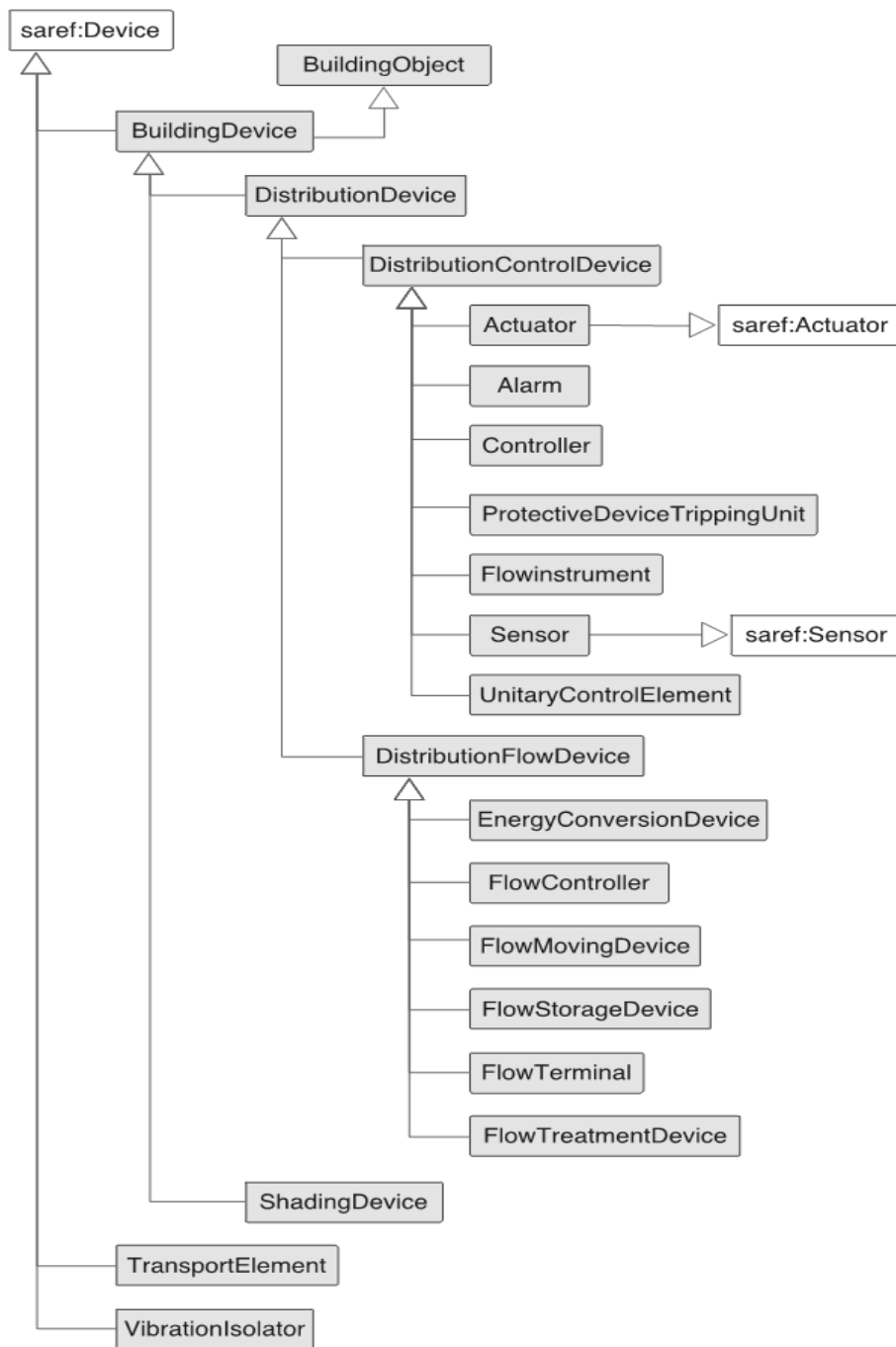


Ilustración 43. SAREF4BLDG: jerarquía para Device (Part.1) [83]



Ilustración 44. SAREF4BLDG: jerarquía para Device (Part.2) [83]

En la extensión de este estándar aparece altamente implicado el *Ontological Engineering Group* de la Universidad Politécnica de Madrid mediante desarrollos de los Dres. María Poveda Villalón and Raúl García Castro integrantes del grupo. Las adaptaciones y recomendaciones de este grupo a SAREF4BLDG serán las aplicadas en el desarrollo del TFG como requisito del proyecto.

Capítulo 4

4. Metodología

Como se ha expresado en el Apartado 2.4.7, el desarrollo de ontologías en proyectos de ingeniería, viene acompañado de una serie de metodologías específicas. Estas metodologías incorporan a los modelos ya conocidos de Ingeniería de Software elementos para su adaptación al dominio ontológico. En este capítulo se procede a detallar la metodología NeOn, empleada en el desarrollo del TFG.

4.1. Introducción

La metodología NeOn [85] [86] propuesta por Gómez-Pérez y Suárez-Figueroa [87] pertenecientes al *Ontological Engineering Group* de la Universidad Politécnica de Madrid, surge en 2009 con la necesidad de aportar un enfoque práctico a los diseños ontológicos y diferenciarse de otros enfoques más metodológicos imperantes. Para abordar esta necesidad se propone una metodología basada en múltiples escenarios combinables entre sí de manera flexible, junto con un conjunto de componentes y recursos que facilitan el desarrollo de ontologías y redes de ontologías desde cero, o partiendo de trabajos ya existentes. Se normaliza la reutilización de recursos tanto ontológicos como no ontológicos, y se incluye la evolución y mantenimiento de las ontologías o sus modelos en red.

Los componentes principales que se definen en esta metodología son:

- Nueve escenarios, solos o combinados permiten ajustar la construcción de ontologías o redes de ontologías en base a los requisitos iniciales que se ajusten a ellos.
- Un Glosario, identifica y define las actividades y procesos a seguir para la construcción de ontologías o redes de ontologías.
- Dos modelos de ciclo de vida para la construcción de ontologías y redes de ontologías, especifican el flujo que se debe seguir en su desarrollo.
- Un conjunto de pautas metodológicas definidas para desarrollar todas las actividades y los procesos para completar el desarrollo utilizando los componentes anteriores.

4.2. Escenarios

El activo clave de la metodología NeOn es el conjunto de nueve escenarios para la construcción de ontologías y redes de ontologías. Se han creado enfatizando la reutilización de recursos ontológicos y no ontológicos, generalizando a partir de experiencias previas, abarcando los inconvenientes de las metodologías existentes, y teniendo en cuenta la colaboración y el dinamismo. La Ilustración 45 presenta los escenarios planteados para cualquier desarrollo, a excepción del escenario 1 que es obligatorio, el resto de escenarios pueden combinarse dependiendo de las necesidades del proyecto.

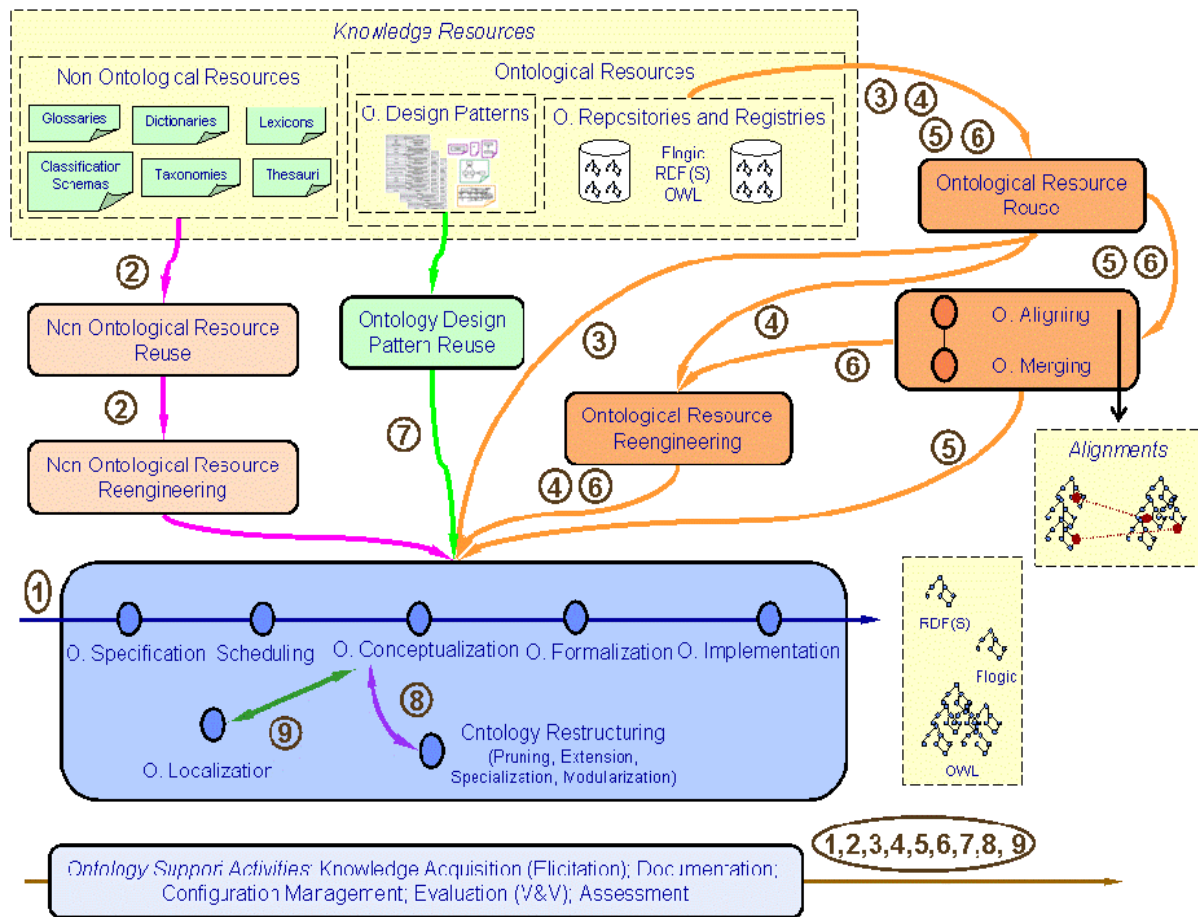


Ilustración 45. Escenarios para la metodología NeOn [87]

- **Escenario 1:** Representa un ciclo de vida habitual, visto en otras metodologías anteriores, para la creación de una ontología desde cero. Va desde la especificación hasta la implementación, sin reutilizar los recursos existentes. Los desarrolladores deben especificar los requisitos de la ontología.
- **Escenario 2:** Desarrollo de redes de ontologías mediante reutilización y reingeniería de recursos no ontológicos (NOR). En este escenario se identifican y seleccionan recursos no ontológicos para su utilización dentro de la ontología.
- **Escenario 3:** Desarrollo de redes de ontologías mediante reutilización de recursos ontológicos. Este escenario aborda directamente la reutilización de recursos ontológicos existentes. Una vez definido el contexto para el desarrollo de la ontología, se procede a la búsqueda y selección de otros recursos ontológicos desarrollados con anterioridad y que se adecúen al contexto.
- **Escenario 4:** Desarrollo de redes de ontologías mediante reutilización y reingeniería de recursos ontológicos. Es una variación sobre el escenario anterior donde una vez buscados y seleccionados recursos ontológicos existentes que puedan diferir ligeramente de nuestro contexto, se realiza una reingeniería para adaptarlos a las necesidades del proyecto en curso.
- **Escenario 5:** Desarrollo de redes de ontologías mediante reutilización y mezcla de recursos ontológicos. Se realiza la fusión de recursos existentes a reutilizar (ontologías o redes de ontologías) con los nuevos que se crean.
- **Escenario 6:** Desarrollo de redes de ontologías mediante reutilización, mezcla y reingeniería de recursos ontológicos. Sería el escenario anterior, pero incluyendo una reingeniería de recursos en base a los requerimientos de la nueva ontología, al estilo del escenario 4.

- **Escenario 7.** Desarrollo de redes de ontologías mediante reutilización de patrones de diseño ontológico (ODP²³). Para este escenario se buscan y reutilizan patrones existentes de diseño ontológico.
- **Escenario 8.** Desarrollo de redes de ontologías mediante reestructuración de recursos ontológicos. En este escenario se reestructura una ontología existente, se puede ampliar, reducir, o especializar en base a las necesidades del dominio.
- **Escenario 9.** Desarrollo de redes de ontologías mediante localización de recursos ontológicos. En este escenario se realiza la modificación de otras ontologías existentes modificando características generales tales como: idioma, cultura, localidad, etc., en base a los requerimientos del dominio.

4.3. Glosario de procesos y actividades

Este glosario recopila los principales procesos y actividades potencialmente involucrados en la construcción de una ontología o red de ontologías (dentro del campo específico de la ingeniería de ontologías) y proporciona algunas definiciones y explicaciones en lenguaje natural comúnmente utilizadas. El vocabulario incluido en el glosario es monolingüe (inglés) e incluye 59 definiciones de procesos y actividades ordenadas alfabéticamente. Incluye notas para aclarar procesos y actividades sinónimos o en apariencia similares. La Ilustración 46 muestra el listado de todos los procesos y actividades, marcando varios en azul para ampliarlos como ejemplo.

Ontology Aligning	Ontology Implementation	Ontology Restructuring
Ontology Annotation	Ontology Integration	Ontology Repair
Ontology Assessment	Knowledge Acquisition for Ontologies	Non-ontological Resource Reuse
Ontology Comparison	Ontology Learning	Ontological Resource Reuse
Ontology Conceptualization	Ontology Localization	Ontology Reuse
Ontology Configuration Management	Ontology Mapping	Ontology Reverse Engineering
Control	Ontology Matching	Scheduling
Ontology Customization	Ontology Merging	Ontology Search
Ontology Design Pattern Reuse	Ontology Modification	Ontology Selection
Ontology Diagnosis	Ontology Modularization	Ontology Specialization
Ontology Documentation	Ontology Module Extraction	Ontology Requirements Specification
Ontology Elicitation	Ontology Module Reuse	Ontology Statement Reuse
Ontology Enrichment	Ontology Partitioning	Ontology Summarization
Ontology Environment Study	Ontology Population	Ontology Translation
Ontology Evaluation	Ontology Pruning	Ontology Update
Ontology Evolution	Ontology Quality Assurance	Ontology Upgrade
Ontology Extension	Non-Ontological Resource Reengineering	Ontology Validation
Ontology Feasibility Study	Non-Ontological Resource Reverse Engineering	Ontology Verification
Ontology Formalization	Non-Ontological Resource Transformation	Ontology Versioning
Ontology Forward Engineering	Ontology Reengineering	

Ilustración 46. NeOn: procesos y actividades incluidas en el glosario [42]

Los ejemplos de definiciones marcados en el Glosario NeOn son:

²³ ODP: Ontology Design Patterns, http://ontologydesignpatterns.org/wiki/Main_Page

- **Modularización de ontologías.** Se refiere a la actividad de identificar uno o más módulos en una ontología con el propósito de soportar la reutilización o el mantenimiento. Podemos hacer distinciones entre la extracción de módulos de ontología y el particionamiento de ontología.

- **Reutilización de módulos de ontología.** Se refiere al proceso de utilizar módulos de ontología en la solución de diferentes problemas.

- **Reutilización de recursos no ontológicos.** Se refiere al proceso de tomar los recursos no ontológicos (bases de datos, vocabularios controlados, etc.) disponibles para el desarrollo de ontologías

El Glosario de Procesos y Actividades de NeOn está publicado en la web oficial de la metodología, y puede considerarse como un primer paso para solucionar la falta de un glosario estándar en la Ingeniería Ontológica:

<http://www.neon-project.org/web-content/images/Publications/neonglossaryofactivities.pdf>

Se puede ver un extracto del mismo en la Ilustración 47. De hecho, actualmente es un proceso abierto cuyo objetivo es obtener retroalimentación de la comunidad de Ingeniería Ontológica sobre el glosario actual y se pueden realizar contribuciones a través del enlace:

http://cicero.uni-koblenz.de/wiki/index.php/Prj:NeOn_Glossary_of_Processes_and_Activities

NeOn Glossary of Activities

- **Ontology Aligning** refers to the activity of finding the correspondences between two or more ontologies and storing/exploiting them. A synonym for this activity is Ontology Mapping.
- **Ontology Annotation** refers to the activity of enriching the ontology with additional information, e.g. metadata or comments.
- **Ontology Assessment** refers to the activity of checking an ontology against user requirements, such as usability, usefulness, abstraction, quality, etc.
- **Ontology Comparison** refers to the activity of finding differences between two or more ontologies or between two or more ontology modules.
- **Ontology Conceptualization** refers to the activity of organizing and structuring the information (data, knowledge, etc.), obtained during the acquisition process, into meaningful models at the knowledge level according to the ontology specification document. This activity is independent of the way in which the ontology implementation will be carried out.
- **Ontology Configuration Management** refers to the activity of recording all the versions of the documentation, software and ontology code, and of controlling the changes.
- **Control** refers to the activity of guaranteeing that scheduled activities in the ontology development process are completed in the manner intended to be performed.
- **Ontology Customization** refers to the activity of adapting an ontology to a specific user's needs.
- **Ontology Diagnosis** refers to the activity of identifying parts of the ontology directly responsible for incorrectness and incompleteness. Ontology diagnosis is triggered by ontology validation.
- **Ontology Documentation** refers to the collection of documents and explanatory comments generated during the entire ontology building process.

Note: Examples of documents external to the implemented ontology include ontology specification documents, sources used for acquiring knowledge, ontology conceptualization document, design and decision criteria, ontological commitments, etc.

Information inside the implemented ontology includes natural language

Ilustración 47. NeOn: extracto del Glosario de Procesos y Actividades [42]

4.4. Ciclos de vida

El ciclo de vida de la red de ontologías es la secuencia específica ordenada de procesos y actividades a llevar a cabo durante la vida de la red de ontologías. La metodología NeOn propone hasta siete fases, con diferentes procesos asociados dependiendo de los escenarios elegidos como se puede comprobar en la Ilustración 45. Dentro de cada proceso además propone un organigrama con las actividades que lo compone. Las fases se pueden comprobar en la Tabla 5, y los diferentes procesos con sus escenarios en la Ilustración 45.

FASE	PROCESOS
Iniciación	Especificación de requisitos Planificación Evaluación
Reutilización	Reutilización de NOR Búsqueda de ontologías Reutilización de ontologías de dominio Reutilización de declaraciones ontológicas Evaluación de ontologías
Fusión	Alineación Evaluación
Reingeniería	Reingeniería de NOR Modularización Evaluación
Diseño	Conceptualización Evolución Localización Evaluación
Implementación	Evaluación
Mantenimiento	Evaluación

Tabla 5. Metodología NeOn, tabla de fases y procesos

NeOn propone dos modelos de ciclo de vida: el modelo de ciclo de vida en cascada y el modelo de ciclo de vida iterativo-incremental.

4.4.1. Modelo en cascada

Representa las etapas del desarrollo de la ontología como una cascada, donde se debe completar una etapa concreta antes de que comience la etapa siguiente y donde se permite retroceder desde la fase de mantenimiento hasta la fase que sigue a la de los requisitos. Debido a la importancia de la reutilización y la reingeniería de los recursos de conocimiento, así como la fusión de ontologías en el desarrollo de la red de ontologías, establece cinco versiones diferentes para este modelo Tabla 6, dependientes de la elección de escenarios realizada.

a) Modelo de cascada de 4 fases. Representa las etapas de una red de ontologías, comenzando con la fase de inicio y pasando por el diseño, la implementación y el mantenimiento.

b) Modelo de Cascada de 5 Fases. Extiende el modelo de 4 fases con la reutilización de los recursos ontológicos existentes sin modificarlos.

<p>c) Modelo en cascada de 5 fases + fase de fusión. Caso especial del modelo de 5 fases que incluye la Fase de Fusión para obtener un nuevo recurso ontológico a partir de dos o más recursos ontológicos previamente seleccionados en la fase de reutilización.</p>
<p>d) Modelo de Cascada de 6 Fases. Extiende el modelo de 5 fases con la Fase de Reingeniería. Permite la reingeniería de recursos de conocimiento (ontológicos y no ontológicos).</p>
<p>e) Modelo en cascada de 6 fases + fase de fusión. Extiende el modelo de 6 fases incluyendo la Fase de fusión después de la reingeniería de los recursos de conocimiento.</p>
<pre> graph TD A[Fase de Iniciación] --> B[Fase de Reutilización] B --> C[Fase de Fusión] C --> D[Fase de Reingeniería] D --> E[Fase de Diseño] E --> F[Fase de Implementación] F --> G[Fase de Mantenimiento] G --> B </pre>

Tabla 6. Metodología NeOn, familia de modelo en cascada

La suposición fundamental para utilizar uno de los modelos en cascada propuestos es que los requisitos son completamente conocidos, sin ambigüedades e inalterables al comienzo del desarrollo de la ontología. Este modelo se puede utilizar en las siguientes situaciones:

- En proyectos ontológicos de corta duración (por ejemplo, 2 meses).
- En proyectos de ontología cuyo objetivo sea desarrollar una ontología existente en un formalismo o lenguaje diferente.
- En proyectos de ontología en los que los requisitos son cerrados (por ejemplo, para implementar el contenido de una norma ISO).
- En proyectos de ontologías donde las ontologías deben representar un dominio pequeño y bien entendido

4.4.2. Modelo iterativo-incremental

Este modelo organiza el desarrollo de la ontología en un conjunto de iteraciones (o mini-proyectos cortos de duración fija). Cualquier iteración se programa como un proyecto de desarrollo de ontología que utiliza una de las versiones del modelo en cascada mostrados en la Tabla 6.

Se propone la sucesiva mejora y ampliación de la ontología mediante múltiples iteraciones con retroalimentación y adaptación, produciéndose un crecimiento incremental de la misma a lo largo del desarrollo. Se permiten requisitos nuevos o modificados en cada iteración, dependiendo del número de

iteraciones y del conocimiento inicial que tengamos de los requisitos del proyecto. El resultado de cualquier iteración en este modelo es una ontología que cumple con los requisitos identificados en la iteración. Este modelo no permite retroceder entre las fases de una iteración concreta y el refinamiento debe ser realizado en la siguiente iteración. Además, en la fase de inicio de cada iteración, se deben realizar revisiones de los requisitos y el cronograma de la ontología. Este modelo se recomienda en:

- proyectos con grandes grupos de desarrolladores en los que se plantean escenarios complejos, como la reingeniería de recursos no ontológicos o la alineación de recursos ontológicos
- proyectos en los que los requisitos no se conocen completamente al principio o pueden cambiar durante el desarrollo de la ontología
- proyectos en que los requisitos tienen diferentes prioridades

A la hora de elegir el número de fases asociadas a los modelos en cascada, ya sean en ciclo único o por interacción incremental, la metodología propone realizar una serie de preguntas para su resolución, Tabla 7.

1	¿Usará algún recurso no ontológico (NOR) en el desarrollo de su ontología?	6 fases
2	¿Utilizará algún recurso ontológico en el desarrollo de su ontología?	5 fases
3	¿Utilizará y modificará algún recurso ontológico en el desarrollo de su ontología?	6 fases
4	¿Utilizará y combinará un conjunto de recursos ontológicos en el desarrollo de su ontología?	5 fases + fase de fusión
5	¿Usará, fusionará y modificará un conjunto de recursos ontológicos en el desarrollo de su ontología?	6 fases + fase de fusión
6	¿Usará patrones de diseño de ontología en su desarrollo de ontología?	5 fases
7	¿Reestructurarás tu ontología?	4 fases
8	¿Desarrollará su ontología en diferentes lenguajes naturales?	4 fases

Tabla 7. Metodología NeOn, preguntas y versión de modelo

4.5. Pautas metodológicas

Todos los procesos y actividades se describen con plantillas, un flujo de trabajo y ejemplos según las necesidades. El desarrollo de una ontología o red de ontologías requiere cumplir una serie de fases, y cada fase tiene asociado un conjunto de procesos. Cada proceso consta de diferentes actividades, algunas de las cuales producen documentación y plantillas. En este apartado se describirán las plantillas y actividades asociadas a cada proceso con los esquemas extraídos de la metodología [42], [86]. Hay que tener en cuenta que dependiendo de la los escenarios elegidos para el desarrollo, no siempre se incluirán las mismas fases ni los mismos procesos en cada fase.

4.5.1. Especificación de requisitos

El objetivo de este proceso de la metodología, es definir el propósito de la ontología, cuáles son sus usos previstos y los finales, cuáles son los requisitos que debe cumplir y su nivel de formalidad. Para lograr este propósito se establecen ocho tareas que se muestran en la Ilustración 48:

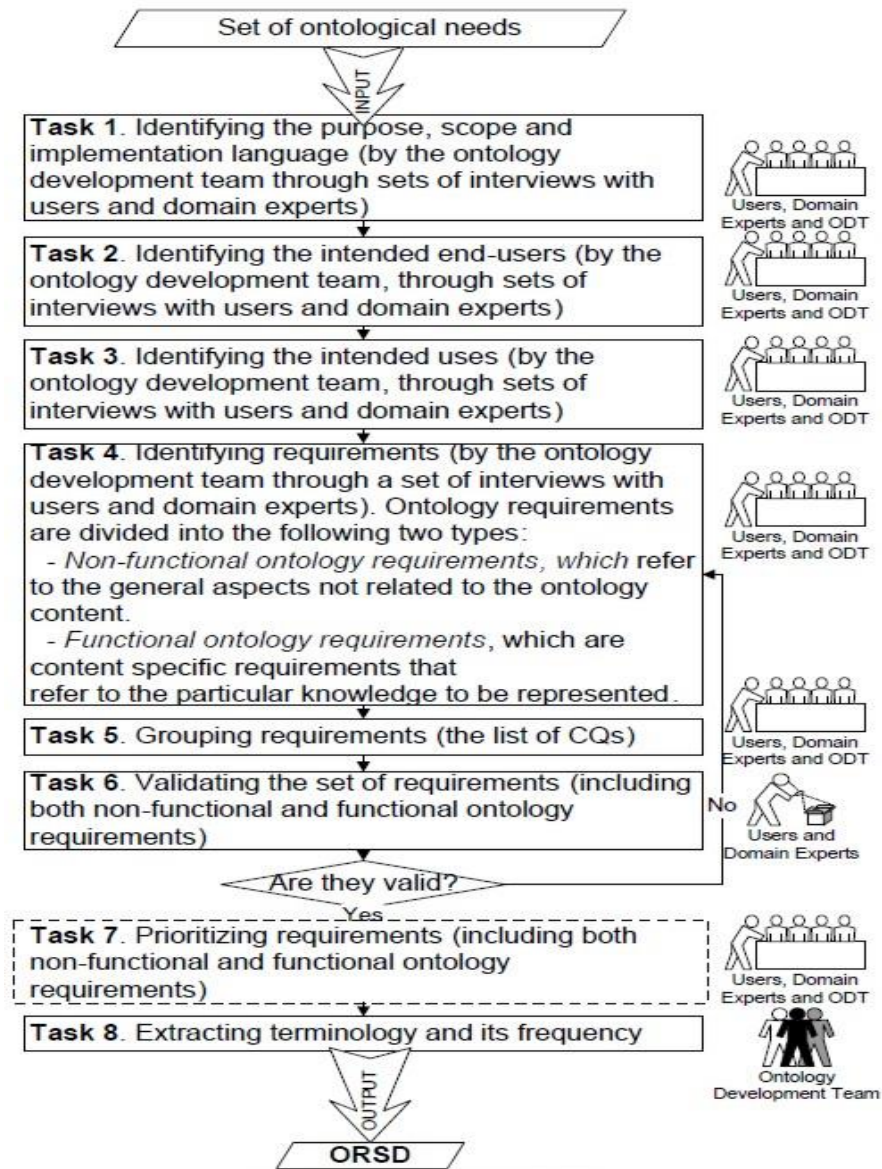


Ilustración 48. NeOn: tareas para la especificación de requisitos [86]

- **Tarea 1.** Identificar el propósito, alcance y lenguaje de implementación de la ontología. El equipo de desarrollo realizará una serie de entrevistas con usuarios y expertos del dominio.
- **Tarea 2.** Identificar los usuarios finales previstos. El equipo de desarrollo realizará una serie de entrevistas con usuarios y expertos del dominio.
- **Tarea 3.** Identificar los usos previstos. El equipo de desarrollo realizará una serie de entrevistas con usuarios y expertos del dominio.
- **Tarea 4.** Identificar los requisitos. El equipo de desarrollo realizará una serie de entrevistas con usuarios y expertos del dominio. Los requisitos se dividen en Requisitos No Funcionales que hacen referencia a las características, cualidades o aspectos generales no relacionados con el conocimiento representado en la ontología y los Requisitos Funcionales que se consideran requisitos específicos de contenido.
- **Tarea 5.** Agrupar requisitos funcionales. Los Requisitos Funcionales se representan mediante las denominadas Preguntas de Competencia (CQ's²⁴) sobre las que la metodología sugiere varias

²⁴ CQ: competence questions, preguntas que requieren respuestas con ejemplos de la vida real

herramientas y recomendaciones (aproximaciones *Top-Down*, *Botton-Up*, *Middle-Out*, herramientas para Mapas Mentales, etc.), siempre orientado a su escritura en lenguaje natural. Esta tarea consiste en agrupar la lista de CQs identificadas en la Tarea 4 en varias categorías conforme a los resultados de las entrevistas.

- **Tarea 6.** Validación de los requisitos. Se validarán los requisitos, funcionales y no funcionales mediante entrevistas con usuarios y expertos del dominio.
- **Tarea 7.** Priorización de los requisitos. Se establecen diferentes niveles de prioridad mediante entrevistas con usuarios y expertos del dominio.
- **Tarea 8.** Extracción de terminología y sus frecuencias. Es extraer de las CQ y de sus respuestas un glosario previo (documento pre-glosario) de términos como nombres, adjetivos y verbos.

El resultado final de recopilar la información obtenida en cada tarea se plasma en una plantilla denominada plantilla ORSD²⁵ que podemos consultar en la Ilustración 49. Los requisitos incluidos en el ORSD facilitan el desarrollo de ontologías de diferentes maneras:

- permitir la identificación de qué conocimiento particular debe estar representado en la ontología
- facilitar la reutilización de recursos de conocimiento mediante el enfoque de la búsqueda de recursos hacia el conocimiento particular a representar en la ontología
- permitir la verificación de la ontología con respecto a los requisitos que la ontología debe cumplir.

Ontology Requirements Specification Document Template	
1 Purpose	The main goal of the ontology. In other words, the main function or role that that the ontology should have.
2 Scope	The general coverage and the degree of detail that the ontology should have.
3 Implementation Language	The formal language that the ontology should use.
4 Intended End-Users	The intended end-users of the ontology.
5 Intended Uses	The intended uses of the ontology.
6 Ontology Requirements	
a. Non-Functional Requirements	The general requirements or aspects that the ontology should fulfil, including optionally priorities for each requirement.
b. Functional Requirements: Groups of Competency Questions	The content specific requirements that the ontology should fulfil in the form of groups of competency questions and their answers, including optional priorities for each group and for each competency question.
7 Pre-Glossary of Terms	
a. Terms from Competency Questions	The list of terms included in the competency questions and their frequencies.
b. Terms from Answers	The list of terms included in the answers and their frequencies.
c. Objects	The list of objects included in the competency questions and in their answers.

Ilustración 49. NeOn: modelo para la Plantilla ORSD [42]

²⁵ ORSD: *Ontology Requirements Specification Document*, Documento de Especificación de Requisitos de la Ontología

4.5.2. Planificación

La metodología NeOn nos propone dividir este proceso en cuatro tareas, Ilustración 50, con el objetivo de identificar los procesos y actividades a realizar, así el tiempo y recursos necesarios para su realización.

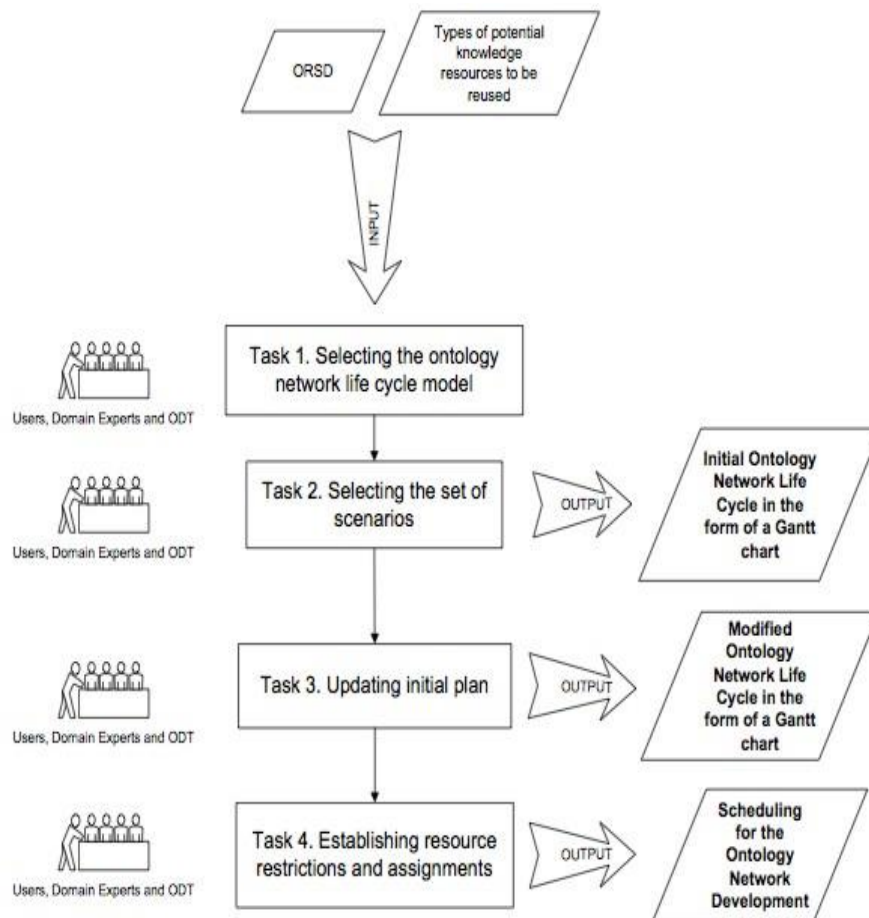


Ilustración 50. NeOn: tareas para el proceso de planificación [86]

- **Tarea 1.** Selección del modelo del ciclo de vida. Conforme a las preguntas de la Tabla 7 del Apartado 4.4, se asigna un modelo de ciclo de vida.
- **Tarea 2.** Selección del conjunto de escenarios. Al igual que en la tarea anterior, las preguntas de la Tabla 7 del Apartado 4.4 establecen el conjunto de escenarios a desarrollar. En esta tarea se integran los procesos de cada escenario en su correspondiente fase del ciclo de vida seleccionado, y se muestran en un gráfico como diagrama de Gantt como planificación inicial.
- **Tarea 3.** Actualización del plan inicial. El equipo de desarrollo realizará una serie de entrevistas con usuarios y expertos del dominio para contrastar la planificación inicial, e incluirá los cambios que puedan surgir sobre la misma en esta tarea.
- **Tarea 4.** Establecimiento de restricciones y asignación de recursos. En esta tarea se incluye cualquier información adicional a la planificación o asignación de recursos humanos en cualquier proceso o actividad.

4.5.3. Reutilización de recursos no ontológicos (NOR)

El objetivo de este proceso es identificar los recursos no ontológicos más apropiados para la construcción de la ontología. La Ilustración 51 nos presenta una serie de tareas y subtareas para alcanzarlo.

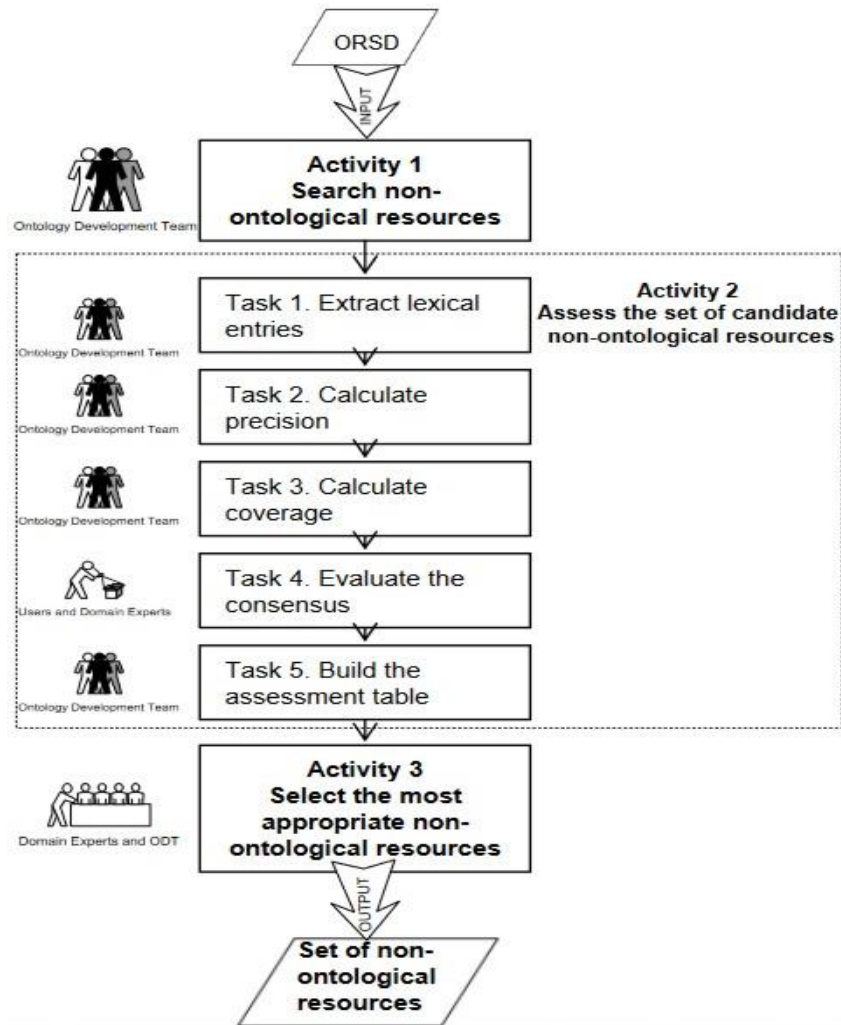


Ilustración 51. NeOn: tareas para la reutilización de recursos no ontológicos [86]

- **Tarea 1.** Búsqueda de recursos no ontológicos. Se procede a la búsqueda de recursos no basados en ontologías que ayuden a la representación del dominio y a la construcción de la ontología teniendo como base el ORSD.
- **Tarea 2.** Valoración de recursos no ontológicos. Esta tarea cubre la valoración del conjunto de recursos no ontológicos candidatos, resultado de la tarea anterior, es de vital importancia los análisis que figuran en el ORSD. Para alcanzar su objetivo se subdivide en cinco pasos:
 - 1) Extraer las entradas léxicas
 - 2) Calcular la precisión
 - 3) Calcular la cobertura
 - 4) Evaluar en consenso
 - 5) Construir la tabla de valoración
- **Tarea 3.** Selección de los recursos no ontológicos más apropiados. Se seleccionan los recursos no ontológicos más apropiados para la construcción de la ontología de entre los candidatos que surjan de la tarea anterior con el consenso entre el equipo de trabajo y los expertos del dominio.

4.5.4. Búsqueda de ontologías

En este proceso se realiza la actividad de localizar ontologías que pueden reutilizarse para construir la ontología o aplicación requerida. La Ilustración 52 define sus tareas:

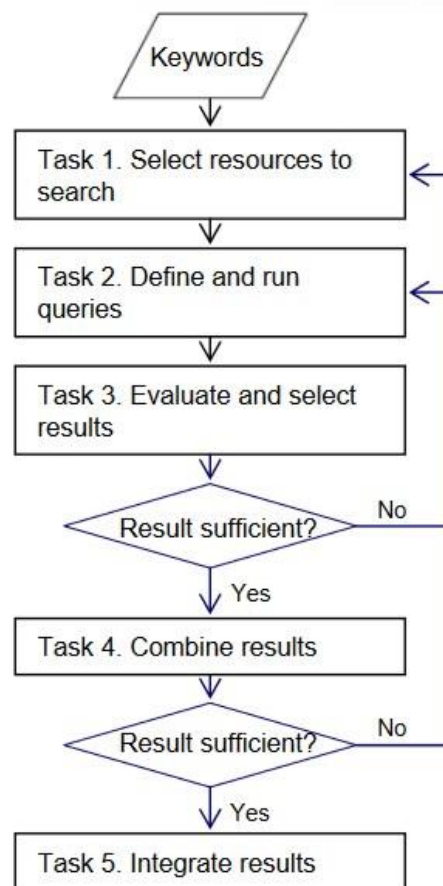


Ilustración 52. NeOn: tareas para la búsqueda de ontologías [42]

- **Tarea 1.** Selección de recursos de búsqueda. Para realizar el proceso de búsqueda de ontologías es necesario seleccionar un motor de búsqueda o repositorio adecuado, teniendo como prioridad aquellos repositorios con ontologías específicas para el dominio que se va a desarrollar.
- **Tarea 2.** Establecer y ejecutar las búsquedas. Se definen y ejecutan las consultas al motor de búsqueda o repositorios para obtener un conjunto de ontologías candidatas.
- **Tarea 3.** Evaluación y selección de resultados. La evaluación dependerá de la aplicación de destino de las ontologías de búsqueda. Se pueden emplear múltiples criterios como la cobertura, complejidad, corrección y grado de reutilización de las ontologías. Algunos repositorios incluyen mecanismos de evaluación para ayudar en esta tarea. Hay que tener en cuenta que también se pueden elegir partes de ontologías y no la ontología completa, en cuyo caso, el proceso de búsqueda de ontologías se relaciona con los de modularización de ontologías y reutilización de enunciados de ontologías (Apartado 4.5.10). Si los resultados seleccionados se consideran suficientes, el proceso puede continuar. De lo contrario, la consulta original se puede refinar para obtener otro conjunto de ontologías candidatas.
- **Tarea 4.** Unificación de resultados. En los casos en los que se hayan obtenido varios conjuntos de ontologías candidatas, tanto por haber realizado varias consultas, como por haber usado diferentes motores o repositorios, los resultados deben combinarse, teniendo especial cuidado en no caer en redundancias y mantener la coherencia.
- **Tarea 5.** Integración de resultados. Una vez seleccionado un conjunto válido de ontologías, o módulos de ontología, debe integrarse dentro de la ontología o la aplicación a desarrollar.

4.5.5. Reutilización de ontologías de dominio

Construir ontologías desde cero consume muchos recursos. La reutilización de ontologías de dominio reduce el tiempo y los costos en el proceso de desarrollo, difunde buenas prácticas cuando se reutilizan ontologías bien desarrolladas y aumenta la calidad general de los modelos ontológicos.

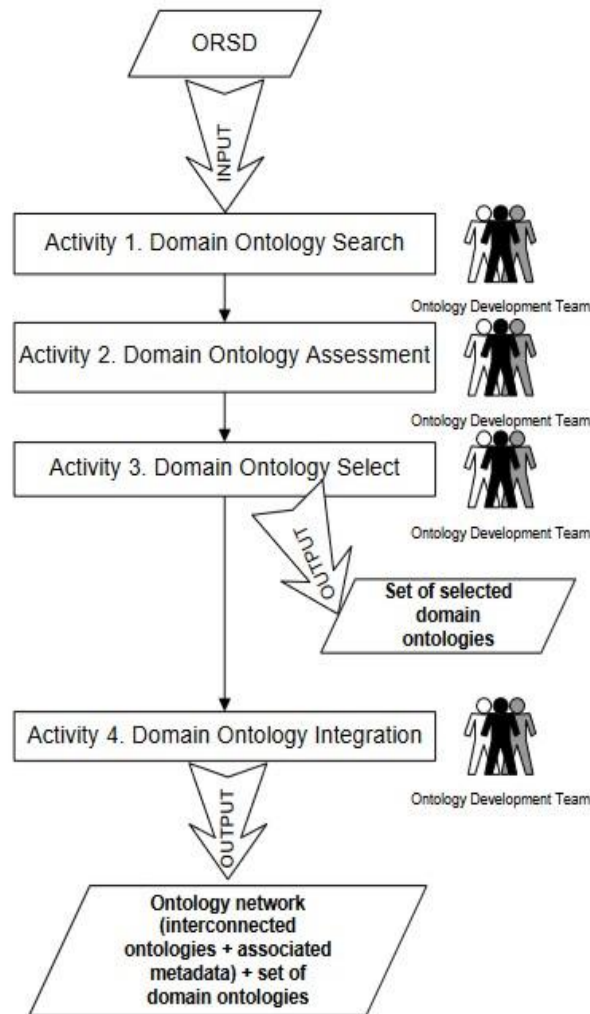


Ilustración 53. NeOn: reutilización de ontologías de dominio completas [42]

- **Actividad 1.** Búsqueda de ontologías de dominio. El objetivo de esta actividad es buscar en bibliotecas, repositorios y registros ontologías de dominio candidatas que puedan satisfacer las necesidades de la red de ontologías que se está desarrollando. El equipo de desarrollo debe tomar como entrada aquellos términos que tienen una alta frecuencia en el ORSD. El resultado de la actividad es un conjunto de ontologías candidatas. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.
- **Actividad 2.** Evaluación de Ontologías de Dominio. El equipo de desarrollo parte del conjunto de ontologías candidatas del apartado anterior y debe emplear los siguientes criterios de selección para decidir si una determinada ontología de dominio es útil o no:
 - comprobar si el alcance y finalidad establecidos en el ORSD son similares a los de las ontologías de dominio candidatas
 - consultar los requisitos de ontología funcional establecidos en la ORSD (por ejemplo, el lenguaje de implementación de la ontología, si los términos a emplear provienen de estándares, si debe incluir varios idiomas, etc.)

- comprobar los CQs incluidos en el ORSD respecto a las ontologías de dominio candidatas, teniendo en cuenta los niveles terminológicos y semánticos:
 - Nivel terminológico: el equipo de desarrollo debe calcular la precisión e inclusión de las ontologías del dominio candidato con respecto a la terminología incluida en los CQ
 - Nivel Semántico: el equipo de desarrollo debe comprobar si las ontologías de dominio candidatas son capaces de responder a los CQ incluidos en el ORSD

Como resultado de esta actividad se desarrolla una tabla de evaluación que analiza cada ontología de dominio candidata con respecto a los criterios mencionados. Para decidir que una ontología de dominio es útil, se debe satisfacer el conjunto de criterios relacionados con los requisitos de la ontología y los CQ. Las ontologías de dominio útiles deberán aparecer sombreadas en la tabla.

- **Actividad 3.** Selección de ontología de dominio. El equipo de desarrollo parte de las ontologías útiles de la tabla del apartado anterior y selecciona las más adecuadas según los siguientes criterios:
 - Comprensibilidad de recursos ontológicos: se verifica si la ontología del dominio tiene documentación precisa
 - Esfuerzo de modularización de recursos ontológicos: se comprueba si la ontología del dominio está bien modularizada
 - Esfuerzo de integración de recursos ontológicos: para comprobar si el esfuerzo de estimación para integrar la ontología del dominio es bajo y si la ontología del dominio utiliza convenciones de nomenclatura
 - Confiabilidad de recursos ontológicos: se comprueba si la ontología del dominio es reutilizada por otras ontologías u otros proyectos basados en ontologías, y si la ontología ha sido evaluada. En ese caso, las ontologías que satisfacen el mayor número de criterios se seleccionan en la tabla de selección sombreando su columna. El resultado de la actividad es un conjunto de ontologías de dominio seleccionadas (sombreadas) de la tabla de selección.
- **Actividad 4.** Integración de ontologías de dominio. El equipo de desarrollo realiza esta actividad tomando como entrada la tabla de selección de la actividad anterior, y para cada ontología incluida decide uno de los siguientes modos de integración:
 - La ontología de dominio seleccionada se reutiliza tal cual.
 - La ontología de dominio seleccionada se reutiliza con cambios significativos (por ejemplo, cambiando el lenguaje de implementación). En este caso, la actividad de reingeniería de recursos ontológicos debe realizarse con la ontología de dominio seleccionada y cambia el escenario a seguir por el escenario 4 con sus correspondientes actividades
 - Se fusionan varias ontologías de un mismo dominio para obtener una nueva ontología de dominio. En este caso, se debe seguir el escenario 5 o el escenario 6

Antes de reutilizar las ontologías de dominio seleccionadas siguiendo cualquier modo de reutilización, también es conveniente evaluar las ontologías de dominio a través de la actividad de evaluación de ontologías Apartado 4.5.7 . El resultado de la actividad es una ontología o red de ontologías que incluye el conjunto de ontologías de dominio seleccionadas.

4.5.6. Reutilización de declaraciones ontológicas

La reutilización de ontologías grandes o generales puede complicar un desarrollo al contener una gran cantidad de conocimiento que puede no ser necesario para el uso particular que se quiera. Este proceso describe cómo reutilizar fragmentos de conocimiento para integrarlos en la nueva ontología que se está construyendo en lugar de reutilizar toda la información de una ontología mayor.

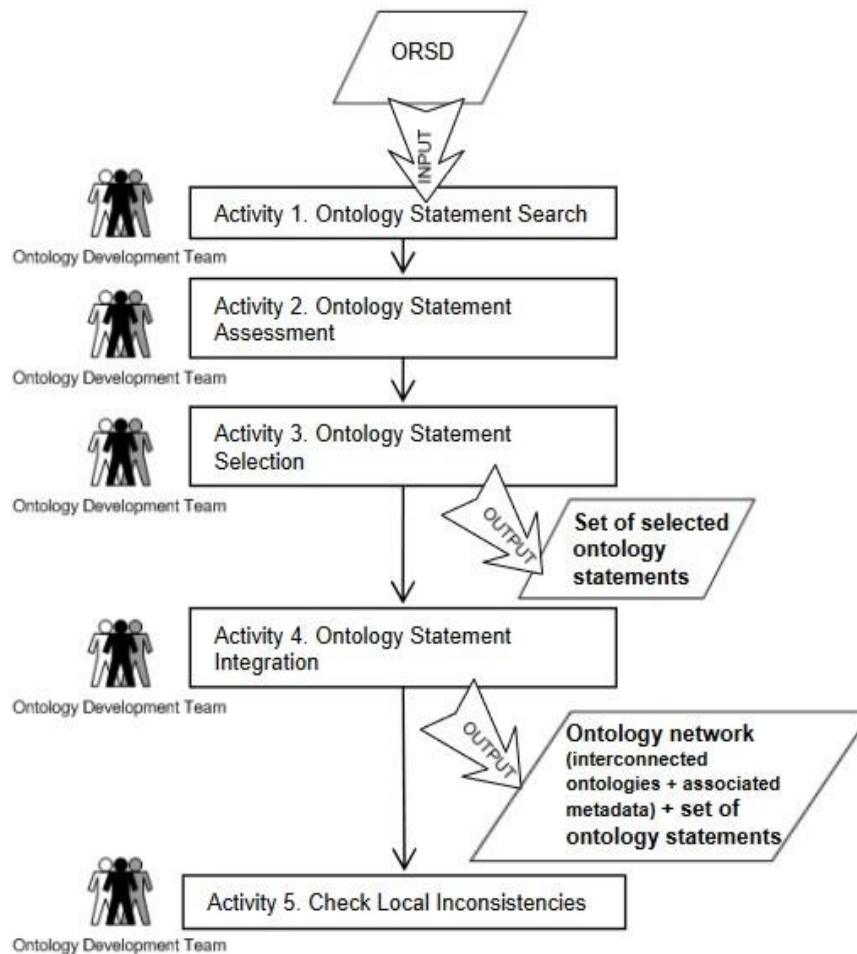


Ilustración 54. NeOn: reutilización de declaraciones ontológicas [42]

- **Actividad 1.** Búsqueda de enunciados ontológicos. Se trata de buscar, fundamentalmente en la web, candidatos a declaraciones de ontología que puedan satisfacer las necesidades ontológicas. Partiendo del ORSD el equipo de desarrollo debe buscar en concreto aquellos términos que tienen una alta frecuencia de aparición. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.
- **Actividad 2.** Evaluación del enunciado de la ontología. El equipo de desarrollo debe inspeccionar el contenido y la granularidad de las declaraciones de ontologías obtenidas de la actividad anterior, para evaluar si satisfacen las necesidades del desarrollo. Para esta labor se facilita un listado de criterios:
 - comprobar si el enunciado pertenece a una ontología de alcance similar a la ontología que se está desarrollando
 - verificar si la declaración pertenece a una ontología con un propósito similar a la ontología que se está desarrollando
 - comprobar la claridad del enunciado ontológico. Las declaraciones ambiguas no son útiles y no deben ser reutilizadas
 - comprobar el contenido de la información de la declaración. En algunos casos, las declaraciones recuperadas facilitan poca información adicional, por lo que no deben ser reutilizadas
 - Evaluar la corrección del enunciado desde la perspectiva del modelado formal:

- verificando que la denominación de los conceptos en la declaración de la ontología refleje el significado previsto de la declaración dado su contexto ontológico y tener cuidado con conceptos sinónimos. En estos casos, es importante cambiar el nombre de los conceptos para no dejar dudas sobre la declaración
- verificando si la declaración de la ontología no es inválida desde una perspectiva formal, por ejemplo, confundiendo las relaciones "subclassOf" con otras relaciones como las relaciones "partOf" o "relatedTo"
- **Actividad 3.** Selección de declaraciones de la ontología. Se decide de entre el conjunto de declaraciones obtenidas del apartado anterior, cuáles son las mejores o las más convenientes para la ontología que se está desarrollando. El equipo de desarrollo debe seleccionar aquellas sentencias que requieran un mínimo esfuerzo de integración.
- **Actividad 4.** Integración de enunciados de la ontología. El equipo de desarrollo decide cómo se integrarán las declaraciones de ontología obtenidas en la actividad anterior en la ontología o red de ontologías que se está desarrollando. Existen tres modos de integración:
 - las declaraciones se reutilizarán tal como están
 - las declaraciones serán rediseñadas
 - las declaraciones se fusionarán

Aparte de estos modos de integración, también tiene que decidir entre:

- Importación de las sentencias de ontología. La ventaja es que mantiene un vínculo con la ontología de la que se originó el enunciado. Como efecto secundario, otros elementos de dicha ontología pueden tener un impacto en la ontología que se está construyendo
- Copiar las declaraciones de la ontología. Al reproducir el enunciado, la copia asegura que no aparecerá ningún efecto secundario, es decir, solo se integra el enunciado mismo, pero se pierde el vínculo con la ontología original.
- Establecer mapeos con los enunciados ontológicos. Esto puede verse como una solución de compromiso en la que la declaración se copia primero en la ontología que se está construyendo y las entidades recién creadas se alinean con las entidades de la ontología original. De esta manera, se mantienen los vínculos entre estas ontologías y los efectos secundarios se pueden controlar más fácilmente.
- Después de integrar una declaración de ontología, se recomienda realizar el siguiente trabajo:
 - cambiar nombres (conceptos, propiedades) para adaptarlos a las convenciones de nomenclatura utilizadas en la red de ontologías que se está desarrollando
 - agregar rango en propiedades y cambiar cardinalidades
 - añadir restricciones.
- **Actividad 5.** Revisar inconsistencias locales. Se verifica si se han introducido inconsistencias locales en la ontología o red de ontologías a desarrollar al haber agregado nuevos conocimientos. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.

4.5.7. Evaluación de ontologías

La evaluación de ontologías es un tema importante de trabajo en el campo de la Web Semántica. El proceso de evaluación debe tenerse en cuenta durante todo el desarrollo de la ontología, en la reingeniería (para evaluar la calidad y corrección de la ontología obtenida) y durante la selección de la misma (para comparar su calidad de entre el conjunto de ontologías candidatas).

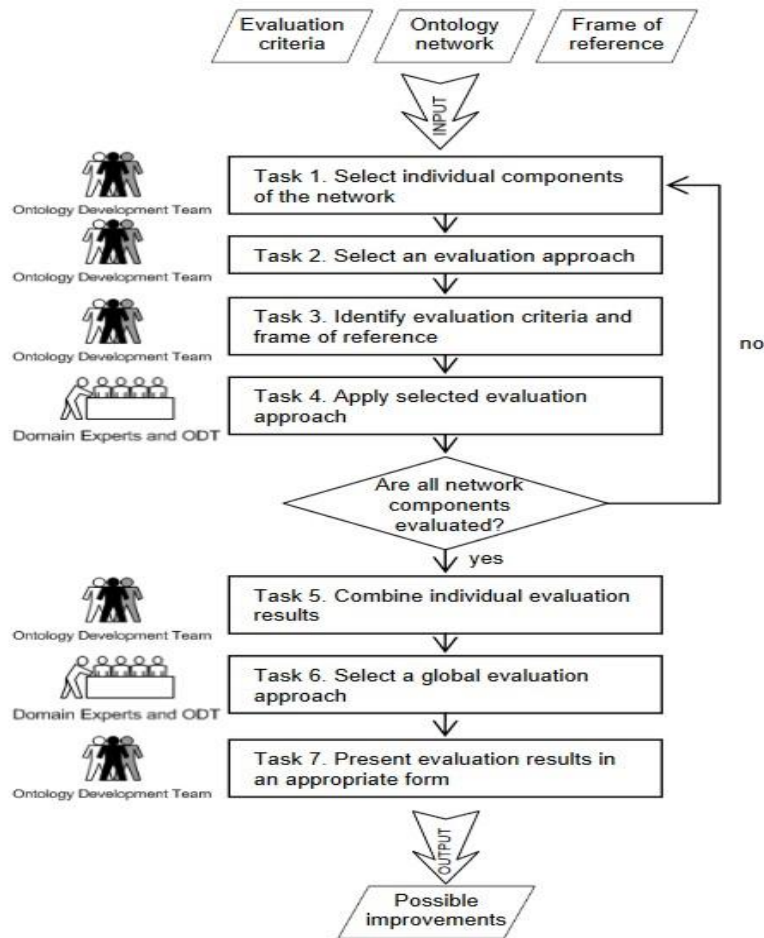


Ilustración 55. NeOn: evaluación de ontologías [42]

- **Tarea 1.** Selección de componentes individuales de la red de ontologías. En esta tarea, se identifica ontologías individuales dentro de la red, o pares de ontologías relacionadas y las asignaciones/alineaciones entre ellas, según dos criterios:
 - 1) qué ontologías y mapeos son críticos para la red general
 - 2) que se puede evaluar realmente: debe existir algún marco de referencia contra el que evaluar los componentes individuales
- **Tarea 2.** Selección de un enfoque de evaluación. Para evaluar ontologías individuales, los enfoques más comunes son:
 - comparar la ontología con una ontología estándar meritoria
 - utilizar la ontología en una aplicación y evaluar los resultados
 - comparar la ontología con una fuente de datos sobre el dominio a cubrir
 - evaluación por parte de expertos humanos de cómo la ontología cumple con los requisitos
 - evaluación en términos de patrones de diseño de ontologías.

Cuando se evalúan asignaciones y alineaciones entre ontologías, normalmente existen tres enfoques de evaluación:

 - 1) la evaluación abierta se realiza con alineaciones de referencia ya publicadas
 - 2) la evaluación ciega es realizada por evaluadores de alineaciones de referencia, desconocidos para los métodos utilizados para calcular las alineaciones
 - 3) la evaluación consensuada, cuando no existe un alineamiento de referencia estándar meritorio, se obtiene alcanzando consenso sobre los resultados encontrados por diferentes métodos.
- **Tarea 3.** Identificar criterios de evaluación y el marco de referencia. Dependiendo del enfoque de evaluación elegido, se debe especificar un marco de referencia para definir medidas de evaluación

adecuadas. El marco de referencia pueden ser otros recursos existentes (ontologías o alineaciones de ontologías de referencia), fuentes de datos sobre los que se realizaron mapeados ontológicos (por ejemplo, corpus de documentos), o expertos humanos sobre el dominio. Los criterios de evaluación corresponden a varias métricas que se pueden aplicar sobre las ontologías y mapeados basadas en costes, medida de ajuste entre una ontología, un mapeo y un corpus (conocimiento del dominio) así como medidas léxicas.

- **Tarea 4.** Aplicar el enfoque de evaluación seleccionado. Requiere una configuración adecuada para los experimentos de evaluación y la implementación de las herramientas de software para calcular las medidas de evaluación. Se puede complementar mediante el compromiso de los expertos humanos con sesiones de recopilación y evaluaciones no automatizadas.
- **Tarea 5.** Combinación de los resultados de evaluación individuales. En esta tarea se trata de destacar los puntos más débiles de la red de ontologías al considerar cada resultado de la evaluación individual y comprobar cómo afecta su integración en el resto de la red.
- **Tarea 6.** Selección de un enfoque de evaluación global. Al contrario que el punto anterior, se trata de ver cómo los resultados de la aplicación se ven afectados por el uso de la red de ontologías en cuestión. También se puede centrar en una evaluación desde el punto de vista del usuario individual o de la organización que utilizará la red de ontologías.
- **Tarea 7.** Presentación de los resultados de la evaluación. En la tarea final se presenten los resultados de la evaluación en forma apropiada para posibles arreglos (correcciones y adiciones), mejoras y para facilitar la evolución futura de la red de ontologías.

4.5.8. Alineación entre ontologías

El proceso de establecer alineaciones entre ontologías se denomina coincidencia de ontologías. El emparejamiento de ontologías ha sido el foco de mucha atención en los últimos años, sin embargo, se ha trabajado poco en el soporte metodológico para encontrar alineaciones. La metodología NeOn intenta cubrir este vacío, mediante el esquema de la Ilustración 56, ya que esto facilita:

- trabajar con módulos pequeños y autosuficientes en lugar de grandes ontologías monolíticas
- expresar los vínculos entre dos versiones de la misma ontología, facilitando la actualización de datos de una ontología a otra; o simplificando el reposicionamiento de una ontología en el contexto de otra de nivel superior

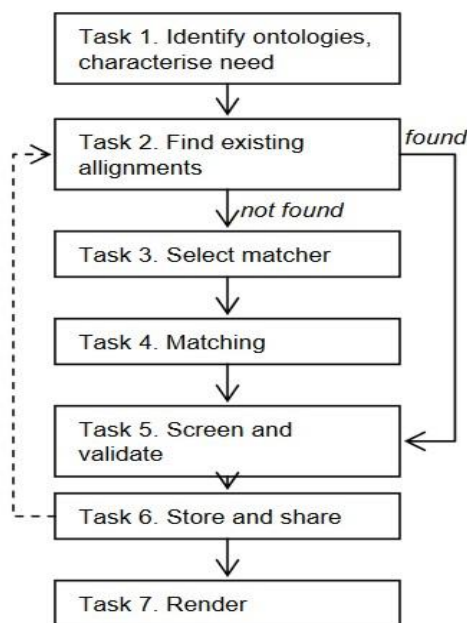


Ilustración 56. NeOn: tareas para la alineación entre ontologías [42]

- **Tarea 1.** Identificación de ontologías y caracterización de necesidades. A la hora de identificar ontologías para su alineación es importante distinguir si el objetivo es fusionar dos ontologías en un sistema basado en el conocimiento o agregar otra fuente de datos a un motor de consultas. En el primer caso la alineación debe ser perfecta para poder hacer inferencias correctas. En el segundo, otras fuentes pueden completar las respuestas faltantes. También se recomienda caracterizar el tipo de ontologías respondiendo a las siguientes preguntas: ¿Están etiquetadas en el mismo lenguaje natural? ¿Cuál es su expresividad? ¿Hay individuos relacionados con las ontologías disponibles?
- **Tarea 2.** Localización de alineaciones existentes. Encontrar alineaciones existentes que satisfagan las necesidades del desarrollo a ejecutar. Los alineamientos se pueden publicar directamente en la web o en servidores de alineamientos especializados. Idealmente, deberían venir con anotaciones que caractericen su nivel de confiabilidad, el propósito para el que han sido construidos y el tipo de relaciones que utilizan. Si se dispone de alineaciones aparentemente adecuadas, se puede pasar directamente al paso 5 de validación, de lo contrario, es necesario construir una nueva alineación.
- **Tarea 3.** Selección de un emparejador adecuado. Se debe seleccionadas una clase o propiedad en función de las características de las ontologías y de los alineamientos esperados, existen estudios que indican como buscarlas, aunque lo mejor es realizar diversas campañas de evaluación de emparejadores hasta localizar uno disponible que se adaptarse a la tarea a desarrollar.
- **Tarea 4.** Emparejamiento. Se trata de la realización de pruebas y consultas con el comparador contra las ontologías y recopilar los resultados para verificar la alineación resultante. Se recomienda ejecutar las pruebas del comparador varias veces o ejecutar pruebas con varios emparejadores, probando diferentes conjuntos de parámetros y diferentes umbrales. Puede ser útil probar los resultados con herramientas de verificación de consistencia. También es útil procesar la coincidencia de forma incremental refinando la alineación devuelta y enviándola nuevamente al comparador para mejorarla.
- **Tarea 5.** Proceso de criba y validación. Una vez que se ha obtenido una alineación satisfactoria, es necesario realizar una evaluación y validación final. La recomendación es hacerlo mediante un experto independiente que evalúe la calidad de la alineación y realice alguna edición manual.
- **Tarea 6.** Compartir y Almacenar. Un paso adicional es guardar la alineación obtenida en un formato declarativo para que pueda compartirse y aplicar anotaciones adecuadas para registrar su procedencia y propósito. Esto ayudará a otros desarrolladores para reutilizarlo.
- **Tarea 7.** Traducción o adaptación de formato. Aunque el formato o el lenguaje original sea otro, una vez realizada la alineación, se puede adaptar o guardar en otro formato si se ajusta mejor al su uso esperado.

4.5.9. Reingeniería de recursos no ontológicos (NOR)

Un recurso no ontológico (NOR) es un recurso perteneciente al ámbito de conocimiento del dominio (glosario, léxico, tesauro, esquema de clasificación, etc.) cuya semántica aún no ha sido formalizada por una ontología. Los NOR son muy heterogéneos en su modelo de datos y contenido: codifican diferentes tipos de conocimiento y pueden modelarse e implementarse de diferentes maneras. Cuando los desarrolladores de ontologías crean una ontología a partir de un esquema de clasificación, tesauro, etc., normalmente utilizan algoritmos ad-hoc para la transformación. La metodología NeOn propone el uso de Patrones de Reingeniería para transformar tales recursos en ontologías siempre que sea posible frente a una solución ad-hoc.

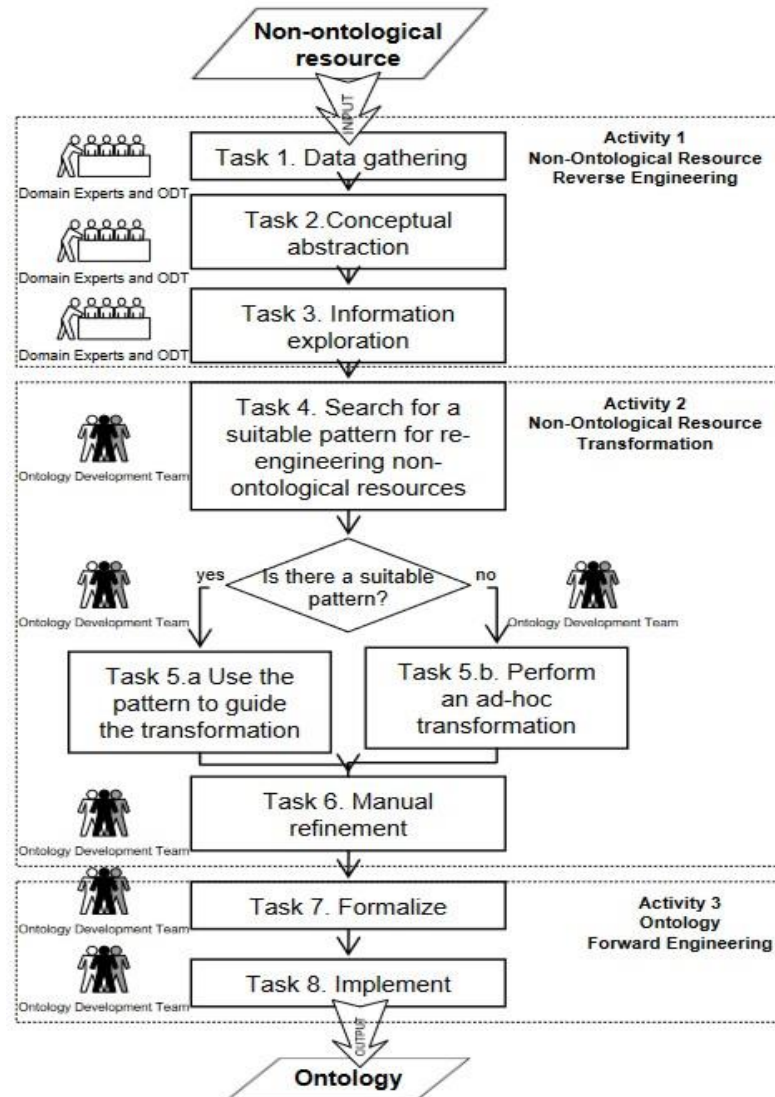


Ilustración 57. NeOn: reingeniería de recursos no ontológicos [42]

- **Actividad 1.** Ingeniería inversa de recursos no ontológicos. Para desarrollar esta actividad se deben buscar y recopilar todos los datos y documentación disponibles sobre aquellos recursos que no son ontologías o declaraciones ontológicas (**Tarea 1**). Una vez localizados y recopilados, se debe identificar el esquema y el modelo de datos asociado a cada recurso o conjunto de recursos (**Tarea 2**). Por último, una vez localizados los componentes subyacentes, se deben crear representaciones de cada recurso en los diferentes niveles de abstracción: diseño, requisitos y conceptual (**Tarea 3**).
- **Actividad 2.** Transformación de recursos no ontológicos. Hay que generar un modelo conceptual a partir de los recursos no ontológico obtenidos de la actividad anterior. Para ello, se debe averiguar si existe algún patrón de reingeniería aplicable, ver Apartado 4.6, dentro de la biblioteca de recursos PR-NOR para transformar el recurso en un modelo conceptual. El criterio de búsqueda ha de ser el siguiente:
 - por tipo de recurso no ontológico
 - por modelo de datos del recurso
 - por el enfoque de transformación

Si se encuentra un patrón adecuado para la reingeniería del recurso no ontológico, entonces se crea el modelo conceptual siguiendo el procedimiento establecido en el patrón para la reingeniería.

De lo contrario, tenemos que establecer un procedimiento ad-hoc para transformar el recurso no ontológico en un modelo conceptual. Este procedimiento ad-hoc puede generalizarse para crear un nuevo patrón para la reingeniería de recursos no ontológicos.

- **Actividad 3.** Ingeniería hacia adelante de la ontología. Su objetivo es generar la ontología o recursos de la misma, unificando los recursos generados con el proceso de desarrollo general de la ontología (Escenario 1). La inclusión de recursos en la ontología se realiza mediante 3 patrones definidos en la biblioteca de recursos PR-NOR que se pueden consultar en el Apartado 4.6, Ilustración 62.

4.5.10. Modularización

Las grandes ontologías monolíticas son difíciles de manejar y mantener, la tarea de modularizar una ontología consiste en identificar componentes (módulos) que se pueden considerar por separado, y mantener sus interrelaciones con otros componentes identificados. Los beneficios que se pueden obtener son, entre otros: mejorar del rendimiento (facilita la gestión distribuida, el procesamiento más focalizado), facilitar el desarrollo y mantenimiento al contar con componentes autónomos acoplados libremente, o facilitar la reutilización en subdominios o partes más ligeras.

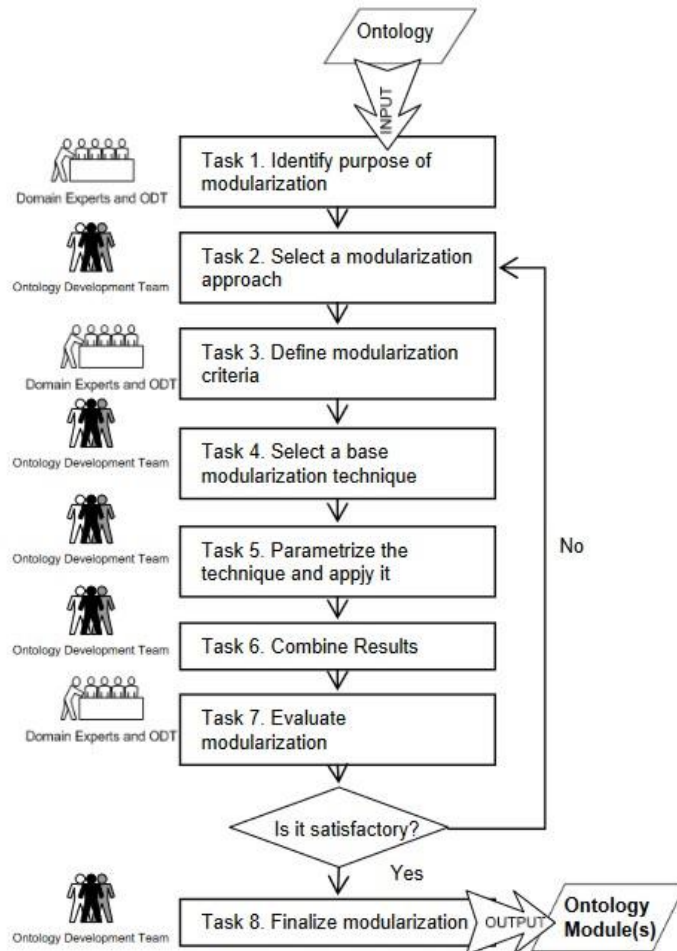


Ilustración 58. NeOn: modularización de ontologías [42]

- **Tarea 1.** Identificar el propósito de la modularización. El resultado de esta tarea es una descripción clara del escenario de aplicación para el que se realizará la modularización, definir qué módulos serán utilizados y el beneficio esperado del enfoque modular. Identificar el propósito de la modularización es crítico a la hora de seleccionar la técnica y los criterios de modularización apropiados para maximizar el beneficio esperado.

- **Tarea 2.** Selección del enfoque de modularización. Se distinguen dos tipos diferentes de modularización de ontologías:
 - por partición de ontología: es la descomposición automática de una ontología en una serie de módulos interrelacionados que cubren todos juntos la ontología completa
 - por extracción de módulo: corresponde a la creación de un módulo a partir de una subparte de la ontología que es específicamente relevante para un subdominio de la ontología

En general, es fácil decidir qué ontología elegir según el propósito de la modularización:

- Siempre que el propósito se relacione con toda la ontología (es decir, mejorar el mantenimiento y, en algunos casos, el rendimiento), se debe considerar un enfoque de partición.
- Siempre que el propósito se relacione con la extracción de partes específicas de una ontología (por ejemplo, para personalizarla o reutilizarla parcialmente), se debe considerar la extracción de módulos.

Puede ocurrir que los dos enfoques se combinen, extrayendo, por ejemplo, módulos del resultado de una técnica de partición, ya que se recomienda realizar la tarea aplicando un refinamiento iterativo.

- **Tarea 3.** Definición de criterios de modularización. Se definen las características básicas que deben contener los módulos resultantes. Ejemplos de criterios típicamente empleados son la integridad lógica y corrección con respecto a la ontología original, tamaño, relación entre módulos, etc., siempre deben decidirse en función del propósito de la modularización. Por ejemplo, si el objetivo es mejorar el procedimiento de razonamiento, se deben aplicar criterios lógicos. La elección de los criterios correctos dependerá del equipo de desarrollo y del propósito de la aplicación a realizar.
- **Tarea 4.** Selección de una técnica de modularización base. Existe una gran variedad de técnicas y herramientas para la modularización de ontologías y, por lo tanto, no existe una definición universal de lo que debería contener un módulo de ontología. Es necesario seleccionar la técnica más adecuada en función de los criterios a aplicar: la metodología propone recursos de apoyo que se pueden consultar, ver Apartado 4.6.
- **Tarea 5.** Parametrizar la técnica y aplicarla. Dependiendo de la técnica seleccionada en la tarea anterior, pueden ser necesarios varios parámetros para obtener resultados interesantes y útiles. Por ejemplo, los métodos basados en el análisis de grafos suelen utilizar parámetros como la densidad de interconexión entre entidades o el nivel de recurrencia al que se debe atravesar el gráfico.
- **Tarea 6.** Combinación de resultados. Se favorece un proceso iterativo donde los módulos adecuados se producen refinando y combinando los resultados obtenidos con varios parámetros, técnicas y enfoques. En cada iteración, cada vez que se produce un nuevo módulo (o módulos), es necesario integrarlo con los módulos de iteraciones anteriores. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.
- **Tarea 7.** Evaluación del resultado. La evaluación del conjunto completo de módulos generados es una parte crucial del proceso iterativo ya que la finalización del mismo depende de evaluar si el módulo (o módulos) obtenido es satisfactorio considerando el escenario de aplicación. Hay dos formas de evaluar la modularización: verificando los criterios o probando contra el propósito de la modularización.
- **Tarea 8.** Finalización de la modularización. Cuando la modularización producida ha resultado satisfactoria, se puede requerir un paso adicional para implementarla y explotarla en una aplicación. Se considera recomendable, incluso necesario, revisar los identificadores de cada uno de los módulos para que sigan las convenciones empleadas en la aplicación de destino, restablecer vínculos entre módulos o implementar los módulos restantes, y no solamente los

seleccionados, de manera que sean accesibles por la aplicación de destino o estén disponibles para futuras necesidades.

4.5.11. Conceptualización: utilización de patrones de diseño de ontologías

La aplicación de patrones se ha aplicado con gran éxito en diversas áreas como la arquitectura y la ingeniería de software. NeOn propone el uso de patrones, *Ontology Design Patterns* (ODPs) como base para la reutilización en el diseño de ontologías, y los acompaña de un marco teórico y métodos para su aplicación (ver Apartado 4.6 sobre otros recursos).

Un ODP es una solución de modelado para resolver un problema de diseño recurrente mediante una plantilla que representa un esquema con la solución de diseño específica. Se representan con un conjunto de entidades ontológicas prototipo que constituyen la “forma abstracta” de un patrón; y se adjunta un conjunto de metadatos sobre sus casos de uso, motivaciones, procedencia, pros y contras de su aplicación, los enlaces a otros patrones, etc. Tipos de patrón ODP:

- Estructurales: incluyen ODPs Lógicos (composiciones de construcciones lógicas que resuelven un problema de expresividad) y ODPs Arquitectónicos (composición de OPs Lógicos).
- De correspondencia: incluye ODP de reingeniería, para proporcionar a los diseñadores soluciones al problema de transformar un modelo conceptual, y ODP de mapeo, para crear asociaciones semánticas entre ontologías.
- Razonamiento ODPs: aplicaciones de OPs Lógicos orientadas a obtener determinados resultados de razonamiento, basados en el comportamiento implementado en un motor de razonamiento.
- Presentación ODPs: tratan la usabilidad y legibilidad de las ontologías desde la perspectiva del usuario.
- PAO léxico-sintácticos: estructuras o esquemas lingüísticos que permiten generalizar y extraer algunas conclusiones sobre el significado que expresan.
- ODP de contenido: codifica patrones de diseño conceptuales en lugar de lógicos y propone patrones para resolver problemas de diseño para las clases de dominio y las propiedades que pueblan una ontología.

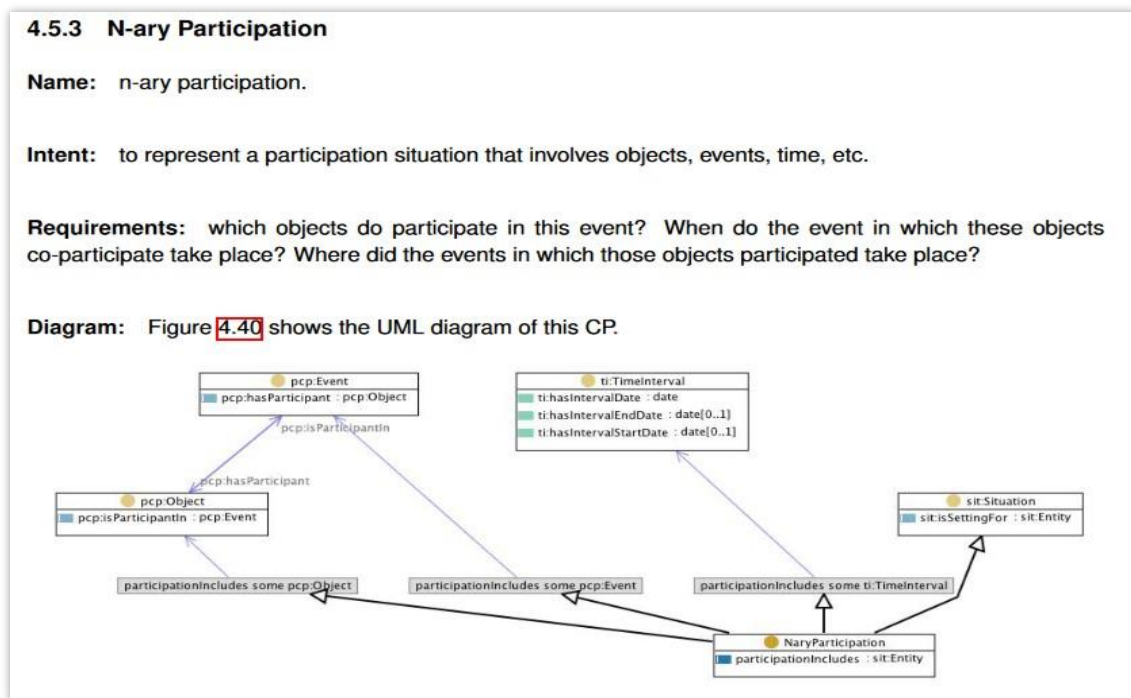


Ilustración 59. NeOn: Ejemplo de patrón conceptual de participación n-aria [88]

4.5.12. Evolución

Las ontologías son entidades dinámicas que evolucionan con el tiempo. Los dominios no son fijos y la conceptualización o la especificación formal pueden cambiar. El manejo de esta dinámica va desde el control adecuado de los cambios, hasta la administración de versiones de ontología. La evolución de la ontología se enfoca en la modificación de una ontología preservando su consistencia, mientras que el control de versiones de la ontología se enfoca en crear y administrar diferentes versiones de la misma.

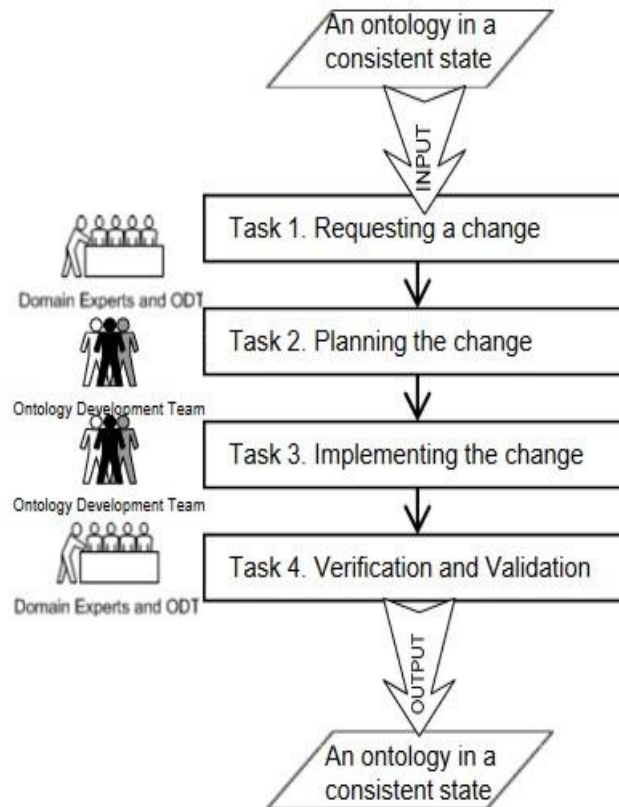


Ilustración 60. NeOn: proceso de evolución de la ontología [42]

- **Tarea 1.** Solicitar un cambio. En esta tarea se recogen propuestas de cambio por parte de los usuarios o desarrolladores de la ontología, o incluso el descubrimiento de cambios por modificación de las fuentes de datos y modelos externos a la aplicación desarrollada, pero que modifican el dominio o requisitos inicial. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6, incluso algunas de ellas permiten detectar cambios externos mediante técnicas de aprendizaje automático.
- **Tarea 2.** Planificación del cambio. Cuando se decide modificar o eliminar un elemento de la ontología, hay que tener en cuenta sus dependencias con la ontología existente y con otros artefactos relacionados, como instancias, mapeos, aplicaciones y metadatos. Esas dependencias se consideran mediante un análisis del impacto y el coste en la tarea. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6 que le permite verificar el costo de implementar el cambio.
- **Tarea 3.** Implementar el cambio. Se actualizan todos los artefactos relacionados con la ontología (si es necesario), asegurando la consistencia de la ontología y propagando dichos cambios por la red de ontologías si existe y es necesario. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.
- **Tarea 4.** Verificación y validación. Se trata de evaluar la corrección de la ontología, también se tienen que tener en cuenta los artefactos relacionados para garantizar que la ontología o la red de

ontologías se comporte como se esperaba (mantiene su consistencia). Cualquier conflicto que pueda surgir debe detectarse y esto puede afectar, por ejemplo, la decisión de si se debe implementar definitivamente el cambio. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.

4.5.13. Localización (adaptación lingüística y cultural)

El objetivo de este proceso es hacer frente a la demanda de aplicaciones basadas en ontologías que admitan el multilingüismo en tareas de recuperación de información, respuesta a preguntas o la gestión del conocimiento.

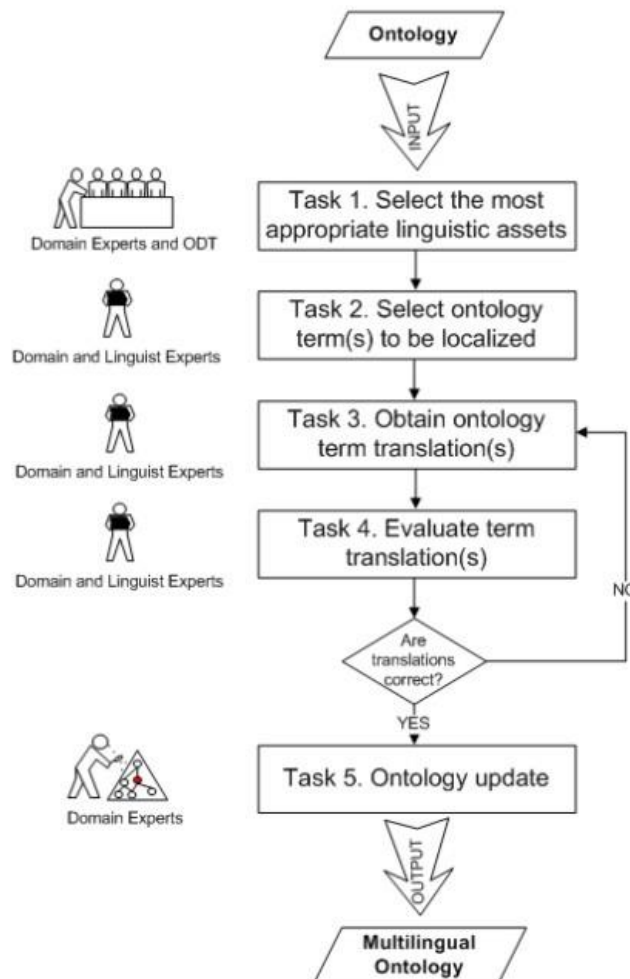


Ilustración 61. NeOn: adaptación lingüística y cultural ontológica [42]

- **Tarea 1.** Seleccionar los activos lingüísticos más adecuados. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.
- **Tarea 2.** Selección de los términos ontológicos a localizar. Hay que seleccionar y clasificar las etiquetas que deben traducirse. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.
- **Tarea 3.** Obtención de traducción de términos ontológicos. Se busca obtener la traducción más adecuada para cada etiqueta de ontología, teniendo especial cuidado con las diferentes interpretaciones y sinónimos que se pueden dar entre idiomas y culturas. Es importante llevar a cabo un proceso de desambiguación de sentido de palabra. Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.

- **Tarea 4.** Evaluación de la traducción de términos. Se debe revisar y validar cada traducción de etiquetas, y verificar tanto su fidelidad semántica (si las etiquetas representan términos conceptualmente equivalentes), como su corrección estilística (si la sintaxis y el estilo son equivalentes). Existen herramientas y complementos de apoyo desarrolladas por el equipo de NeOn: ver Apartado 4.6.
- **Tarea 5.** Actualización de la ontología. La ontología se actualiza con los datos lingüísticos resultantes.

4.6. Otros recursos

Además de los recursos fundamentales vistos hasta ahora, la metodología NeOn incluye una serie de recursos adicionales que sirven como apoyo o consulta [42].

4.6.1. NeOn Toolkit

Consiste en un software de apoyo desarrollado a modo de *framework* al que se pueden añadir una amplia colección de plugin o complementos, con un alto respaldo entre universidades europeas. Sirve para administrar específicamente algunas de las actividades que caracterizan el proceso de ingeniería de ontología y que se han indicado en alguna de las tareas o actividades. Se puede descargar y consultar tanto el programa como sus módulos en la web oficial de la metodología:

http://neon-toolkit.org/wiki/Neon_Plugins.html

4.6.2. Biblioteca para ODP

Se trata de una biblioteca de patrones de diseño de ontologías, disponible en documento [88] y en la web de la metodología:

<http://ontologydesignpatterns.org/>

y una guía metodológica compatible con herramientas asociadas (*eXtreme Design*) para ayudar en el desarrollo de ontologías. Los patrones de diseño de ontologías proporcionan soluciones de modelado que se pueden aplicar para resolver problemas recurrentes de diseño, se ha realizado un desarrollo de este concepto en el Apartado 4.5.11.

4.6.3. Biblioteca para reingeniería de PR-NOR

Se incluye junto con los patrones de diseño, proporciona una secuencia bien definida de actividades para transformar estos recursos (tesauros, esquemas de clasificación, etc.) en ontologías, teniendo en cuenta el tipo de recurso y su modelo de datos subyacente. Las ventajas que proporciona el uso de los patrones PR-NOR se pueden resumir en:

- Mejora de la eficiencia del proceso de reingeniería.
- Facilitar el proceso de transformación tanto para los ingenieros de ontologías como para los expertos en dominios.
- Mejora de la reutilización de NORs.

Estos patrones realizan las siguientes transformaciones:

- **Transformación TBox:** para transformar el recurso contenido en un esquema de ontología
- **Transformación ABox:** para transformar el recurso esquema en un esquema de ontología, y el recurso contenido en instancias de ontología

- **Población:** para transformar el recurso contenido en instancias de ontología

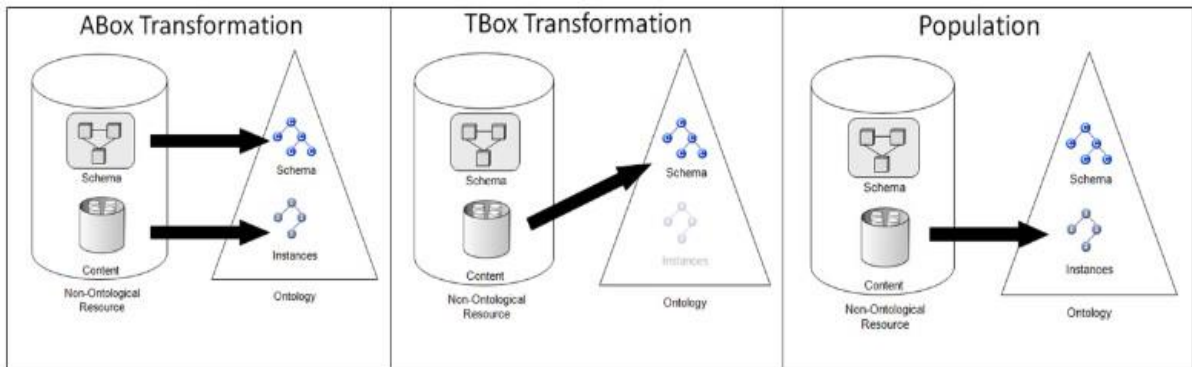


Ilustración 62. NeOn, transformaciones NOR en ontologías [89]

Capítulo 5

5. Plan de proyecto

Este proyecto se ha desarrollado siguiendo la metodología NeOn, al contener gran cantidad de documentación y ejemplos claros, tener un seguimiento actualizado y por su enfoque práctico. En todo momento del ciclo de vida asociado a un desarrollo ontológico establece pautas y procedimientos bien definidos. En este capítulo se comenzará a desarrollar el escenario 1 descrito en el tema anterior y de obligado cumplimiento, ya que se establecen los pasos necesarios para elaborar el plan de proyecto con los procesos de “Especificación de requisitos” y “Planificación”, ver Apartado 4.5.1 y 4.5.2 respectivamente.

5.1. Especificación de requisitos

Establece el objetivo de definir el propósito de la ontología para la aplicación a desarrollar, cuáles son sus usos previstos y los finales, cuáles son los requisitos que debe cumplir y su nivel de formalidad. Para lograr este propósito se establecen ocho tareas que se muestran en la Ilustración 48 del tema anterior. El resultado final de recopilar la información obtenida en cada tarea se plasma en la plantilla ORSD que será el artefacto de partida para el resto de procesos en el desarrollo de la ontología.

5.1.1. Tarea 1. Identificar el propósito, alcance y lenguaje de implementación

En el Apartado 1.2.1 de la memoria se describen cuáles son los objetivos y el alcance de la ontología: se pretende modelizar mediante SAREF el sistema de climatización AHU-101 del Alice Perry Engineering Building de la NUI en Galway, Irlanda. Se da importancia a la correcta modelización del dispositivo y queda fuera de objetivo la inferencia de conocimiento. Para la implementación de la ontología se ha empleado el lenguaje OWL-DL, ya que es la recomendación del W3C y el estándar SAREF.

5.1.2. Tarea 2. Identificar los usuarios finales previstos

En este apartado se identifican los diferentes actores que interactúan con la ontología. Un actor representa un conjunto de roles que juegan los usuarios al interactuar con nuestro sistema. El rol suelen desempeñarlo tanto personas físicas como dispositivos u otros sistemas, igualmente una misma persona, dispositivo o sistema puede representar varios roles. De los objetivos descritos para la ontología se deducen dos actores:

- un usuario general (actor principal) con acceso a los documentos RDF finales y las gráficas
- un usuario técnico (actor secundario) que interactúa con el sistema generando los datos

5.1.3. Tarea 3. Identificar los usos previstos.

En la Tabla 8 y Tabla 9 se muestran los diferentes actores identificados en la tarea anterior con sus roles y usos previstos a la hora de interactuar con nuestro sistema.

Actor	Usuario General (Actor Principal)				ACT-01	
Descripción	Usuario genérico que visualiza documentos RDF y gráficas					
Características	No requiere ningún registro No tiene acceso al proceso de generación de datos					
Relaciones	Ninguna					
Autor	Ángel Nava	Fecha	10/11/2021	Versión	1.0	

Tabla 8. Actor Usuario General

Actor	Usuario Técnico (Actor Secundario)				ACT-02	
Descripción	interactúa con el sistema para obtener y mantener los datos					
Características	Carga ficheros CSV para ejecutar script Accede y mantiene los datos generados en la BD					
Relaciones	Ninguna					
Autor	Ángel Nava	Fecha	10/11/2021	Versión	1.0	

Tabla 9. Actor Usuario Técnico

5.1.4. Tarea 4. Identificar los requisitos

Los Requisitos No Funcionales (NRF²⁶) hacen referencia a las características, cualidades o aspectos generales no relacionados con el conocimiento representado en la ontología. Los Requisitos Funcionales (RF²⁷) se consideran requisitos específicos de contenido y se representan mediante las denominadas Preguntas de Competencia (CQ's²⁸) sobre las que la metodología sugiere varias herramientas y recomendaciones (aproximaciones *Top-Down*, *Bottom-Up*, *Middle-Out*, herramientas para Mapas Mentales, etc.), siempre orientado a su escritura en lenguaje natural.

La obtención de los requisitos se llevará a cabo con los objetivos generales plasmados en el Apartado 1.2.1, una serie histórica de datos en archivos CSV junto con un excel con descripciones sobre los sensores y las características de las medidas que hay que mapear en la ontología, así como reuniones con los tutores. Con respecto los requisitos funcionales, las CQ's se centrarán en lo que se quiere que responda la ontología cuyo dominio es la AHU101: dispositivos, características y mediciones o cambios que realizan, dónde se localizan, sus nombres, unidades de medida de los datos.

La metodología establece en este punto un ciclo iterativo de refinamientos: tareas 4 a la 6, hasta la obtención de una tabla con los requerimientos válidos.

5.1.5. Tarea 5. Agrupar requisitos funcionales

Se trata de agrupar la lista de CQs identificadas en la Tarea 4 en varias categorías. El listado de CQ's obtenidas es fácilmente clasificable ya que, por un lado, tenemos los dispositivos y por otro las medidas o cambios que realizan los mismos. De esta forma obtenemos tres ítems por el tipo de dispositivo que aparece en el mapeo de datos: "SENSOR", "ACCIONADOR" e "INTERRUPTOR" y, otro ítem para las medidas o cambios realizados por los mismos: "MEDICIÓN".

²⁶ NRF: Non-functional requirement, https://en.wikipedia.org/wiki/Non-functional_requirement

²⁷ RF: functional requirement, https://en.wikipedia.org/wiki/Functional_requirement

²⁸ CQ: competence questions, preguntas que requieren respuestas con ejemplos de la vida real

5.1.6. Tarea 6. Validación de los requisitos

Con los usuarios finales y expertos en el dominio se ha realizado un análisis de las preguntas de competencia para verificar su validez. Estas consultas se han llevado a cabo mediante reuniones con los tutores y responsables de departamento que ya han realizado otros desarrollos sobre el mismo dominio. Los requisitos fundamentales presentados se muestran en la Tabla 10 y en la Tabla 11 de manera parcial, encontrándose el listado completo y ya optimizado en el ORSD de la Tabla 12.

Requisitos no funcionales
RNF-01. La ontología debe reutilizar un modelo conceptual ya definido
RNF-02. La ontología en lo posible, debe reutilizar recursos ontológicos existentes
RNF-03. La ontología debe garantizar la representación fiable de los datos
RNF-04. La ontología debe permitir realizar búsquedas y estadísticas descriptivas básicas visibles mediante gráficas
RNF-05. La ontología en la medida de lo posible, debe representar todos los elementos y sensores
RNF-06. La ontología, en lo posible, debe ser extensible, escalable y exportable
RNF-07. La ontología debe mantener separados el modelo de los datos
RNF-08. La ontología debe garantizar la identificación de cada medida con su sensor
RNF-09. La ontología debe garantizar la identificación de la jerarquía de sensores
RNF-10. La ontología debe garantizar la identificación estructural de cada elemento de la AHU

Tabla 10. Requisitos no funcionales antes de su validación

Requisitos funcionales: Grupos de preguntas de competencia
<ul style="list-style-type: none"> ▪ Preguntas de competencia SENSOR (sensor)
SN_PC01 ¿Qué tipo de sensor es el sensor?
Respuesta: Temperature sensor Humidity sensor Smoke sensor
SN_PC02 ¿Qué tarea realiza el sensor?
Respuesta: Meter reading Safety Comfort
SN_PC03 ¿Qué tipo de tarea realiza el sensor?
Respuesta: Cleaning Comfort Drying Meter reading Safety WellBeing EnergyEfficiency
SN_PC04 ¿Qué tipo de propiedad controla el sensor?
Respuesta: Status Smoke Temperature Occupancy Humidity Time
SN_PC05 ¿Qué propiedad controla el sensor?
Respuesta: Air relative humidity Air temperature Calculated air temperature Average air temperature Average relative humidity Average CO2 concentration Indoor air relative humidity position 1 Indoor air relative humidity position 2 Indoor air temperature position 1 Indoor air temperature position 2 Indoor CO2 concentration position 1 Indoor CO2 concentration position 2 Air Temperature Water flow temperature Water return temperature number of hors running
SN_PC06 ¿Dónde se localiza el sensor?
Respuesta: Engineering Building Ground Floor Room G047 AHU101 After cooling coil Component GFC4 Riser Return duct Supply air outlet duct Component Duct Section After Cooling Coil Component Duct Section After Pre-Heating Coil Component Duct Section Return Duct Component Duct Section Supply Air Outlet
SN_PC07 ¿Dónde se localiza el sensor Ítem físicamente?

Respuesta: | Engineering Building | Ground Floor | Room G047 | AHU101 | After cooling coil | Component GFC4 Riser | Return duct Supply air outlet duct | Component Duct Section After Cooling Coil | Component Duct Section After Pre-Heating Coil | Component Duct Section Return Duct | Component Duct Section Supply Air Outlet |

SN_PC08 ¿A qué parte del climatizador pertenece el sensor?

Respuesta: | Engineering Building | Ground Floor | Room G047 | AHU101 | After cooling coil | Component GFC4 Riser | Return duct Supply air outlet duct | Component Duct Section After Cooling Coil | Component Duct Section After Pre-Heating Coil | Component Duct Section Return Duct | Component Duct Section Supply Air Outlet |

SN_PC09 ¿Qué nombre tiene asociado el sensor en su etiqueta?

Respuesta: | Sensor room G047: average space air temperature (D19-477) | Sensor room G047: indoor air relative humidity position 1 (D12-456) | Sensor room G047: indoor air relative humidity position 2 (D9-464) | Sensor room G047: indoor air temperature position 1 (D11-469) | Sensor room G047: indoor air temperature position 2 (D8-465) | Sensor room G047: indoor CO2 concentration position 1 (D10-462) | Sensor room G047: indoor CO2 concentration position 2 (D7-466) | Sensor room G047: average relative humidity (D28-482) | Sensor After Cooling Coil: Air Temperature (D5-478) | Sensor After Cooling Coil: Air Temperature (D6-459) | Sensor After Pre-Heating Coil: Air Temperature (D3-463) | Sensor of GFC4 Riser: water flow-temperature (D27-457) | Sensor of GFC4 Riser: water return-temperature (D30-460) | Sensor Return Duct: Air Relative Humidity (D13-458) | Sensor Return Duct: Air Temperature (D20-474) | Sensor Return Duct: Room G047 Average CO2 Concentration (D26-481) | Sensor Supply Air Outlet Duct: Air Temperature (D4-472) | Sensor Supply Air Outlet Duct: Calculated Air Temperature (D22-470) |

▪ **Preguntas de competencia ACCIONADOR (actuador)**

AC_PC01 ¿Qué tipo de dispositivo de flujo es el accionador?

Respuesta: | Flow instrument | Valve |

AC_PC02 ¿Qué tipo de accionador es el accionador?

Respuesta: | Differential pressure switch | Percentage opening | Percentage velocity |

AC_PC03 ¿Qué propiedad controla el accionador?

Respuesta: | Extract fan velocity percentage | Supply fan velocity percentage | Percentage opening of valve of cooling coil | Percentage opening of valve of GFC4 riser | Percentage opening of valve of post-heating coil | Percentage opening of valve of pre-heating coil |

AC_PC04 ¿Dónde produce su efecto el accionador?

Respuesta: | Extract air flow | Supply air flow | Valve of GFC4 Riser | Valve of cooling coil | Valve of post-heating coil | Valve of pre-heating coil |

AC_PC05 ¿Qué tipo de servicio ofrece el accionador?

Respuesta: | Filter Monitoring | *ninguno* |

AC_PC06 ¿Dónde se localiza el accionador?

Respuesta: | Engineering Building | Ground Floor | Room G047 | AHU101 | After cooling coil | Component GFC4 Riser | Return duct Supply air outlet duct | Component Duct Section After Cooling Coil | Component Duct Section After Pre-Heating Coil | Component Duct Section Return Duct | Component Duct Section Supply Air Outlet |

AC_PC07 ¿Dónde se localiza el accionador físicamente?

Respuesta: | Engineering Building | Ground Floor | Room G047 | AHU101 | After cooling coil | Component GFC4 Riser | Return duct Supply air outlet duct | Component Duct Section After Cooling Coil | Component Duct Section After Pre-Heating Coil | Component Duct Section Return Duct | Component Duct Section Supply Air Outlet |

AC_PC08 ¿A qué parte del climatizador pertenece el accionador?

Respuesta: | Engineering Building | Ground Floor | Room G047 | AHU101 | After cooling coil | Component GFC4 Riser | Return duct Supply air outlet duct | Component Duct Section After Cooling Coil | Component Duct Section After Pre-Heating Coil | Component Duct Section Return Duct | Component Duct Section Supply Air Outlet |

AC_PC09 ¿Qué nombre tiene asociado el accionador en su etiqueta?

Respuesta: | Actuator Extract Air Flow: Differential Pressure Switch (D18-461) | Actuator Supply Air Flow: Differential Pressure Switch (D17-473) | Actuator Valve Of Cooling Coil: Percentage Opening (D25-468) | Actuator Valve Of Cooling Coil: Percentage Opening (D25-468) | Actuator Valve Of Post-Heating Coil: Percentage Opening (D23-475) | Actuator Valve Of Pre-Heating Coil: Percentage Opening (D14-471) |

▪ **Preguntas de competencia INTERRUPTOR (position)**

IN_PC01 ¿Qué tipo de interruptor de estado es el accionador?

Respuesta: | Alarm | Switching device |

IN_PC02 ¿Sobre qué propiedad controla el estado el interruptor de estado?

Respuesta: | Fire alarm | Gas CO2 alarm | Extract fan enable status | Supply fan enable status|

IN_PC03 ¿Dónde produce su efecto el interruptor de estado?

Respuesta: | AHU101 | Extract bag filter | Extract fan | Supply bag filter | Supply fan | Supply panel filter |

IN_PC04 ¿Qué tipo de servicio ofrece el interruptor de estado?

Respuesta: | Service general alarms | *ninguno* |

IN_PC05 ¿Dónde se localiza el interruptor de estado?

Respuesta: | Engineering Building | Ground Floor | Room G047 | AHU101 | After cooling coil | Component GFC4 Riser | Return duct Supply air outlet duct | Component Duct Section After Cooling Coil | Component Duct Section After Pre-Heating Coil | Component Duct Section Return Duct | Component Duct Section Supply Air Outlet |

IN_PC06 ¿Dónde se localiza el interruptor de estado físicamente?

Respuesta: | Engineering Building | Ground Floor | Room G047 | AHU101 | After cooling coil | Component GFC4 Riser | Return duct Supply air outlet duct | Component Duct Section After Cooling Coil | Component Duct Section After Pre-Heating Coil | Component Duct Section Return Duct | Component Duct Section Supply Air Outlet |

IN_PC07 ¿A qué parte del climatizador pertenece el interruptor de estado?

Respuesta: | Engineering Building | Ground Floor | Room G047 | AHU101 | After cooling coil | Component GFC4 Riser | Return duct Supply air outlet duct | Component Duct Section After Cooling Coil | Component Duct Section After Pre-Heating Coil | Component Duct Section Return Duct | Component Duct Section Supply Air Outlet |

IN_PC08 ¿Qué nombre tiene asociado el interruptor de estado en su etiqueta?

Respuesta: | Position AHU101 Fire Alarm (D29-476) | Position AHU101 Gas CO2 Alarm | Position Extract Bag Filter Differential Pressure Switch | Position Extract Fan Enable Status (D16-479) | Position Supply Bag Filter Differential Pressure Switch | Position Supply Fan Enable Status (D15-455) | Position Supply Panel Filter Differential Pressure Switch |

▪ **Preguntas de competencia MEDICIÓN (meter)**

MD_PC01 ¿A qué tipo de propiedad pertenece la medición?

Respuesta: | Energy | Humidity | Occupancy | Smoke | Status | Temperature | Time |

MD_PC02 ¿Cuáles son las unidades de medida asociadas a la medición?

Respuesta: | Date-Time unit | Humidity unit | Occupancy unit | Smoke unit | Status unit | Temperature unit |

<p>MD_PC03 ¿Qué valor tiene asociado la medición?</p> <p>Respuesta: 112.16331481933594 7.793083190917969 99.0 0.0 18.0 24.767738342285156 2126.0 37.19330978393555 ...</p> <p>MD_PC04 ¿Qué marca temporal tiene asociada la medición?</p> <p>Respuesta: 2018-12-04T04:03:00+01:00 2018-12-04T04:00:00+01:00 2019-03-10T09:23:40+01:00 2018-07-04T04:03:00+01:00 2019-08-07T04:24:00+01:00 2020-03-10T09:23:40+01:00 2021-01-04T04:03:00+01:00 2021-10-04T04:24:00+01:00 2022-03-10T09:23:40+01:00 ...</p> <p>MD_PC05 ¿Qué nombre tiene asociado la medición en su etiqueta?</p> <p>Respuesta: Measure TE 101 1 Frost Coil Temp 2018-12-04T04:03:00.000+01:00 Measure TCV 101 3 Frost Coil HValve 2018-12-04T05:02:00.000+01:00 Measure MCC02 Fire Alarm 2018-12-04T04:19:00.000+01:00 Measure LPHW 2 GFC4 Riser 3 IFM 2018-12-04T04:46:00.000+01:00 Measure 117 Lecture Theatre 3 Av Humidity 2018-12-04T04:06:00.000+01:00 Measure AHU101 Calc Supply Setpt 2018-12-04T04:00:00.000+01:00 ...</p> <p>MD_PC06 ¿Cuál fue el histórico de valores para la temperatura media del aire en la sala?</p> <p>Respuesta: 18.16331481933594, 2018-12-04T04:03:00+01:00 18.793083190917969, 2018-12-04T04:04:00+01:00 18.16331481933594, 2018-12-04T04:05:00+01:00 18.16331471933594, 2018-12-04T04:06:00+01:00 18.767788342285156, 2018-12-04T04:07:00+01:00 18.16231481933594, 2018-12-04T04:08:00+01:00 ...</p> <p>MD_PC07 ¿Cuál fue histórico de valores de la humedad media del aire en la sala entre las fechas 2019-07-25T00:00:00+01:00 y 2019-07-25T23:59:59+01:00 ?</p> <p>Respuesta: 38.16331481933594, 2019-07-04T04:03:00+01:00 38.793083190917969, 2019-07-04T04:04:00+01:00 38.16331481933594, 2019-07-04T04:05:00+01:00 38.16331471933594, 2019-07-04T04:06:00+01:00 38.767788342285156, 2019-07-04T04:07:00+01:00 38.16231481933594, 2019-07-04T04:08:00+01:00 ...</p> <p>MD_PC08 ¿Cuál fue el histórico con valor máximo y medio de la concentración de CO2-Posición 1 en la sala?</p> <p>Respuesta: 538.16331481933594, 2018-12-04T04:03:00+01:00 535.793083190917969, 2018-12-04T04:04:00+01:00 538.16331481933594, 2018-12-04T04:05:00+01:00 537.16331471933594, 2018-12-04T04:06:00+01:00 536.767788342285156, 2018-12-04T04:07:00+01:00 536.16231481933594, 2018-12-04T04:08:00+01:00 ...</p>
--

Tabla 11. Requisitos funcionales antes de la validación

Durante el proceso de validación se ha determinado lo siguiente:

- la correcta comprensión sobre las CQ planteadas
- de la base de análisis se rechazan los requisitos que determinan las CQ que responden a objetos irrelevantes para la conceptualización o que están duplicados. Concretamente desaparecerán todas aquellas CQ que den respuesta a localizaciones físicas de componentes o su posición dentro de cada dispositivo, así como a las tareas que no sean la medición de valores directos al no ser requeridos para los objetivos de realizar búsquedas y valores descriptivos con los datos, desapareciendo por tanto de los ítems “SENSOR”, “ACCIONADOR” e “INTERRUPTOR”
- las CQ quedan enmarcadas dentro del dominio y no presentan ambigüedades o dudas por los usuarios finales

5.1.7. Tarea 7. Priorización de los requisitos

Se establecen diferentes niveles de prioridad para los grupos de CQs y dentro de ellos para cada una de las necesidades detectadas. Estas prioridades se emplean para planificar el desarrollo de la red de ontologías, aunque el grado de complejidad y el tiempo de desarrollo de la ontología en este proyecto no lo requiera especialmente. Las prioridades, en orden decreciente de importancia son:

- **Prioridad 1:** medida, estado, posición, sensor
- **Prioridad 2:** propiedades, función, característica
- **Prioridad 3:** localización, concepto

En el documento ORSD de la Tabla 12 se muestra la lista completa de los CQs priorizados.

5.1.8. Tarea 8. Extracción de terminología y sus frecuencias

Es extraer de las CQs y de sus respuestas el pre-glosario de términos como nombres, adjetivos y verbos. Esta terminología se utiliza en el desarrollo de la red de ontologías para lo que podría emplearse una herramienta automatizada para la extracción de *tokens* y frecuencias sobre las CQs, pero el carácter limitado del dominio facilita su realización manual. En el documento ORSD ya se muestran los términos y sus frecuencias, Tabla 12.

5.1.9. ORSD. Documento de Especificación de Requisitos

Con la finalización de las tareas completada, se elabora el Documento de Especificación de Requisitos de la Ontología que se muestra en la siguiente tabla, Tabla 12.

Ontology Requirements Specification Document	
1	Propósito
	Modelizar semánticamente los datos de sensores del climatizador AHU-101 del Alice Perry Engineering Building de la NUI en Galway, Irlanda.
2	Alcance
	Trasladar la ontología modelizada a un entorno de desarrollo informático sólido y facilitar la realización de desarrollos posteriores. Se prescinde de las partes relacionadas con la inferencia y se prioriza un modelaje correcto del dispositivo.
3	Lenguaje de implementación
	OWL-DL
4	Usuarios finales previstos
	<ul style="list-style-type: none"> ▪ Usuario 1: usuario general, un investigador o alumno que quieran hacer uso de los recursos almacenados para visualizarlos en forma de gráficas ▪ Usuario 2: usuario técnico, investigador que requiere controlar los recursos para su labor docente o investigadora
5	Usos previstos
	<ul style="list-style-type: none"> ▪ Uso 1: búsqueda. Los usuarios finales pueden buscar cualquier medición o elemento de la AHU, así como sus características ▪ Uso 2: navegación. Los usuarios finales pueden acceder al histórico de cualquier medición de la AHU, así como sus características ▪ Uso 3: Visualización. Los usuarios finales pueden buscar visualizar los históricos de registros en un entorno gráfico o realizar consultas sobre estadísticas descriptivas básicas
6	Requisitos
	a. Requisitos no funcionales

- **RNF-01.** La ontología debe reutilizar un modelo conceptual ya definido
- **RNF-02.** La ontología en lo posible, debe reutilizar recursos ontológicos existentes
- **RNF-03.** La ontología debe garantizar la representación fiable de los datos
- **RNF-04.** La ontología debe permitir realizar búsquedas y estadísticas descriptivas básicas visibles mediante gráficas
- **RNF-05.** La ontología en la medida de lo posible, debe representar todos los elementos y sensores
- **RNF-06.** La ontología, en lo posible, debe ser extensible, escalable y exportable
- **RNF-07.** La ontología debe mantener separados el modelo de los datos
- **RNF-08.** La ontología debe garantizar la identificación de cada medida con su sensor
- **RNF-09.** La ontología debe garantizar la identificación de la jerarquía de sensores
- **RNF-10.** La ontología debe garantizar la identificación estructural de cada elemento de la AHU

b. Requisitos funcionales: Grupos de preguntas de competencia

▪ **Preguntas de competencia SENSOR (sensor)**

SN_PC01 ¿Qué tipo de sensor es el sensor?

Respuesta: | Temperature sensor | Humidity sensor | Smoke sensor |

SN_PC02 ¿Qué tipo de propiedad controla el sensor?

Respuesta: | Status | Smoke | Temperature | Occupancy | Humidity | Time |

SN_PC03 ¿Qué propiedad controla el sensor?

Respuesta: | Air relative humidity | Air temperature | Calculated air temperature | Average air temperature | Average relative humidity | Average CO2 concentration | Indoor air relative humidity position 1 | Indoor air relative humidity position 2 | Indoor air temperature position 1 | Indoor air temperature position 2 | Indoor CO2 concentration position 1 | Indoor CO2 concentration position 2 | Air Temperature | Water flow temperature | Water return temperature| number of hors running |

SN_PC04 ¿Qué nombre tiene asociado el sensor en su etiqueta?

Respuesta: | Sensor room G047: average space air temperature (D19-477) | Sensor room G047: indoor air relative humidity position 1 (D12-456) | Sensor room G047: indoor air relative humidity position 2 (D9-464) | Sensor room G047: indoor air temperature position 1 (D11-469) | Sensor room G047: indoor air temperature position 2 (D8-465) | Sensor room G047: indoor CO2 concentration position 1 (D10-462) | Sensor room G047: indoor CO2 concentration position 2 (D7-466) | Sensor room G047: average relative humidity (D28-482) | Sensor After Cooling Coil: Air Temperature (D5-478) | Sensor After Cooling Coil: Air Temperature (D6-459) | Sensor After Pre-Heating Coil: Air Temperature (D3-463) | Sensor of GFC4 Riser: water flow-temperature (D27-457) | Sensor of GFC4 Riser: water return-temperature (D30-460) | Sensor Return Duct: Air Relative Humidity (D13-458) | Sensor Return Duct: Air Temperature (D20-474) | Sensor Return Duct: Room G047 Average CO2 Concentration (D26-481) | Sensor Supply Air Outlet Duct: Air Temperature (D4-472) | Sensor Supply Air Outlet Duct: Calculated Air Temperature (D22-470) |

▪ **Preguntas de competencia ACCIONADOR (actuador)**

AC_PC01 ¿Qué tipo de dispositivo de flujo es el accionador?

Respuesta: | Flow instrument | Valve |

AC_PC02 ¿Qué propiedad controla el accionador?

Respuesta: | Extract fan velocity percentage | Supply fan velocity percentage | Percentage opening of valve of cooling coil | Percentage opening of valve of GFC4 riser | Percentage opening of valve of post-heating coil | Percentage opening of valve of pre-heating coil |

<p>AC_PC03 ¿Qué nombre tiene asociado el accionador en su etiqueta?</p> <p>Respuesta: Actuator Extract Air Flow: Differential Pressure Switch (D18-461) Actuator Supply Air Flow: Differential Pressure Switch (D17-473) Actuator Valve Of Cooling Coil: Percentage Opening (D25-468) Actuator Valve Of Cooling Coil: Percentage Opening (D25-468) Actuator Valve Of Post-Heating Coil: Percentage Opening (D23-475) Actuator Valve Of Pre-Heating Coil: Percentage Opening (D14-471) </p> <p>▪ Preguntas de competencia INTERRUPTOR (position)</p> <p>IN_PC01 ¿Qué tipo de interruptor de estado es?</p> <p>Respuesta: Alarm Switching device </p> <p>IN_PC02 ¿Sobre qué propiedad controla el estado el interruptor de estado?</p> <p>Respuesta: Fire alarm Gas CO2 alarm Extract fan enable status Supply fan enable status </p> <p>IN_PC08 ¿Qué nombre tiene asociado el interruptor de estado en su etiqueta?</p> <p>Respuesta: Position AHU101 Fire Alarm (D29-476) Position AHU101 Gas CO2 Alarm Position Extract Bag Filter Differential Pressure Switch Position Extract Fan Enable Status (D16-479) Position Supply Bag Filter Differential Pressure Switch Position Supply Fan Enable Status (D15-455) Position Supply Panel Filter Differential Pressure Switch </p> <p>▪ Preguntas de competencia MEDICIÓN (meter)</p> <p>MD_PC01 ¿Cuáles son las unidades de medida asociadas a la medición?</p> <p>Respuesta: Date-Time unit Humidity unit Occupancy unit Smoke unit Status unit Temperature unit </p> <p>MD_PC02 ¿Qué valor tiene asociado la medición?</p> <p>Respuesta: 112.16331481933594 7.793083190917969 99.0 0.0 24.767738342285156 2126.0 37.19330978393555 ...</p> <p>MD_PC03 ¿Qué marca temporal tiene asociada la medición?</p> <p>Respuesta: 2018-12-04T04:03:00+01:00 2019-03-10T09:23:40+01:00 2018-12-04T04:03:00+01:00 2020-03-10T09:23:40+01:00 2021-12-04T04:03:00+01:00 2022-03-10T09:23:40+01:00 ...</p> <p>MD_PC04 ¿A qué tipo de propiedad pertenece la medición?</p> <p>Respuesta: Energy Humidity Occupancy Smoke Status Temperature Time </p> <p>MD_PC05 ¿Qué nombre tiene asociado la medición en su etiqueta?</p> <p>Respuesta: Measure TE 101 1 Frost Coil Temp 2018-12-04T04:03:00.000+01:00 Measure TCV 101 3 Frost Coil HValve 2018-12-04T05:02:00.000+01:00 Measure MCC02 Fire Alarm 2018-12-04T04:19:00.000+01:00 Measure LPHW 2 GFC4 Riser 3 IFM 2018-12-04T04:46:00.000+01:00 ...</p> <p>MD_PC06 ¿Cuál fue el histórico de valores para la temperatura media del aire en la sala?</p> <p>Respuesta: 18.16331481933594, 2018-12-04T04:03:00+01:00 18.793083190917969, 2018-12-04T04:04:00+01:00 18.16331481933594, 2018-12-04T04:05:00+01:00 18.16331471933594, 2018-12-04T04:06:00+01:00 18.767788342285156, 2018-12-04T04:07:00+01:00 18.16231481933594, 2018-12-04T04:08:00+01:00 ...</p> <p>MD_PC07 ¿Cuál fue histórico de valores de la humedad media del aire en la sala entre las fechas 2018-12-05T04:00:00+01:00 y 2018-12-05T05:03:00+01:00?</p> <p>Respuesta: 37.16331481933594, 2018-12-04T04:03:00+01:00 37.793083190917969, 2018-12-04T04:04:00+01:00 37.16331481933594, 2018-12-04T04:05:00+01:00 37.16331471933594, 2018-12-04T04:06:00+01:00 37.767788342285156, 2018-12-04T04:07:00+01:00 37.16231481933594, 2018-12-04T04:08:00+01:00 ...</p>

	MD_PC08 ¿Cuál fue el histórico con valor máximo y medio de la concentración de CO2- Posición 1 en la sala? Respuesta: 538.16331481933594, 2018-12-04T04:03:00+01:00 535.793083190917969, 2018-12-04T04:04:00+01:00 538.16331481933594, 2018-12-04T04:05:00+01:00 537.16331471933594, 2018-12-04T04:06:00+01:00 536.767788342285156, 2018-12-04T04:07:00+01:00 536.16231481933594, 2018-12-04T04:08:00+01:00 ...				
7	Pre-Glosario de Términos				
	a. Términos de las CQ y características generales + frecuencia (F * número de medidas)				
	SENSOR (sensor)				
sensor	5	controla	2	etiqueta	1
tipo	2	nombre	1		
propiedad	2	tiene asociado	1		
	ACCIONADOR (actuador)				
tipo	1	propiedad	1	tiene asociado	1
dispositivo de flujo	1	controla	1	etiqueta	1
accionador	3	nombre	1		
	INTERRUPTOR (position)				
tipo	1	propiedad	1	nombre	1
interruptor de estado	3	controla	1	tiene asociado	1
accionador	1	estado	1	etiqueta	1
	MEDICIÓN (meter)				
unidades de medida	1	marca temporal	1	nombre	1
asociadas	2	tipo	1	tiene asociado	1
medición	5	propiedad	1	etiqueta	1
valor	1	pertenece	1		
	b. Términos de las respuestas + frecuencia (F * número de medidas)				
	SENSOR (sensor)				
Temperature sensor	11	Average relative humidity			1
Humidity sensor	4	Average CO2 concentration			1
Smoke sensor	4	Indoor air relative humidity position 1			1
Status	3	Indoor air relative humidity position 2			1
Smoke	3	Indoor air temperature position 1			1
Temperature	1	Indoor air temperature position 2			1
Occupancy	6	Indoor CO2 concentration position 1			1
Humidity	4	Indoor CO2 concentration position 2			1
Time	2	Air Temperature			4
Air relative humidity	3	Water flow temperature			2
Air temperature	3	Water return temperature			2
Calculated air temperature	1	Number of hors running			1
Average air temperature	1				
	ACCIONADOR (actuador)				
Extract fan velocity percentage	1	Perc. opening of valve of GFC4 riser			1
Supply fan velocity percentage	1	Perc. opening of valve of post-heating coil			1

Percentage opening of valve of cooling coil			1
Perc. opening of valve of pre-heating coil			1
Actuator Extract Air Flow: Differential Pressure Switch (D18-461)			1
Actuator Supply Air Flow: Differential Pressure Switch (D17-473)			1
Actuator Valve Of Cooling Coil: Percentage Opening (D25-468)			1
Actuator Valve Of Post-Heating Coil: Percentage Opening (D23-475)			1
Actuator Valve Of Pre-Heating Coil: Percentage Opening (D14-471)			1
INTERRUPTOR (position)			
Fire alarm	1	Supply fan enable status	1
Gas CO2 alarm	1	Position AHU101 Fire Alarm (D29-476)	1
Extract fan enable status	1	Position AHU101 Gas CO2 Alarm	1
Position Extract Bag Filter Differential Pressure Switch			1
Position Extract Fan Enable Status (D16-479)			1
Position Supply Bag Filter Differential Pressure Switch			1
Position Supply Fan Enable Status (D15-455)			1
Position Supply Panel Filter Differential Pressure Switch			1
MEDICIÓN (meter)			
Date-Time unit	2	Humidity unit	4
Humidity unit	4	Occupancy unit	6
Occupancy unit	6	Smoke unit	4
Smoke unit	4	Status unit	3
Status unit	3	Temperature unit	11
Temperature unit	11	# (número entero)	3
Date-Time unit	2	## (número real)	26
yyyy-mm-ddThh:mm:ss+01:00 (timestamp)			29
Measure TE 101 ~ (etiqueta)			29
c. Objetos			
MEDICIÓN (meter)			
Measure TE 101 1 Frost Coil Temp 2018-12-04T04:03:00.000+01:00 Measure TCV 101 3 Frost Coil HValve 2018-12-04T05:02:00.000+01:00 Measure MCC02 Fire Alarm 2018-12-04T04:19:00.000+01:00 2019-03-10T09:23:40+01:00 2018-12-04T04:03:00+01:00 2020-03-10T09:23:40+01:00 2021-12-04T04:03:00+01:00 2022-03-10T09:23:40+01:00 12.16331481933594 7.793083190917969 99.0 0.0 2126.0 37.19330978393555			

Tabla 12. ORSD de la ontología

5.2. Planificación

La metodología NeOn nos propone dividir esta etapa en cuatro tareas, como muestra la Ilustración 50 del tema anterior, con el objetivo de identificar los procesos y actividades a realizar, así el tiempo y recursos necesarios para su realización.

5.2.1. Tarea 1. Selección del modelo del ciclo de vida

Para completar esta tarea se hace uso de la Tabla 7 en el Apartado 4.4 que propone la metodología para estos casos. Con los datos del desarrollo del tema anterior, las preguntas candidatas a cumplirse son la 1, la 2 y la 4. Siendo esta última rechazada porque no se van combinar dominios parecidos ni tener que

redefinir similitudes entre sus vocabularios (alineación). Vamos a emplear las ontologías SAREF y SAREF4BLDN con su vocabulario sin modificar, en versiones reducidas o ampliadas de dominio. Por tanto, tendremos un modelo de 6 fases que correspondería a la Tabla 6 del Apartado 4.4.1, pero sin la fase de Fusión.

Con respecto al ciclo de vida a elegir, todas las características que se recomiendan en el Apartado 4.4.1 sobre ciclo en cascada se cumplen: es un proyecto de corta duración, tiene requisitos cerrados, implementa una ontología ya definida, la norma ISO: SAREF, y por último el dominio es el de una AHU concreta de un edificio bien estudiado.

5.2.2. Tarea 2. Selección del conjunto de escenarios

Esta tarea tiene el objetivo de seleccionar los escenarios que se van a implementar para el desarrollo de la ontología, para integrarlos en un gráfico de ciclo de vida inicial como diagrama de Gantt. Siguiendo el mismo razonamiento de la tarea anterior para responder a las preguntas de la Tabla 7, y contando con que el Escenario 1 de uso obligatorio, se precisa del Escenario 2 para los NOR y el Escenario 8 para los recursos ontológicos. El diagrama con la planificación desarrollada se puede consultar en la Ilustración 63, al ser un desarrollo unipersonal no se van a incluir tareas solapadas.

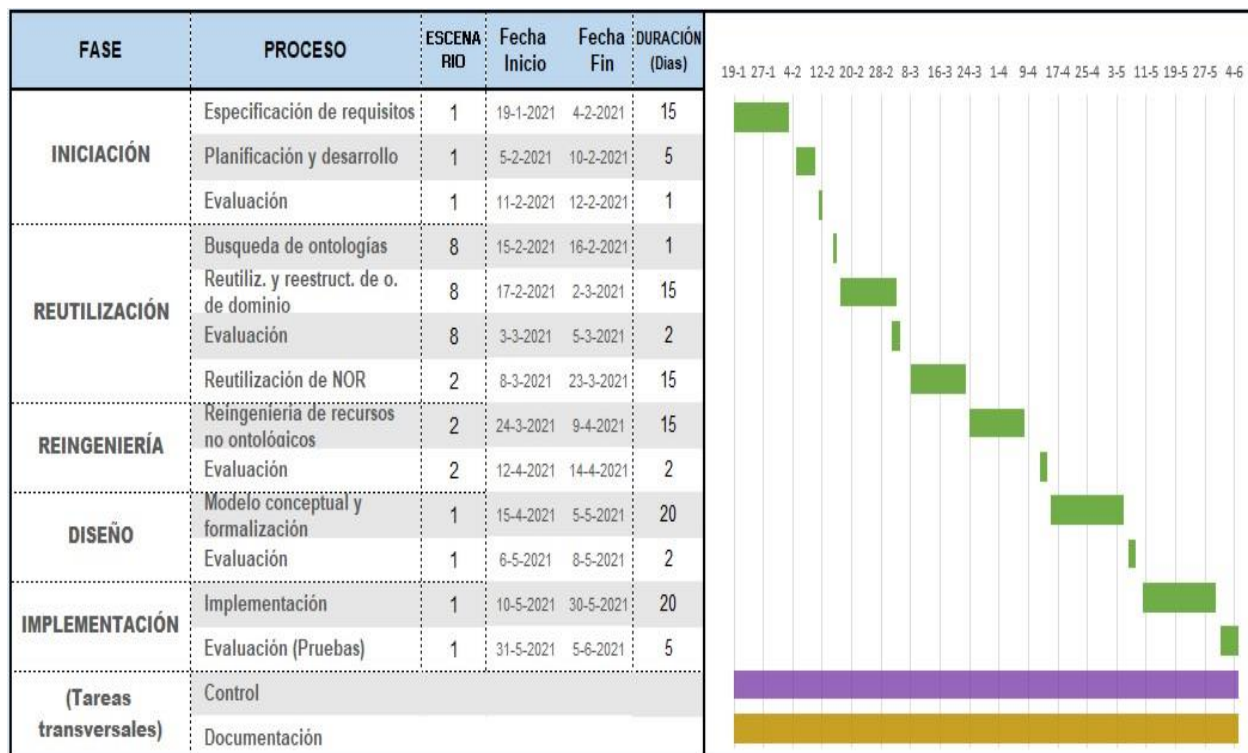


Ilustración 63. NeOn: diagrama de Gantt con la planificación estimada

5.2.3. Tarea 3. Actualización del plan inicial

Esta tarea, y para nuestro modelo en cascada que carece de ciclos y donde no se prevé ninguna modificación de requerimientos o alcance, no sería necesaria. El único cambio que se puede efectuar sobre el plan inicial sería un ajuste en las fechas establecidas para las fases por el cumplimiento de algún riesgo que no se haya podido gestionar, ver Capítulo 6. Los posibles cambios de fechas quedarían reflejados en el Capítulo 10 sobre seguimiento del proyecto. Se actualiza la información contenida en el diagrama de Gantt con un resumen, asignando el total de horas estimadas para cada tarea (incluyendo

las horas de las tareas transversales por fase). El resultado, desvinculado de fechas, se puede consultar en la Tabla 13.

PLANIFICACIÓN ESTIMADA	
FASE	TIEMPO ESTIMADO
INICIACIÓN (plan de proyecto)	20
REUTILIZACIÓN	15
REINGENIERÍA	20
DISEÑO	15
IMPLEMENTACIÓN	40
Total (horas)	110

Tabla 13. Resumen de planificación estimada por horas

5.2.4. Tarea 4. Establecimiento de restricciones y asignación de recursos

En esta tarea se incluye cualquier información adicional a la planificación y asignación de recursos humanos a cualquier proceso o actividad. Debido al corto alcance del TFG y la ontología no se agrega ningún recurso ni se modifica nada. Se incluye como información adicional previa a la planificación una fase denominada "Fase preliminar de documentación y estudio" que incluye las tareas de documentación y estudio necesarias para la realización de la memoria, y estimadas en 200 horas. Para más información al respecto consultar el Apartado 10.2.

Capítulo 6

6. Plan de riesgos y presupuestos

6.1. Gestión de riesgos

Exponerse a un riesgo implica la existencia de una posibilidad de pérdida o daño que impide la consecución de unos objetivos. Un proyecto de software es una actividad dinámica que se ve afectada por multitud de factores, internos y externos, por lo que se encuentra expuesta a una gran cantidad de riesgos que pueden originarse por distintas causas. Los riesgos pueden influir en la consecución de sus objetivos y por tanto en el desarrollo y finalización del mismo, es de una gran importancia, por tanto, la identificación, el análisis y el tratamiento de los diferentes riesgos que pueden afectar a un proyecto.

La gestión de riesgos consiste en identificar y priorizar los que son más probables y que podrían tener un mayor impacto, para abordarlos primero mediante un plan de reducción de riesgos. Los riesgos identificados deben documentarse en un registro de riesgos junto con las soluciones propuestas para mitigarlos. Pasos de la gestión de riesgos:

- **Análisis de los riesgos:** para determinar la probabilidad de que se originen, analizar los factores de riesgo y documentar sus posibles consecuencias.
- **Evaluación y valoración de los riesgos:** realizar auditorías internas y análisis de riesgos para determinar su magnitud. Se debe decidir qué nivel de riesgo es aceptable y cuáles deben abordarse de inmediato.
- **Reducción de los riesgos:** tras determinar la prioridad y la importancia de los riesgos, se puede proceder con una estrategia de respuesta para disminuirlos o controlarlos.
- **Supervisión de los riesgos:** deben supervisarse permanentemente para garantizar que los planes para reducirlos estén funcionando, o para saber si un riesgo se convierte en una mayor amenaza.

Las principales estrategias de gestión de riesgos son:

- **Evasión de riesgos:** consiste en detener y evitar cualquier actividad que pueda implicar un riesgo.
- **Reducción de riesgos:** se centra en las medidas que reducirán la probabilidad de un riesgo o su impacto.
- **Intercambio de riesgos:** se genera cuando se transfiere parte del riesgo a otra empresa, o lo comparte con ella.
- **Conservación de riesgos:** tiene lugar cuando se decide aceptar el riesgo potencial después de haber realizado una evaluación. No se toma ninguna medida para reducir el riesgo, pero podría implementarse un plan de contingencia.

6.1.1. Plan de riesgos

Algunos de los aspectos mencionados en la introducción no pueden ser tenidos en cuenta debido a las condiciones en las que se desarrolla un TFG, como puede ser claramente la estrategia de intercambio de riesgos. En cualquier caso, la corta duración del proyecto no ofrece un marco temporal suficiente para la aparición de muchos riesgos. En la Tabla 14 se muestran los principales riesgos susceptibles de afectar a este proyecto.

IDENTIFICACIÓN DE RIESGOS	
ID	TÍTULO
RISK-01	Retrasos en la planificación
RISK-02	Desconocimiento de las tecnologías a utilizar
RISK-03	Cambios en la especificación de requisitos
RISK-04	Problemas de salud
RISK-05	Ausencia del tutor
RISK-06	Falta de tiempo
RISK-07	Problemas de hardware con el equipo
RISK-08	Falta de experiencia
RISK-09	Llegar a un punto de bloqueo en la investigación
RISK-10	Mala planificación

Tabla 14. Listado de riesgos

Una vez identificados los riesgos a tratar se debe pasar a un análisis cuantitativo de los mismos. Mediante la Matriz de Probabilidad de Impacto se facilita la priorización de riesgos en función de su probabilidad de ocurrencia y el impacto asociado a cada uno, Tabla 15.

RIESGOS PRIORIZADOS				
ID	TÍTULO	IMPACTO	PROBABILIDAD	PRIORIDAD
RISK-01	Retrasos en la planificación	Alto	Media	3
RISK-02	Desconocimiento de las tecnologías a utilizar	Alto	Alta	1
RISK-03	Cambios en la especificación de requisitos	Medio	Baja	9
RISK-04	Problemas de salud	Medio	Media	4
RISK-05	Ausencia del tutor	Medio	Baja	5
RISK-06	Falta de tiempo	Medio	Media	6
RISK-07	Problemas de hardware con el equipo	Medio	Baja	7
RISK-08	Falta de experiencia	Alto	Media	10
RISK-09	Llegar a un punto de bloqueo en la investigación	Alto	Media	2
RISK-10	Mala planificación	Media	Media	8

Tabla 15. Listado de riesgos priorizado

El siguiente paso consiste en llevar a cabo actividades y medidas que permitan gestionar los riesgos, estableciéndose dos tipos de planes:

- **Plan de mitigación:** definido en PMBOOK [90] como el conjunto de acciones a través de las cuales se pretende reducir la probabilidad de ocurrencia del riesgo.
- **Plan de contingencia:** conjunto de acciones realizadas como respuesta a la materialización de un riesgo, se trata de reducir el impacto del riesgo una vez que ya ha ocurrido.

El resultado de este análisis se muestra desde la Tabla 16 hasta la Tabla 25.

ID	RISK-01
Título	Retrasos en la planificación
Tipo	Planificación
Descripción	Es posible que existan retrasos de planificación, a lo largo del desarrollo del proyecto. Estos retrasos pueden derivarse de una mala planificación inicial.
Estrategia	Evitación del riesgo
Plan de mitigación	Se han añadido días de trabajo a mayores de los previstos
Plan de contingencia	Incrementar el número de horas diarias de trabajo

Tabla 16. Risk-01. Retrasos en la planificación

ID	RISK-02
Título	Desconocimiento de las tecnologías a utilizar
Tipo	Técnico
Descripción	El ámbito que rodea al proceso de estudio y documentación y el entorno tecnológico están en continuo desarrollo, incluso son novedosos en algunos aspectos, no formando parte de portafolio de la carrera
Estrategia	Prevención del riesgo
Plan de mitigación	Parte de la dilatada fase inicial se ha empleado para investigar y trabajar con estos conceptos y tecnologías
Plan de contingencia	Incrementar el número de horas diarias de trabajo, comunicaciones con los tutores y tener acceso a documentación y contacto con expertos gracias al entorno universitario

Tabla 17. Risk-02. Desconocimiento de las tecnologías a utilizar

ID	RISK-03
Título	Cambios en la especificación de requisitos
Tipo	Contractual
Descripción	La ampliación de nuevas áreas de desarrollo o modificación de las existentes durante el proceso de documentación y estudio que hagan interesante ampliar o modificar requisitos previos
Estrategia	Prevención de riesgo
Plan de mitigación	Ampliar la fase en la que surja el cambio
Plan de contingencia	Realizar los cambios correspondientes en los requisitos

Tabla 18. Risk-03. Cambios en la especificación de requisitos

ID	RISK-04
Título	Problemas de salud
Tipo	Personal
Descripción	En el contexto sanitario en el que se desarrolla este proyecto, la probabilidad de contraer enfermedad que incapacite para realizar las tareas es considerable
Estrategia	Mitigación de riesgo
Plan de mitigación	La fase inicial tan dilatada resuelve muchos inconvenientes, posibilidad de alargar el tiempo al final para corregir tareas inconclusas
Plan de contingencia	Retrasar ciertas tareas antes de avanzar en cada fase, posponer fases

Tabla 19. Risk-04. Problemas de salud

ID	RISK-05
Título	Ausencia del tutor
Tipo	Contractual
Descripción	Por disponibilidad horaria o riesgo de enfermedad del tutor, se ausenta o no puede acudir a las reuniones importantes en el desarrollo del proyecto
Estrategia	Mitigación del riesgo
Plan de mitigación	Se acuerdan reuniones con antelación suficiente como para prever imprevistos y reservar horarios. Igualmente, disponer de dos tutores minimiza el riesgo de que los dos estén indisponibles a la vez
Plan de contingencia	El desarrollador (alumno) asume una mayor responsabilidad

Tabla 20. Risk-05. Ausencia del tutor

ID	RISK-06
Título	Falta de tiempo
Tipo	Personal
Descripción	Las tareas de una fase llevan más tiempo del esperado o no se dispone de horas por motivos personales/laborales para emplear en dicha fase
Estrategia	Mitigación de riesgo
Plan de mitigación	La fase inicial tan dilatada resuelve muchos inconvenientes, posibilidad de alargar la fase final para corregir tareas inconclusas
Plan de contingencia	Alargar la duración prevista para la fase

Tabla 21. Risk-06. Falta de tiempo

ID	RISK-07
Título	Problemas de hardware con el equipo
Tipo	Técnico
Descripción	La indisponibilidad del ordenador personal o los servidores de la Universidad de Valladolid que pudieran causar pérdida irreparable de información y datos del proyecto o en las aplicaciones que la procesan por caída o pérdida de los mismos
Estrategia	Prevención de riesgo
Plan de mitigación	Se realizarán copias de respaldo regularmente en unidades de almacenamiento independientes. Por otro lado, se puede contar con los <i>backup</i> de la propia universidad. Se aplicará una tecnología que sea capaz de absorber estos posibles percances. En general se facilitará la recuperación de información en caso de problemas de hardware con los equipos.
Plan de contingencia	Realizar cambios pendientes en requisitos de proyecto que posiblemente modifiquen la obtención de datos a una fuente estática debido a inconvenientes con su disponibilidad

Tabla 22. Risk-07. Problemas de hardware

ID	RISK-08
Título	Falta de experiencia
Tipo	Técnico
Descripción	La posible falta de experiencia del equipo de desarrollo del proyecto puede suponer retrasos y sobrecostos en el proyecto, así como una implementación poco eficaz del mismo
Estrategia	Prevención del riesgo
Plan de mitigación	Para prevenir este tipo de problemas se incluye tiempo dedicado a la formación personal dentro de la planificación del proyecto y en la fase inicial
Plan de contingencia	Incrementar los recursos temporales y económicos orientados a la formación personal

Tabla 23. Risk-08. Falta de experiencia

ID	RISK-09
Título	Llegar a un punto de bloqueo en la documentación y revisión
Tipo	Contractual
Descripción	Al realizar un proyecto de alto contenido en documentación y revisión cuyo objetivo final puede ser cambiante, se puede terminar dando lugar a un punto muerto, debido a la falta de herramientas o recursos con los que desarrollarlos
Estrategia	Aceptación del riesgo
Plan de mitigación	Es imposible prevenir que la revisión vaya a alcanzar un punto muerto
Plan de contingencia	Tratar de buscar una reorientación al proyecto, intentando así reciclarlo y que sea de utilidad en otro ámbito

Tabla 24. Risk-09. Llegar a un punto de bloqueo en la documentación y revisión

ID	RISK-10
Título	Mala planificación
Tipo	Planificación
Descripción	Puede haber retrasos si se realiza una mala planificación o se estima de manera incorrecta, bien sea por exceso o por defecto
Estrategia	Mitigación de riesgo
Plan de mitigación	La dilatada fase inicial resuelve muchos inconvenientes, posibilidad de alargar la duración prevista de la fase para corregir tareas inconclusas
Plan de contingencia	Ampliar el plazo de entrega o aumentar el trabajo diario.

Tabla 25. Risk-10. Mala Planificación

6.2. Presupuesto

6.2.1. Presupuesto simulado

Tras una búsqueda infructuosa en el Boletín Oficial del estado, para el salario base actualizado de un Analista Programador, se ha optado por realizar un análisis de los salarios en diversos portales de contratación. Se establece como salario bruto mínimo para un Analista Programador en 27.000,00€

anuales. Suponemos una jornada anual media de 1.750 horas efectivas, el sueldo bruto por hora es de 15,428€. Añadimos el incremento en aproximadamente el 30% del salario base que la empresa abona a la Seguridad Social y obtenemos un montante de 6.217,48€ para las 310 horas de trabajo estimadas. No se tendrán en cuenta costes fijos de carácter general como luz, agua, internet, alquiler, o material de oficina al tratarse de un trabajo desarrollado en edificios de la universidad o el alojamiento del alumno. Con respecto al presupuesto para software, no se ha encontrado la necesidad de utilizar ninguna versión de pago en los productos utilizados.

Los ordenadores como equipo informático se pueden incluir como coste, entendiendo este coste como el prorrateo para su amortización. Para realizar este TFG se ha utilizado un ordenador de sobremesa HP Pavilion TP01-1017ns - i5-10400 - 16 GB RAM, valorado en 730,00€. La Agencia Tributaria establece un coeficiente lineal máximo de amortización del 25% para “Equipos para procesos de información” y un plazo de amortización máximo de 8 años. La duración del proyecto es de 6 meses por lo que aplicando los cálculos tenemos un coste asociado de $730,00€ * 0,25 \text{ anual} * 6 \text{ meses} / 12 \text{ (meses/año)} = 91,25€$.

Para hacer frente a posibles imprevistos, sobrecostes o retrasos se aplica un fondo de provisión del 20% sobre el presupuesto calculado. La Tabla 26 muestra el resumen con el resultado de todas las operaciones presupuestadas.

PRESUPUESTO SIMULADO	
TIPO DE COSTE	CANTIDAD EN EUROS (€)
Herramientas	0,00
Costes fijos	0,00
Licencias	0,00
Material	91,25
Salarios	6.217,48
Subtotal	6.308,73
Imprevistos (20%)	1.261,74
Total	7.570,47 €

Tabla 26. Presupuesto simulado

6.2.2. Presupuesto real

Teniendo en cuenta que no existe ninguna contratación real, ya que el contexto de este trabajo es la realización del TFG, y el material pertenece al alumno estando ya amortizado, en la Tabla 27 puede verse el resumen de los costes reales del mismo que resultan ser 0,00 €.

PRESUPUESTO REAL	
TIPO DE COSTE	CANTIDAD EN EUROS (€)
Herramientas	0,00
Costes fijos	0,00
Licencias	0,00
Material	0,00
Salarios	0,00
Subtotal	0,00
Imprevistos (20%)	0,00
Total	0,00 €

Tabla 27. Presupuesto real

Capítulo 7

7. Tecnologías y software empleados

En este capítulo se muestran las diferentes tecnologías y herramientas empleadas para su implementación.

7.1. Protégé



La construcción del modelo ontológico es, sin duda, la parte práctica fundamental del proceso y se ha optado por un *framework* ampliamente difundido, con una gran interfaz gráfica que facilita el entendimiento de los procesos, y que permite desarrollos y pruebas dentro del mismo con sistemas de bases de conocimiento y razonamiento. Por tanto, y como ya se avanzó en el Apartado 2.4.6 sobre herramientas de desarrollo, se emplea [Protégé](#), que es un software para OWL2 cuyo desarrollo pertenece a la Stanford University de EE.UU., en colaboración con The University of Manchester en R.U. y soportado por el NIGMS²⁹ como recurso para ontologías y bases de conocimiento biomédico. [Protégé](#) es un editor libre de código abierto y se establece como *framework* sobre el que otros desarrolladores sugieren *plugins* que aumentan su funcionalidad. La aplicación está escrita en Java con descripción de interfaces en Swing.

7.2. LODE



[Live OWL Documentation Environment](#) (LODE) es un servicio que extrae automáticamente clases, propiedades de objetos, propiedades de datos, individuos nombrados, propiedades de anotaciones, axiomas generales y declaraciones de espacios de nombres de una ontología OWL y OWL2, y los presenta como listas ordenadas, junto con sus definiciones textuales, en un lenguaje legible por humanos en formato de página HTML diseñada para la navegación mediante enlaces incrustados. Desarrollado por Peroni, S., Shotton, D., Vitali, F. (2012) para aparecer en las Actas de la 18ª Conferencia Internacional sobre Ingeniería del Conocimiento y Gestión del Conocimiento (EKAW 2012). Código abierto.

7.3. WebVOWL



[WebVOWL](#) (*Web-based Visualization of Ontologies!*) es una aplicación web, para la visualización interactiva de ontologías OWL proporcionando representaciones gráficas para elementos del lenguaje. Existe como plugin de Protégé con menos funcionalidades. WebVOWL se publica bajo la licencia MIT. Copyright (c) 2014-2019 Vincent Link, Steffen Lohmann, Eduard Marbach, Stefan Negru, Vitalis Wiens.

²⁹ NIGMS: National Institute of General Medical Sciences EE.UU., <https://www.nigms.nih.gov/>

7.4. Oops!



[OOPS!](#) (*Ontology Pitfall Scanner!*) es una herramienta web que ayuda a detectar algunas de las trampas de diseño más comunes que aparecen al desarrollar ontologías, ya mencionada en el Apartado 2.5.4.1. También se puede utilizar como [Servicio Web](#), por lo que puede ser fácilmente integrado en software de terceros. Amplía la lista de recomendaciones que detectan los sistemas de evaluación de ontologías ya existentes y más recientes, proporcionando ejemplos y descripciones de malas prácticas. Es un desarrollo del Ontology Engineer Ingroup de la Universidad Politécnica de Madrid llevado a cabo por María Poveda-Villalón, Asunción Gómez-Pérez y Mari Carmen Suárez-Figueroa.

7.5. Python



El lenguaje de programación que se va a emplear es [Python 3](#). Python es un lenguaje de programación de alto nivel cuya popularidad ha alcanzado grandes cuotas, sobre todo asociado a desarrollos científicos y entornos de investigación, entre otros sobre Inteligencia Artificial y Minería de Datos para los que incluye numerosas bibliotecas, aunque también tiene la capacidad de integrar desarrollos existentes en otros lenguajes de programación como puede ser C o Java. Posee licencia de código abierto y su gestión la realiza The Python Software Foundation. Es importante destacar que se emplea la versión 3 del lenguaje ya que incluye una serie de cambios que la hacen incompatible con versiones anteriores.

7.6. Anaconda



En cuanto al entorno de programación y pruebas se ha optado por [Anaconda](#), un *framework* integrador sobre Python y R. Su desarrollo pertenece a Anaconda Inc. y está escrito en Python con algún módulo en C. Emplea entornos gráficos que facilitan su tratamiento. Las diferentes versiones de paquetes se administran mediante Conda como gestor de paquetes y versiones, Con su utilización básicamente tenemos un instalador y gestor de paquetes de libre distribución para recursos en Python, guiado por interfaz gráfica desde el que podemos arrancar los recursos para trabajar sobre Python, sin necesidad de configurar entornos o utilizando comandos mediante una Shell. Con su interfaz gráfica, de fácil manejo, podemos configurar diferentes perfiles de trabajo sobre Python, modificando sus versiones y paquetes instalados. También permite la instalación de nuevos recursos de programación como plugin con los que trabajar en los perfiles generados.

7.7. Jupyter Notebook



Dentro de los recursos de programación que se autoinstalan en Anaconda, tenemos el editor de código [Jupyter Notebook](#). Es una aplicación sobre web que permite crear documentos multicontenido (código, ecuaciones, texto formateado, material multimedia) y ofrecer un uso interactivo entre ellos. Tiene capacidad de dar soporte a más de 50 lenguajes de programación y a multitud de recursos por lo que es una herramienta popular a la hora de mostrar desarrollos de una manera interactiva y gráfica, sin perder las ventajas de un entorno tradicional. Es un software de libre distribución implementado por NumFOCUS. Un documento “jupyter notebook” es un documento JSON para la gestión de los diferentes recursos empleados. Para poder visualizar e interactuar con estos documentos el entorno proporciona un navegador basado en REPL. Una vez iniciado el navegador, se puede conectar a numerosos *kernel* de programación y recursos, aunque por defecto se establece Python.

7.8. Apache Spark



[Apache Spark](#) es un *framework* de código abierto que se utiliza como motor de análisis unificado para el procesamiento de datos a gran escala. Proporciona un entorno para programar *clústeres* con paralelismo de datos implícito y tolerancia a fallos. Actualmente se encuentra bajo el desarrollo de la Apache Software Foundation y proporciona APIs en Java, Scala, R y Python. [Apache Spark](#) tiene la base de su arquitectura en el llamado RDD (*Resilient Distributed DataSet*) consistente en una estructura de datos de tipo *dataframe* de solo lectura que permite guardar ítems de distintos tipos distribuidos a la largo de un *clúster* de máquinas. También soporta múltiples herramientas de alto nivel como SparkSQL, que permite procesar los datos en estructuras basadas en SQL.

7.9. JSON API



[JSON](#) (*JavaScript Object Notation*) es un formato para el intercambio de datos que describe a los mismos con una sintaxis que se usa para identificarlos y gestionarlos a la vez. Se considera una alternativa al XML ya visto en el Apartado 2.5.2 y sigue el estándar [RFC 7159](#). Dentro de la biblioteca básica del estándar de Python tenemos la API [JSON](#) que incluye todas las funciones necesarias para el tratamiento de los datos en este formato.

7.10. Owready2



[Owready2](#) es un paquete para programación orientada a ontologías en Python. Puede cargar ontologías OWL 2.0 como objetos de Python, modificarlos, guardarlos y realizar razonamientos a través del razonador Hermit³⁰ (incluido). Permite un acceso transparente a las ontologías OWL frente a su alternativa en la API basada en Java. Incluye una base de almacenamiento (triplestore/quadstore) optimizada basada en SQLite³¹ y es capaz de manejar grandes ontologías sin pérdida de eficiencia. [Owready2](#) es gestionado por el laboratorio de investigación LIMICS, University Paris 13, Sorbonne Paris Cité, INSERM UMRS 1142, Paris 6 University, por Jean-Baptiste Lamy. Está disponible bajo la licencia GNU LGPL v3 [91].

7.11. Ipywidgets



Los [ipywidgets](#), también conocidos como jupyter-widgets o simplemente widgets, son widgets HTML interactivos para desarrollos en Jupyter bajo el *kernel* de Python. Este paquete se engloba en las bibliotecas habituales de Python, y básicamente contiene objetos Python con gran cantidad de eventos asociados representables en el navegador de la plataforma. Desarrollado por Project Jupyter Contributors, bajo licencia libre.

7.12. Numpy



[NumPy](#) es una biblioteca para el lenguaje de programación Python que da soporte para crear vectores y matrices multidimensionales de gran tamaño, junto con una gran colección de funciones matemáticas de alto nivel para operar con ellas. Es un proyecto comunitario bajo supervisión del The NumPy Steering Council, con licencia de uso libre.

³⁰ Hermit OWL Reasoner, University of Oxford, es un razonador para ontologías, ver sección 2.4.5. <http://www.hermit-reasoner.com/>

³¹ SQLite3, biblioteca en C que provee de una base de datos relacional basada en almacenamiento en disco. <https://www.sqlite.org/index.html>

7.13. Matplotlib



[Matplotlib](#) es una biblioteca para la generación de gráficos a partir de datos contenidos en listas o arrays en el lenguaje de programación Python y su extensión matemática [NumPy](#). Proporciona una API, pylab, diseñada para recordar a la de MATLAB. Autor: John D. Hunter, bajo licencia libre.

Capítulo 8

8. Desarrollo de la ontología

Siguiendo las pautas establecidas por la metodología NeOn, se está implementando un ciclo de vida en cascada de 6 fases, aunque la fase de mantenimiento queda fuera del ámbito de este TFG y por tanto no se realizará. Los escenarios a cubrir y sus fases son:

- **Escenario 1:** desde la especificación a la implementación.
 - Especificación de requisitos
 - Planificación del desarrollo
 - Implementación de la ontología
 - Evaluación de la ontología

- **Escenario 2:** reutilización y reingeniería de Recursos No Ontológicos (NOR).
 - Búsqueda de NOR para el desarrollo de la ontología
 - Conceptualización de los NOR e integración en la ontología mediante ingeniería

- **Escenario 8:** reutilización y reestructuración de recursos ontológicos.
 - Búsqueda de recursos ontológicos
 - Evaluación de las ontologías encontradas
 - Selección de las más idóneas
 - Reestructuración e integración en la ontología

En este punto de la memoria ya se ha cubierto la primera fase de iniciación asociada a las dos primeras tareas del Escenario 1 de la metodología en el Capítulo 5 del plan de proyecto. Estas fases son la especificación de requisitos y planificación. En este capítulo se procederá a desarrollar las fases siguientes descritas en el ciclo de vida asignado (ver Apartado 5.2.1).

8.1. Reutilización de ontologías de dominio

Este proceso pertenece íntegramente al Escenario 8. Su objetivo es encontrar y seleccionar ontologías cuyos dominios sean o estén relacionados con la ontología que se está desarrollando para poder ser utilizadas. La metodología establece una pauta de 4 tareas que se puede revisar en el Apartado 4.5.5, Ilustración 53.

8.1.1. Tarea 1. Búsqueda de ontologías de dominio

Esta tarea se centra en la búsqueda en bibliotecas, repositorios y registros, de ontologías de dominio que sean candidatas porque puedan satisfacer las necesidades de la ontología que se está desarrollando. El punto de entrada es el ORSD (Tabla 12), y el resultado de la actividad es un conjunto de ontologías de dominio candidatas.

En este proyecto la búsqueda está cerrada desde el principio ya que se reutilizará la ontología SAREF y su extensión SAREF4BLDG como requisito, pero entraría dentro de la búsqueda cualquier ontología capaz de representar el modelo funcional del climatizador AHU101, se incluye, por tanto, ifcOWL como ontología candidata al compartir dominio y compatibilidad. Podemos verlas todas en la Tabla 28.

Prefijo	Nombre	URI
SAREF	Smart Applications REFERENCE Ontology	https://saref.etsi.org/core/
SAREF4BLDG	SAREF extension for the Building domain	https://saref.etsi.org/saref4bldg/
IFC	ifcOWL ontology	https://w3id.org/ifc/IFC4_ADD1

Tabla 28. NeOn: selección de ontologías candidatas

8.1.2. Tarea 2. Evaluación de Ontologías de Dominio

Esta tarea comprueba si el conjunto de ontologías de dominio candidatas de la fase anterior son útiles para el desarrollo de la ontología. El equipo de desarrollo debe utilizar para ello los siguientes criterios:

- Comprobar si el alcance y finalidad establecidos en el ORSD son similares a los de las ontologías de dominio candidatas
- Revisar los requisitos de ontología funcional establecidos en la ORSD
- Revisar las CQs incluidas en el ORSD respecto a las ontologías del dominio candidato, teniendo en cuenta los niveles terminológicos y semánticos
- Nivel terminológico: calcular la precisión de los términos de las ontologías del dominio candidato con respecto a la terminología incluida en las CQs
- Nivel Semántico: comprobar si las ontologías de dominio candidatas son capaces de responder a las CQs incluidas en el ORSD

Se procede a revisar si las ontologías seleccionadas en el apartado anterior son capaces de representar todas las partes descritas en el modelo funcional de un climatizador visto en el Apartado 3.1.1 y más concretamente las instanciadas por el modelo AHU101 en estudio, respondiendo siempre a los criterios establecidos sobre el ORSD. La Tabla 29 es el resultado de la evaluación de las ontologías candidatas, que como es de esperar, son todas ya que la ontología SAREF proporciona compatibilidad con IFC y esta última es una ontología superior cuyo dominio incluye al de SAREF y sus extensiones.

Criterio	Ontologías de dominio candidatas		
	SAREF	SAREF4BLDG	IFC
Alcance similar	Sí	Sí	Sí
Semántica similar	Sí	Sí	Sí
Terminología similar	Sí	Sí	Sí
Cobertura de los RF	Sí	Sí	Sí
Cobertura de los RNF	Sí	Sí	Sí

Tabla 29. NeOn, tabla para la valoración de ontologías de dominio

8.1.3. Tarea 3. Selección de Ontología de Dominio

El objetivo de esta tarea es averiguar qué ontologías de dominio son las más adecuadas para el desarrollo de la ontología, tomando como entrada las seleccionadas en la anterior. Para ello se consideran dos aspectos:

- Los criterios definidos por la metodología NeOn en esta tarea
- Las ontologías más utilizadas en el dominio, y las especificadas en los requisitos

Para el primer punto, el equipo de desarrollo debe evaluar las ontologías utilizando los criterios propuestos por NeOn y su peso asociado en función de su importancia, el peso, como valor de escala, que puede ser positivo o negativo. A cada criterio se le asocia un valor cualitativo: Alto, Medio, Bajo o Desconocido, según el nivel de cumplimiento en la ontología evaluada. Posteriormente, estos valores cualitativos se transforman mediante una tabla de equivalencia con valores cuantitativos (5, 3, 1 y 0 respectivamente), que junto al valor peso, permiten el cálculo de un *Score* o puntuación asociada a los pesos negativos y positivos de cada ontología mediante una fórmula ponderada. El *Score* total, sobre el que establecer la selección de las ontologías, sale de restar a las ponderaciones positivas las negativas [85]. La Tabla 30 es el resultado de la aplicación de los criterios sobre las ontologías seleccionadas en la Tarea 2.

Criterio	Peso	Valores		
		SAREF	SAREF4BLDG	IFC
Comprensibilidad de los recursos				
Información externa disponible	+7	alto	alto	alto
Calidad de la documentación	+8	alto	alto	alto
Claridad del código	+8	alto	alto	alto
Esfuerzo de integración				
Sistema de nombrado adecuado	+5	alto	alto	medio
Conflicto entre conocimiento representado	-7	bajo	bajo	medio
Necesidad de términos-puente	-6	bajo	bajo	desconocido
Adaptación al razonamiento	+7	alto	alto	desconocido
Extracción de conocimiento adecuada	+9	alto	alto	desconocido
Lenguaje de implementación adecuado	+7	alto	alto	alto
Fiabilidad de los recursos				
Disponibilidad del test	+8	alto	alto	alto
Reputación del equipo de desarrollo	+8	alto	alto	alto
Evaluación anterior	+8	alto	alto	alto
Fiabilidad del propósito	+3	alto	alto	alto
Soporte práctico	+7	alto	alto	alto
Coste de reutilización				
Coste en tiempo	-7	medio	medio	medio
Coste económico	-9	medio	medio	medio

Tabla 30. NeOn, tabla para la selección de ontologías de dominio

Una simple observación de los valores cualitativos para las tres ontologías ya confirma que tanto SAREF como su extensión SAREF4BLDG tienen valores iguales. En cuanto a IFC, como ontología superior más completa, tiene valores iguales salvo en la parte de integración, ya que requerirá más esfuerzo y recursos su adaptación a un proyecto de dominio más reducido. Con estos datos no se ve

necesaria la transformación en valores cuantitativos ni el cálculo de los Scores asociados y se seleccionan tanto SAREF como SAREF4BLDG como únicas ontologías a tener en cuenta.

Para el segundo punto tenemos como requisitos del proyecto el uso de SAREF y SAREF4BLDG, Apartado 1.2.1, pero además, en el Apartado 3.3 ya se presenta a SAREF como el estándar de preferencia para este tipo de desarrollos.

En la Tabla 31 se detallan algunos de los elementos que se van a reutilizar para el desarrollo de la ontología. Se utiliza el prefijo **s4bldg** para definir la URI de SAREF4BLDG y **saref** para la URI de SAREF. El resto de elementos (clases, propiedades y relaciones) reutilizados se muestran en el Apéndice B: Reutilización de elementos.











rdf:label	rdf:type	URI	rdfs:comment
 Building	owl:Class	s4bldng/Building	A building represents a structure that provides shelter for its occupants or contents and stands in one place...
 Coil	owl:Class	s4bldng/Coil	A coil is a device used to provide heat transfer between non-mixing media...
 Device	owl:Class	saref/Device	A tangible object designed to accomplish a particular task in households, common public buildings or offices...
 Task	owl:Class	saref/Task	The goal for which a device is designed (from a user perspective) ...
 hasValue	owl:DatatypeProperty	saref/hasValue	A relationship defining the value of a certain property, e.g., energy or power
 hasTimestamp	owl:DatatypeProperty	saref/hasTimestamp	A relationship stating the timestamp of an entity (e.g., a measurement).
 contains	owl:ObjectProperty	s4bldng/contains	A relation between a physical space and the objects located in such space.
 hasSpace	owl:ObjectProperty	s4bldng/hasSpace	Relation between a building or a building space and the spaces it can be divided into.
 consistOf	owl:ObjectProperty	saref/consistOf	A relationship indicating a composite entity that consists of other entities...
 offers	owl:ObjectProperty	saref/offers	A relationship between a device and a service

Tabla 31. Muestra de reutilización de elementos

8.1.4. Tarea 4. Integración de ontologías de dominio

Con esta actividad se integran las ontologías de dominio seleccionadas en la ontología que se está desarrollando según los tres modos propuestos por la metodología. El resultado de todas las tareas preliminares confirma el modo de integración total de SAREF y SAREF4BLDG como elementos de dominio base de la ontología desarrollada.

Estas ontologías a su vez importan a otras como recursos ontológicos (Tabla 32) formando la red de ontologías que se muestra en la Ilustración 64.

Prefijo	Nombre	URI
GEO	WGS84 Geo Positioning	http://www.w3.org/2003/01/geo/wgs84_pos
TIME	OWL-Time	http://www.w3.org/2006/time

Tabla 32. Ontologías importadas por SAREF y SAREF4BLDG

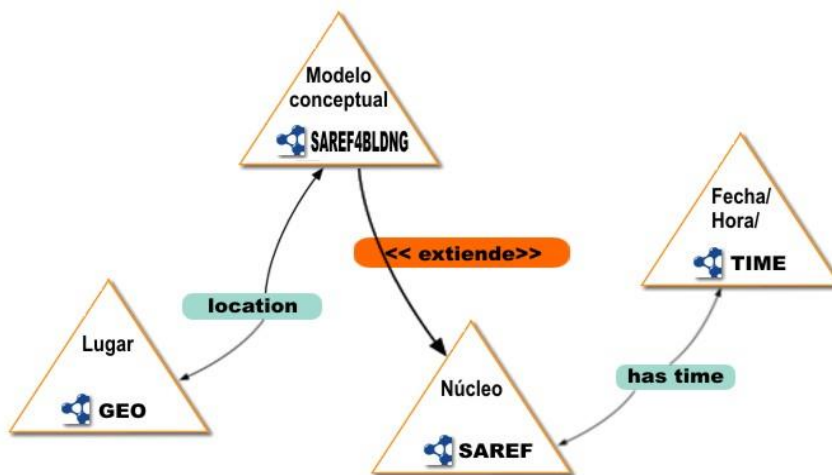


Ilustración 64. Integración de recursos ontológicos

8.2. Reutilización de recursos no ontológicos (NOR)

El objetivo de esta fase es identificar los recursos no ontológicos más apropiados para la construcción de la ontología. La Ilustración 51 del Apartado 4.5.3 muestra la serie de tareas y subtareas para alcanzar el objetivo.

8.2.1. Tarea 1. Búsqueda de recursos no ontológicos

El objetivo de esta tarea es buscar recursos no ontológicos que ayuden a la representación del dominio y a la construcción de la ontología. Debido a limitaciones de seguridad y privacidad, la única fuente de información disponible son un conjunto de ficheros de tipo CSV³² que contienen una serie histórica con medidas de sensores asociados al sistema AHU101 fuente de estudio, se puede ver una muestra en la Ilustración 65:

	A	B	C	D	E	F	G	H	I	J	K
1	id,ReadingTime,HS_101_1_SAF_Enable_DATA	LOG15,HE_147_1_Lecture_Theatre_3_Humidity_DATA	LOG12,TE_2_GFC4_1_Riser_3_Flow_Temp_DATA								
2	1048511,2018-12-04 04:00:00,0,0,35.57564926147461,	22.677305221557617,39.36426544189453,	19.554208755493164,25.399761199951172,0,0,512.81121								
3	1048512,2018-12-04 04:01:00,0,0,35.59758377075195,	22.66918182373047,39.219970703125,	19.55238914489746,25.399824142456055,0,0,512.749267578								
4	1048513,2018-12-04 04:02:00,0,0,35.519203186035156,	22.65974998474121,39.213287353515625,	19.54937171936035,25.40226173400879,0,0,512.297607								
5	1048514,2018-12-04 04:03:00,0,0,35.511714935302734,	22.651147842407227,39.31456756591797,	19.54812240600586,25.406675338745117,0,0,511.04010								
6	1048515,2018-12-04 04:04:00,0,0,35.4631233215332,	22.642351150512695,39.289337158203125,	19.54644012451172,25.41153907775879,0,0,513.2393188								
7	1048516,2018-12-04 04:05:00,0,0,35.464576721191406,	22.6337833404541,39.25542449951172,	19.544395446777344,25.41747283935547,0,0,511.3685607								
8	1048517,2018-12-04 04:06:00,0,0,35.43080520629883,	22.623332977294922,39.2889518737793,	19.542984008789062,25.423480987548828,0,0,511.373077								
9	1048518,2018-12-04 04:07:00,0,0,35.3514404296875,	22.618316650390625,39.3408088684082,	19.542333602905273,25.425111770629883,0,0,511.2917785								
10	1048519,2018-12-04 04:08:00,0,0,35.47620391845703,	22.61004638671875,39.3338508605957,	19.540014266967773,25.429096221923828,0,0,510.3457336								
11	1048520,2018-12-04 04:09:00,0,0,35.581939697265625,	22.603410720825195,39.30508041381836,	19.53874397277832,25.431867599487305,0,0,509.49087								
12	1048521,2018-12-04 04:10:00,0,0,35.62036895751953,	22.595495223999023,39.29438400268555,	19.537328720092773,25.432796478271484,0,0,509.05630								
13	1048522,2018-12-04 04:11:00,0,0,35.578243255615234,	22.5869083404541,39.36853790283203,	19.533798217773438,25.433067321777344,0,0,509.213592								
14	1048523,2018-12-04 04:12:00,0,0,35.62940979003906,	22.579286575317383,39.4365348815918,	19.53343391418457,25.432390213012695,0,0,508.4624328								
15	1048524,2018-12-04 04:13:00,0,0,35.64421463012695,	22.570764541625977,39.54701232910156,	19.53226661682129,25.428421020507812,0,0,508.880676								
16	1048525,2018-12-04 04:14:00,0,0,35.53599548339844,	22.563261032104492,39.56917953491211,	19.532001495361328,25.42388153076172,0,0,508.737792								
17	1048526,2018-12-04 04:15:00,0,0,35.486854553222656,	22.55569076538086,39.58075714111328,	19.52733612060547,25.41678810119629,0,0,509.0553588								
18	1048527,2018-12-04 04:16:00,0,0,35.69383239746094,	22.547088623046875,39.66196060180664,	19.524646759033203,25.40874671936035,0,0,508.677612								

Ilustración 65. Ejemplo de CSV en bruto del edificio Alice Perry

³² CSV: *comma-separated values*, https://en.wikipedia.org/wiki/Comma-separated_values

Y, junto con los archivos CSV, un documento excel con descripciones sobre los sensores y las características de las medidas que figuran en la cabecera de los CSV, Ilustración 66, y que además contiene dos imágenes detalladas con la localización y nombre interno de los sensores dentro de la AHU101.

	A	B	C	D	E	G	H	M	P	Q
1	ans	Site	Building	Floor	Room	Description	Original Measure ID	Unit	Med	Observation Ty
5	175	NUIG North Ca	NUIG Engineerii	Ground Floor		outdoor wind speed	NUIG_AspectGroup_Weather_Current_Wind_Speed	MeterPerSec	Air	WindSpeed
6	176	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	Room G047 average relative humidity	NUIG_AHU101_Ctrls_117_Lecture_Theatre_3_Av_Hi	Percent	Air	Humidity
7	177	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 calculated supply air temper	NUIG_AHU101_Ctrls_AHU101_Calc_Supply_Setpt_D2	DegreeCelsi	Air	Temperature
8	178	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 extract fan velocity percentag	NUIG_AHU101_Ctrls_AHU101_EAF_VSD_Speed_D18	Percent	Object	ContinuousState
9	179	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 number of hours in free cooli	NUIG_AHU101_Ctrls_AHU101_Hours_Free_Cool_D32	Hour	State	RunningTime
10	180	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 number of hours running	NUIG_AHU101_Ctrls_AHU101_Hours_Run_D31_480	Hour	State	RunningTime
11	181	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 supply fan velocity percentag	NUIG_AHU101_Ctrls_AHU101_SAF_VSD_Speed_D17	Percent	Object	ContinuousState
12	182	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	Room G047 indoor CO2 concentration	NUIG_AHU101_Ctrls_CO_147_1_Lecture_Theatre_3_	PartsPerMill	Air	Co2Rate
13	183	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	Room G047 indoor CO2 concentration	NUIG_AHU101_Ctrls_CO_147_2_Lecture_Theatre_3_	PartsPerMill	Air	Co2Rate
14	185	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 return duct air relative humic	NUIG_AHU101_Ctrls_HE_101_1_Return_Duct_Humid	Percent	Air	Humidity
15	186	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	Room G047 indoor air relative humid	NUIG_AHU101_Ctrls_HE_147_1_Lecture_Theatre_3_	Percent	Air	Humidity
16	187	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	Room G047 indoor air relative humid	NUIG_AHU101_Ctrls_HE_147_2_Lecture_Theatre_3_	Percent	Air	Humidity
17	188	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 supply fan enable status	NUIG_AHU101_Ctrls_HS_101_1_SAF_Enable_D15_45	Number	Object	DiscreteState
18	189	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 extract fan enable status	NUIG_AHU101_Ctrls_HS_101_2_EAF_Enable_D16_47	Number	Object	DiscreteState
19	190	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	Room G047 average CO2 concentratio	NUIG_AHU101_Ctrls_Lecture_Theatre_3_Avg_CO2_D	PartsPerMill	Air	Co2Rate
20	191	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	Room G047 average air temperature	NUIG_AHU101_Ctrls_Lecture_Theatre_3_Avg_Temp	DegreeCelsi	Air	Temperature
21	192	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 percentage opening of valve	NUIG_AHU101_Ctrls_LPHW_2_GFC4_Riser_3_IFM_D1	1_467		
22	193	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 fire alarm	NUIG_AHU101_Ctrls_MCC02_Fire_Alarm_D29_476	Number	Object	DiscreteState
23	194	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 percentage opening of valve	NUIG_AHU101_Ctrls_TCV_101_1_Main_HValve_D23	Percent	Object	ContinuousState
24	195	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 percentage opening of valve	NUIG_AHU101_Ctrls_TCV_101_2_Main_CValve_D25	Percent	Object	ContinuousState
25	196	NUIG North Ca	NUIG Engineerii	Ground Floor	G047 Lecture	AHU101 percentaee openine of valve	NUIG_AHU101_Ctrls_TCV_101_3_Frost_Coll_HValve	Percent	Obiect	ContinuousState

Ilustración 66. Segmento del excel con anotaciones sobre los sensores

Estos documentos son, por tanto, los únicos documentos disponibles y los tratamos como recursos no ontológicos a reutilizar.

8.2.2. Tarea 2. Valoración de recursos no ontológicos

Esta tarea cubre la valoración del conjunto de recursos no ontológicos candidatos, resultado del anterior apartado. Para alcanzar su objetivo se subdivide en cinco pasos:

1. Extraer las entradas léxicas
2. Calcular la precisión
3. Calcular la cobertura
4. Evaluar en consenso
5. Construir la tabla de valoración

Disponemos de dos fuentes de datos relacionadas como único recurso a valorar y además son también la única fuente de referencia en el ORSD por lo que se concluye que su estructura tanto a nivel de precisión como cobertura es aceptable para su reutilización.

De los documentos CSV obtendremos el mapeado de las medidas de cada sensor identificadas en la cabecera, y las fechas y valores de lectura indicados en el cuerpo. Con el excel identificaremos cada cabecera con el tipo de sensor, sus características y localización en el modelo conceptual de la AHU101. Esta información será útil sobre todo en la parte de implementación de la ontología, Apartado 8.5, para unificar los nombres de las instancias de cada componente y sus características con los de los datos de cabecera del CSV. Los componentes y características extraídos de los datos se muestran en la Tabla 33.

Componentes	
Supply Air Fan	Off Coil Cooling Duct Temperature Sensor
SAF Variable Speed Drive	Pre-Heat Valve and Actuator
SAF Airflow Differential Pressure Switch	Thermal Wheel
Extract Air Fan	Frost Stat (located after Pre-Heat coil)
Extract Variable Speed Drive	Supply Panel Filter Differential Pressure Switch
Extract Airflow Differential Pressure Switch	Supply Bag Filter Differential Pressure Switch
Supply Duct Temperature Sensor	Extract Bag Filter Differential Pressure Switch
Return Duct Temperature Sensor	Lecture Theatre 03 Space Temperature Sensors
Off Coil Pre-Heat Duct Temperature Sensor	Lecture Theatre 03 Carbon Dioxide Sensors
Heating Valve and Actuator	Lecture Theatre 03 Space Humidity Sensor
Cooling Valve and Actuator	Return Duct Humidity Sensor
Características	
Temperature Control	Filter Monitoring
Relative Humidity Control	Frost Monitoring
Air Quality/CO2 Control	Thermistor Trip Monitoring

Tabla 33. Componentes y características de la AHU101

8.2.3. Tarea 3. Selección de los recursos no ontológicos más apropiados

En esta última actividad se seleccionan los recursos no ontológicos más apropiados para la construcción de la ontología. De las dos tipologías de archivos que tenemos, el excel nos proporciona información sobre los nombres y localizaciones para las instancias que resulten del modelado de la AHU101, con los elementos del apartado anterior. Estos datos son de utilidad en la implementación de la ontología, pero también facilita el tipo y rango de los valores que se miden, y estas mediciones se encuentran en los archivos CSV que contienen el mapeado de los datos de la AHU101. Se seleccionarán, por tanto, ambos recursos.

8.3. Reingeniería de recursos no ontológicos

Finalizado el proceso de selección de los recursos no ontológicos, se procede a su tratamiento y adaptación para su empleo en la ontología, cuyas actividades y tareas se detallan en la Ilustración 57 del Apartado 4.5.9.

8.3.1. Actividad 1. Ingeniería inversa de recursos no ontológicos

Tomando como entrada los recursos no ontológicos se deben identificar sus componentes subyacentes y crear de una representación del recurso a niveles altos de abstracción. Para ello, se realizan las tareas:

1. Recopilación de datos
2. Abstracción conceptual
3. Exploración de la información

La recopilación de archivos se hace de manera automática mediante un software capaz de realizar tratamiento de ficheros CSV, dentro de los requisitos del proyecto se incluye que este software sea versátil y permita ingestas de datos en otros formatos, incluyendo datos de *streaming* por si fuera posible la conexión directa al Data Lake del Alice Perry Engineering Building en algún desarrollo futuro.

La abstracción conceptual de los datos ya viene dada por los resultados de la fase anterior, en la Tabla 33 podemos ver los componentes y características a las que se asocian los datos contenidos en los ficheros, y como se relacionan con nuestro modelo conceptual de un sistema AHU. Los archivos CSV contienen el mapeo de sensor-medición asociado con una fecha y hora concreta.

Con los elementos de las cabeceras CSV y sus correspondencias con los sensores y tipos de variables contrastados en el excel, se elabora la descripción de la Tabla 34.

ID	CAMPO	TIPO DATO	DESCRIPCIÓN
C01	Id	Integer	Identificador de registro
C02	ReadingTime	TimeStamp	Fecha y hora de lectura del registro
C03	HS_101_1_SAF_Enable_DATALOG15	Integer	AHU101 estado de habilitación del ventilador de suministro
C04	HE_147_1_Lecture_Theatre_3_Humidity_DATALOG12	Double	Sala G047 humedad relativa del aire interior, posición 1
C05	TE_2_GFC4_1_Riser_3_Flow_Temp_DATALOG27	Double	AHU101 temperatura del flujo de agua del elevador GFC4
C06	HE_101_1_Return_Duct_Humidity_DATALOG13	Double	AHU101 humedad relativa del aire en el conducto de retorno
C07	TE_102_2_Cooling_Off_Coil_Temp_DATALOG6	Double	AHU101 temperatura del aire después del serpentín de enfriamiento
C08	TE_2_GFC4_2_Riser_3_Return_Temp_DATALOG30	Double	AHU101 temperatura de retorno del agua del tubo ascendente GFC4
C09	AHU101_EAF_VSD_Speed_DATALOG18	Double	AHU101 porcentaje de velocidad del ventilador extractor
C10	CO_147_1_Lecture_Theatre_3_CO2_DATALOG10	Double	Sala G047 concentración de CO ₂ interior, posición 1
C11	TE_101_1_Frost_Coil_Temp_DATALOG3	Double	AHU101 temperatura del aire después del serpentín de precalentamiento
C12	HE_147_2_Lecture_Theatre_3_Humidity_DATALOG9	Double	Sala G047 humedad relativa del aire interior, posición 2
C13	TE_147_2_Lecture_Theatre_3_Temp_DATALOG8	Double	Sala G047 temperatura del aire interior, posición 2
C14	CO_147_2_Lecture_Theatre_3_CO2_DATALOG7	Double	Sala G047 concentración de CO ₂ interior, posición 2
C15	LPHW_2_GFC4_Riser_3_IFM_DATALOG21	Double	AHU101 porcentaje de apertura de la válvula del tubo ascendente GFC4
C16	TCV_101_2_Main_CVValve_DATALOG25	Double	AHU101 porcentaje de apertura de la válvula del serpentín de refrigeración
C17	TE_147_1_Lecture_Theatre_3_Temp_DATALOG11	Double	Sala G047 temperatura del aire interior, posición 1
C18	AHU101_Calc_Supply_Setpt_DATALOG22	Double	AHU101 cálculo de la temperatura del aire de suministro
C19	TCV_101_3_Frost_Coil_HVValve_DATALOG14	Double	AHU101 porcentaje de apertura de la válvula del serpentín de precalentamiento

C20	TE_101_2_Supply_Duct_Temp_DATALOG4	Double	AHU101 temperatura del aire del conducto de suministro
C21	AHU101_SAF_VSD_Speed_DATALOG17	Double	AHU101 porcentaje de velocidad del ventilador de suministro
C22	TE_101_4_Return_Duct_Temp_DATALOG20	Double	AHU101 temperatura del aire del conducto de retorno
C23	TCV_101_1_Main_HVValve_DATALOG23	Double	AHU101 porcentaje de apertura de la válvula del serpentín de postcalentamiento
C24	MCC02___Fire_Alarm_DATALOG29	Integer	AHU101 alarma de incendio
C25	Lecture_Theatre_3_Avg_Temp_DATALOG19	Double	Sala G047 temperatura media del aire
C26	TE_101_3_Cooling_Off_Coil_Temp_DATALOG5	Double	AHU101 temperatura del aire después del serpentín de enfriamiento
C27	HS_101_2_EAF_Enable_DATALOG16	Integer	AHU101 estado de habilitación del ventilador de extracción
C28	AHU101_Hours_Run_DATALOG31	Double	AHU101 número de horas funcionando
C29	Lecture_Theatre_3_Avg_CO2_DATALOG26	Double	Sala G047 concentración media de CO ₂
C30	117_Lecture_Theatre_3_Av__Humidity_DATALOG28	Double	Sala G047 humedad relativa media
C31	GF_East_Rads_Energy_Acc_Total_DATALOG2	Double	deshabilitado
C32	AHU101_Hours_Free_Cool_DATALOG32	Double	AHU101 número de horas en refrigeración libre

Tabla 34. Campos asociados al CSV de entrada

Se decide leer y tratar todos los campos analizados siguiendo el requerimiento de facilitar trabajos futuros con la disponibilidad de toda la información posible. De la observación de los datos del excel, y la propia Tabla 34, se deduce la estructura subyacente o metadatos de cada fila del CSV:

- Nombre del campo tratado (la cabecera de su columna)
- Tipo de dato contenido (Integer, TimeStamp o Double)
- Posibilidad de haber nulos en el dato (cierto ya que no se indica lo contrario)
- Descripción del campo (aportada por la correspondencia de la cabecera con el excel)

8.3.2. Actividad 2. Transformación de los recursos no ontológicos

Es necesario generar un modelo conceptual ontológico a partir de los recursos no ontológicos. Mediante los recursos proporcionados por la ontología, ver Apartado 4.6.3, se accede a una biblioteca con patrones nombrados bajo el prefijo PR-NOR. El objetivo es averiguar si existe algún patrón aplicable para transformar el recurso en un modelo conceptual.

Si se encuentra un patrón adecuado siguiendo las recomendaciones del modelo (Apartado 4.5.9), entonces se crea el modelo conceptual, mediante el procedimiento establecido en la biblioteca, de lo contrario, tenemos que establecer un procedimiento ad-hoc para transformar el recurso no ontológico en un modelo conceptual. Este procedimiento ad-hoc puede generalizarse para crear un nuevo patrón para

la reingeniería de recursos no ontológicos. En el caso de los NOR objeto del proyecto, dada su sencillez y con los metadatos obtenidos de su análisis, es muy fácil establecer la relación existente entre los mismos y el modelo de datos definido por la ontología a desarrollar, y más teniendo en cuenta las integraciones de las ontologías SAREF y SAREF4BLDG.

Antes de aplicar el proceso ad-hoc se ve necesario aplicar un proceso ETL³³ típico de *Machine Learning* para mejorar los datos iniciales (datos en crudo) y obtener un conjunto final estructurado y estable (datos transformados), adaptados a las características exactas que precisa la aplicación objeto de desarrollo de esta fase.

Como parte anterior al proceso ETL y para poder tratar los datos automáticamente se profundiza en la estructura de metadatos observada en la tarea anterior. Como el objetivo final es el mapeado de medidas, se prescinde de la descripción del campo y se añade, para futuros desarrollos un campo genérico que se deja vacío. La estructura quedará definida de la siguiente manera:

- nombre de la etiqueta: campo "**name**"
- tipo de variable: campo "**type**"
- puede contener nulos: campo "**nullable**"
- otros metadatos: campo "**metadata**"

El formato elegido para definir la estructura de metadatos es JSON-LD³⁴ al ser un formato ligero, de uso estandarizado tanto en entornos de bases de datos como en la Web Semántica para los documentos *Linked Data* (Apartado 2.3), por lo que asegura compatibilidad con cualquier software que se utilice. Se genera el archivo "**schema.json**", Ilustración 67 con los metadatos de todos los campos de la Tabla 34.

```
{
  "type": "struct",
  "fields": [
    {"name": "id", "type": "integer", "nullable": true, "metadata": {}},
    {"name": "ReadingTime", "type": "timestamp", "nullable": true, "metadata": {}},
    {"name": "HS_101_1_SAF_Enable_DATALOG15", "type": "string", "nullable": true, "metadata": {}},
    {"name": "HE_147_1_Lecture_Theatre_3_Humidity_DATALOG12", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_2_GFC4_1_Riser_3_Flow_Temp_DATALOG27", "type": "double", "nullable": true, "metadata": {}},
    {"name": "HE_101_1_Return_Duct_Humidity_DATALOG13", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_102_2_Cooling_Off_Coil_Temp_DATALOG6", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_2_GFC4_2_Riser_3_Return_Temp_DATALOG30", "type": "double", "nullable": true, "metadata": {}},
    {"name": "AHU101_EAF_VSD_Speed_DATALOG18", "type": "double", "nullable": true, "metadata": {}},
    {"name": "CO_147_1_Lecture_Theatre_3_CO2_DATALOG10", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_101_1_Frost_Coil_Temp_DATALOG3", "type": "double", "nullable": true, "metadata": {}},
    {"name": "HE_147_2_Lecture_Theatre_3_Humidity_DATALOG9", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_147_2_Lecture_Theatre_3_Temp_DATALOG8", "type": "double", "nullable": true, "metadata": {}},
    {"name": "CO_147_2_Lecture_Theatre_3_CO2_DATALOG7", "type": "double", "nullable": true, "metadata": {}},
    {"name": "LPHW_2_GFC4_Riser_3_IFM_DATALOG21", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TCV_101_2_Main_CValve_DATALOG25", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_147_1_Lecture_Theatre_3_Temp_DATALOG11", "type": "double", "nullable": true, "metadata": {}},
    {"name": "AHU101_Calc_Supply_Setpt_DATALOG22", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TCV_101_3_Frost_Coil_HValve_DATALOG14", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_101_2_Supply_Duct_Temp_DATALOG4", "type": "double", "nullable": true, "metadata": {}},
    {"name": "AHU101_SAF_VSD_Speed_DATALOG17", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_101_4_Return_Duct_Temp_DATALOG20", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TCV_101_1_Main_HValve_DATALOG23", "type": "double", "nullable": true, "metadata": {}},
    {"name": "MCC02_Fire_Alarm_DATALOG29", "type": "string", "nullable": true, "metadata": {}},
    {"name": "Lecture_Theatre_3_Avg_Temp_DATALOG19", "type": "double", "nullable": true, "metadata": {}},
    {"name": "TE_101_3_Cooling_Off_Coil_Temp_DATALOG5", "type": "double", "nullable": true, "metadata": {}},
    {"name": "HS_101_2_EAF_Enable_DATALOG16", "type": "string", "nullable": true, "metadata": {}},
    {"name": "AHU101_Hours_Run_DATALOG31", "type": "double", "nullable": true, "metadata": {}},
    {"name": "Lecture_Theatre_3_Avg_CO2_DATALOG26", "type": "double", "nullable": true, "metadata": {}},
    {"name": "117_Lecture_Theatre_3_Av_Humidity_DATALOG28", "type": "double", "nullable": true, "metadata": {}},
    {"name": "GF_East_Rads_Energy_Acc_Total_DATALOG2", "type": "double", "nullable": true, "metadata": {}},
    {"name": "AHU101_Hours_Free_Cool_DATALOG32", "type": "double", "nullable": true, "metadata": {}},
  ]
}
```

Ilustración 67. Metadatos de los CSV en JSON-LD

³³ ELT: Extract, Load, Transform; <https://docs.microsoft.com/es-es/azure/architecture/data-guide/relational-data/etl>

³⁴ JSON-LD, <https://json-ld.org/>

El proceso ETL comienza con la carga en bloque de los archivos CSV mediante Apache Spark, herramienta ya descrita en Apartado 7.8, descartando las cabeceras de los mismos, eliminando errores que puedan generar caracteres extraños o valores nulos, o conjuntos de datos que no guarden el formato correcto. En la misma orden de carga de archivos, se emplea el archivo “**schema.json**” para forzar la correcta detección de formato en cada columna, especificando para el caso de las fechas en los datos tipo *timestamp* la descripción americana tal y como figura en las cadenas de texto de los CSV. Los datos son almacenados en una estructura denominada de tipo *dataframe* habitual en este tipo de herramientas, y destinada a la manipulación de grandes volúmenes de datos. Con los datos ya preprocesados y disponibles en el sistema, se continúa con el proceso de limpieza descartando duplicados, tanto por equivalencia exacta entre todas las columnas, como por similitud entre las columnas con identificador único DT01 (**‘id’**) o DT02 (**‘ReadingTime’**).

Con los datos ya limpios, se realizan tareas de transformación de datos, asegurándonos de la correcta discretización de las variables con carácter lógico. Al provenir los datos de sensores digitales, tiene bastante importancia que mantengan valores enteros de ‘0’ o ‘1’ por lo que se asegura que los datos asociados al valor de DT03(**‘HS_101_1_SAF_Enable_DATALOG15’**), el valor DT24 (**‘MCC02__Fire_Alarm_DATALOG29’**) y el DT27 (**‘HS_101_2_EAF_Enable_DATALOG16’**) sean enteros y no se hayan transformado en lógicos por el software de detección automática. Como la carga de archivos en sistema puede ser múltiple, se ordenan mediante el campo DT02 (**‘ReadingTime’**). Este paso no es realmente necesario para el desarrollo de la aplicación ad-hoc, pero si en algún momento se hace un volcado de los datos en formato JSON, el archivo resultante tendría más coherencia y legibilidad.

En este punto se tiene una representación final de los datos, ordenados temporalmente, en una lista de elementos JSON donde cada elemento de la lista correspondería a una fila de los archivos CSV originales. A la hora de manipular el acceso a los diferentes campos se decide, por comodidad, organizarlos en una estructura de diccionario con pares (clave, valor) donde la clave corresponde al campo “**name**” con el nombre de la columna cabecera y el valor a su contenido, teniendo acceso directo al valor de cada elemento solamente con indicar su nombre. Esta fase emplea la biblioteca de recursos JSON para Python descrita en el Apartado 7.9.

En este punto ya podemos aplicar el proceso ad-hoc a los NOR que va a consistir en una población de las medidas de los dispositivos. Cada elemento de nuestra cadena JSON corresponde a una medición de los diferentes dispositivos en su marca temporal, cada dispositivo es un campo “**name**” salvo los campos informativos **‘id’** y **‘ReadingTime’** que se utilizarán para construir las URI, y el campo en desuso **‘GF_East_Rads_Energy_Acc_Total’** que no se utilizará.

El patrón propuesto por SAREF para las medidas junto con su tabla de restricciones se muestran en la Ilustración 40 del Apartado 3.3. Este modelo representa un patrón n-ario que permite relacionar las medidas para diferentes propiedades que se miden en diferentes unidades. Es decir, la clase *saref:Measurement* apunta a describir una medida de una cantidad física (usando la propiedad *saref:hasValue*) para un determinado *saref:Propiedad* y según una determinada *saref:UnidadDeMedida*.^[83]. Las propiedades y las unidades de medida ya las tendremos instanciadas en nuestra ontología (ver Apartado 8.4.2), tan solo tenemos que poblar instanciando la medida (“**Measurement**”) con su valor (“**hasValue**”) y su valor temporal (“**dateTime**”), para luego vincularla al resto de elementos de la ontología, a través de la propiedad que mide (“**Property**”) que también nos servirá para asignar correctamente las unidades de medida (“**UnitOfMeasure**”).

La Ilustración 68 muestra un ejemplo de transformación para un dato temporal, en este caso el C18 cuyo detalle figura en la Tabla 34, su instanciación en la ontología, y como quedaría serializado en el archivo RDF/XML de la ontología. Se amplía este proceso en el Capítulo 9.

Cadena JSON

```
'ReadingTime': '2018-12-04T04:00:00.000+01:00', 'HS_101_1_SAF_Enable_DATALOG15': 0, 'HE_1
XG12': 35.57564926147461, 'TE_2_GFCA_1_Riser_3_Flow_Temp_DATALOG27': 22.677305221557613,
13': 39.36426544189453, 'TE_102_2_Cooling_Off_Coil_Temp_DATALOG6': 19.554208755491464, 'TE
10': 25.399761199951172, 'AHU101_EAF_VSD_Speed_DATALOG18': 0.0, 'CO_147_1_Lecture_Theatre_
TE_101_1_Frost_Coil_Temp_DATALOG3': 9.233405113220215, 'HE_147_2_Lecture_Theatre_3_Humidi
147_2_Lecture_Theatre_3_Temp_DATALOG8': 19.349262237548828, 'CO_147_2_Lecture_Theatre_3_C
W_2_GFCA_Riser_3_IPM_DATALOG21': 112.1878695678711, 'TCV_101_2_Main_CV_Valve_DATALOG25': 99
DATALOG11': 19.461599349975586, 'AHU101_Calc_Supply_Setpt_DATALOG22': 18.0, 'TCV_101_3_Fre
11_2_Supply_Duct_Temp_DATALOG4': 13.640135765075684, 'AHU101_SAF_VSD_Speed_DATALOG17': 0.0
3': 17.224287033081055, 'TCV_101_1_Main_HValve_DATALOG23': 15.378217697143555, 'MCC02_Fi
vtre_3_Avg_Temp_DATALOG19': 18.678354263305664, 'TE_101_3_Cooling_Off_Coil_Temp_DATALOG5':
sble_DATALOG16': 0, 'AHU101_Hours_Run_DATALOG31': 12786.0, 'Lecture_Theatre_3_Avg_CO2_DATA
re_Theatre_3_Av_Humidity_DATALOG28': 37.193309978393555, 'GF_East_Rads_Energy_Acc_Total_D
we_Cool_DATALOG32': 2126.0)
```

'AHU101_Calc_Supply_Setpt_DATALOG22': 18.0,

Instancia RDF/XML en la ontología

```
<core:Measurement rdf:about="https://saref.etsi.org/core/Measure_1048511_AHU101_Calc_Supply_Setpt_2018_12_04T04_00_00_000_01_00">
<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#NamedIndividual"/>
<core:relatesToProperty rdf:resource="http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_AHU101_Ctrls_AHU101_Calc_Supply_Setpt_D22_470">
<core:isMeasuredIn rdf:resource="http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#CelsiusDegrees"/>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Measure AHU101 Calc Supply Setpt 2018-12-04T04:00:00.000+01:00</rdfs:label>
<core:hasTimeStamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2018-12-04T04:00:00+01:00</core:hasTimeStamp>
<core:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">18.0</core:hasValue>
</core:Measurement>
```



Ilustración 68. Transformación de población para un NOR en la ontología

Con todos estos datos disponibles, se diseña el siguiente algoritmo de transformación:

- Para cada <elemento> (medida) en la **cadena** JSON
- Generamos la <uri> de manera unívoca con el 'id' y 'ReadingTime'
- Si la <uri> no está en <world> (**Base de Conocimiento Ontológico**) entonces
 - Crear nueva instancia: <instancia> desde la clase **Measurement** con la <uri>
 - Asignar a <instancia> una anotación, etiqueta **label**, con un nombre basado en <uri>
 - Asignar a <instancia> la *data property* "**has value**" con <valor> de <elemento>
 - Asignar a <instancia> la *data property* "**has timestamp**" con <fecha> de <elemento>
 - Acceder a la instancia de clase **Property**, <propiedad> que define la medida por <uri>
 - Asignar la *object property* "**relates to property**" del <elemento> a la <propiedad>
 - Si la <propiedad> es de clase **Temperature**
 - Asignar la *object property* "**is measured in**" con la instancia "**Celsius Degrees**"
 - Si la <propiedad> es de clase **Humidity**
 - Asignar la *object property* "**is measured in**" con la instancia "**PercentageRH**"
 - Si la <propiedad> es de clase **Occupancy**
 - Asignar la *object property* "**is measured in**" con la instancia "**Percentage**"
 - Si la <propiedad> es de clase **Smoke**
 - Asignar la *object property* "**is measured in**" con la instancia "**PartsPerMillion**"
 - Si la <propiedad> es de clase **Status**
 - Asignar la *object property* "**is measured in**" con la instancia "**Boolean**"
 - Si la <propiedad> es de clase **Time**
 - Asignar la *object property* "**is measured in**" con la instancia "**Hour**"

Fin-Si
Fin-Para

8.3.3. Actividad 3. Ingeniería hacia adelante de la ontología

El objetivo es generar la ontología en base a los niveles de abstracción ontológicos definidos en la tarea anterior e integrarlos con el proceso de desarrollo ontológico general. Por tanto, una vez realizada la transformación de cada recurso no ontológico se integra directamente como instancia ya que solamente realizamos transformaciones de población de los datos, ver Ilustración 62 del Apartado 4.6.3.

8.4. Formalización de la ontología

En este apartado se realiza la construcción de la ontología proyectada, y es el resultado de la composición de los escenarios 1, 2 y 3 propuestos por la metodología NeOn, aplicados al contexto de la unidad de climatización AHU101 del Alice Perry Engineering Building. Se puede consultar una versión más detallada del diagrama de la ontología en el [Apéndice C](#) o en su versión navegable en HTML en la documentación en el gitHub del proyecto, consultar [Apéndice A](#).

8.4.1. Diseño de las URIs

Existe una recomendación de W3C sobre el uso de las *Dereferencing HTTP URI* especialmente pensado para distinguir vocabulario y recursos en entornos de Datos enlazados. Básicamente proponen dos formas de construcción las *URI-Hash* y las *URI-Slash* o 303. Las *URI-Hash* se utilizan para diferenciar los recursos ontológicos de los www comunes, mediante un fragmento final de su URI que comienza por “#” identificando el recurso. La segunda opción emplea el carácter “/” para identificar el recurso ontológico, por lo que realmente no es distinguible de una URL web. La ventaja de la primera es que siempre se distinguirá el tipo de petición al servidor y dar un documento RDF, mientras que la segunda puede tardar algo más si la responde otro servicio primero, como puede ser un REST. La ventaja de la segunda es que, para cada recurso, se puede configurar por separado un objeto de redirección, facilitando el uso distribuido cuando hay grandes cantidades de datos.

La ontología del proyecto utilizará la URI-Hash ya que es la más empleada en los recursos ontológicos embebidos de SAREF y SAREF4BLD. Es una ontología limitada, de uso local que no precisa de un despliegue distribuido y sí más eficiencia en las pruebas. A continuación, se muestra el esquema y algunos ejemplos aplicados en la Tabla 35.

PATRÓN:	HTTP://{URI BASE}#{CLASE PROPIEDAD INSTANCIA}
URI BASE	https://www.uva.es/gii/tfg/angnava/ontology
Ejemplos	
ONTOLOGÍA	https://www.uva.es/gii/tfg/angnava/ontology#NUIG_AHU101OpenValveCoolingCoil
SAREF	https://saref.etsi.org/core/OpenCloseState
SAREF4BLD	https://saref.etsi.org/saref4bldg/operationTemperatureMax

Tabla 35. Esquema y ejemplos de URIs

8.4.2. Vocabulario

El Apartado 8.1 generó como resultado todos los elementos de dominio necesarios para la instanciación de la AHU101 mediante las clases y propiedades de SAREF y SAREF4BLDG y se pueden consultar en el [Apéndice B](#). Estos elementos dan soporte al edificio Alice Perry, su estructura y dispositivos, se pueden comprobar fácilmente las descripciones de nivel superior en las ilustraciones del Apartado 3.3 y 3.4 dedicadas a los estándares.

A nivel particular, para instanciar cada dispositivo que contine las medidas a mapear, contamos con la información contenida en el excel: por un lado, obtendremos la localización de los dispositivos y elementos de la AHU101 en el edificio, y por otro, la clasificación de dichos elementos en función de si son Actuadores(“**actuador**”), Sensores (“**sensor**”) o interruptores (“**position**”). Se pueden observar alguno de estos detalles en la Ilustración 69 y la Ilustración 70.

11	AHU101 supply fan velocity percentage	NUIG_AHU101_Ctrls_AHU101_SAF_VSD_Speed_D17_473	AHU101_SAF_VSD_Speed_DATALOG17	Percent	Actuador
12	Room G047 indoor CO2 concentration position 1	NUIG_AHU101_Ctrls_CO_147_1_Lecture_Theatre_3_CO2_D10_462	CO_147_1_Lecture_Theatre_3_CO2_DATALOG10	PartsPerMillion	Sensor
13	Room G047 indoor CO2 concentration position 2	NUIG_AHU101_Ctrls_CO_147_2_Lecture_Theatre_3_CO2_D7_466	CO_147_2_Lecture_Theatre_3_CO2_DATALOG7	PartsPerMillion	Sensor
14	AHU101 return duct air relative humidity	NUIG_AHU101_Ctrls_HE_101_1_Return_Duct_Humidity_D13_458	HE_101_1_Return_Duct_Humidity_DATALOG13	Percent	Sensor
15	Room G047 indoor air relative humidity position 1	NUIG_AHU101_Ctrls_HE_147_1_Lecture_Theatre_3_Humidity_D12_456	HE_147_1_Lecture_Theatre_3_Humidity_DATALOG12	Percent	Sensor
16	Room G047 indoor air relative humidity position 2	NUIG_AHU101_Ctrls_HE_147_2_Lecture_Theatre_3_Humidity_D9_464	HE_147_2_Lecture_Theatre_3_Humidity_DATALOG9	Percent	Sensor
17	AHU101 supply fan enable status	NUIG_AHU101_Ctrls_HS_101_1_SAF_Enable_D15_455	HS_101_1_SAF_Enable_DATALOG15	Number	posición
18	AHU101 extract fan enable status	NUIG_AHU101_Ctrls_HS_101_2_EAF_Enable_D16_479	HS_101_2_EAF_Enable_DATALOG16	Number	posición
19	Room G047 average CO2 concentration	NUIG_AHU101_Ctrls_Lecture_Theatre_3_Avg_CO2_D26_481	Lecture_Theatre_3_Avg_CO2_DATALOG26	PartsPerMillion	Sensor
20	Room G047 average air temperature	NUIG_AHU101_Ctrls_Lecture_Theatre_3_Avg_Temp_D19_477	Lecture_Theatre_3_Avg_Temp_DATALOG19	DegreeCelsius	Sensor
21	AHU101 percentage opening of valve of GFC4 riser	NUIG_AHU101_Ctrls_LPHW_2_GFC4_Riser_3_IFM_D21_467	LPHW_2_GFC4_Riser_3_IFM_DATALOG21	Percent	Actuador
22	AHU101 fire alarm	NUIG_AHU101_Ctrls_MCC02_Fire_Alarm_D29_476	MCC02_Fire_Alarm_DATALOG29	Number	posición
23	AHU101 percentage opening of valve of post-heating coil	NUIG_AHU101_Ctrls_TCV_101_1_Main_HValve_D23_475	TCV_101_1_Main_HValve_DATALOG23	Percent	Actuador
24	AHU101 percentage opening of valve of cooling coil	NUIG_AHU101_Ctrls_TCV_101_2_Main_CValve_D25_468	TCV_101_2_Main_CValve_DATALOG25	Percent	Actuador
25	AHU101 percentage opening of valve of pre-heating coil	NUIG_AHU101_Ctrls_TCV_101_3_Frost_Coil_HValve_D14_471	TCV_101_3_Frost_Coil_HValve_DATALOG14	Percent	Actuador
26	AHU101 air temperature after pre-heating coil	NUIG_AHU101_Ctrls_TE_101_1_Frost_Coil_Temp_D3_463	TE_101_1_Frost_Coil_Temp_DATALOG3	DegreeCelsius	Sensor
27	AHU101 supply duct air temperature	NUIG_AHU101_Ctrls_TE_101_2_Supply_Duct_Temp_D4_472	TE_101_2_Supply_Duct_Temp_DATALOG4	DegreeCelsius	Sensor
28	AHU101 air temperature after cooling coil	NUIG_AHU101_Ctrls_TE_101_3_Cooling_Off_Coil_Temp_D5_478	TE_101_3_Cooling_Off_Coil_Temp_DATALOG5	DegreeCelsius	Sensor
29	AHU101 return duct air temperature	NUIG_AHU101_Ctrls_TE_101_4_Return_Duct_Temp_D20_474	TE_101_4_Return_Duct_Temp_DATALOG20	DegreeCelsius	Sensor
30	AHU101 air temperature after cooling coil	NUIG_AHU101_Ctrls_TE_102_2_Cooling_Off_Coil_Temp_D6_459	TE_102_2_Cooling_Off_Coil_Temp_DATALOG6	DegreeCelsius	Sensor
31	Room G047 indoor air temperature position 1	NUIG_AHU101_Ctrls_TE_147_1_Lecture_Theatre_3_Temp_D11_469	TE_147_1_Lecture_Theatre_3_Temp_DATALOG11	DegreeCelsius	Sensor
32	Room G047 indoor air temperature position 2	NUIG_AHU101_Ctrls_TE_147_2_Lecture_Theatre_3_Temp_D8_465	TE_147_2_Lecture_Theatre_3_Temp_DATALOG8	DegreeCelsius	Sensor

Ilustración 69. Detalle del excel con la columna de tipo de dispositivo

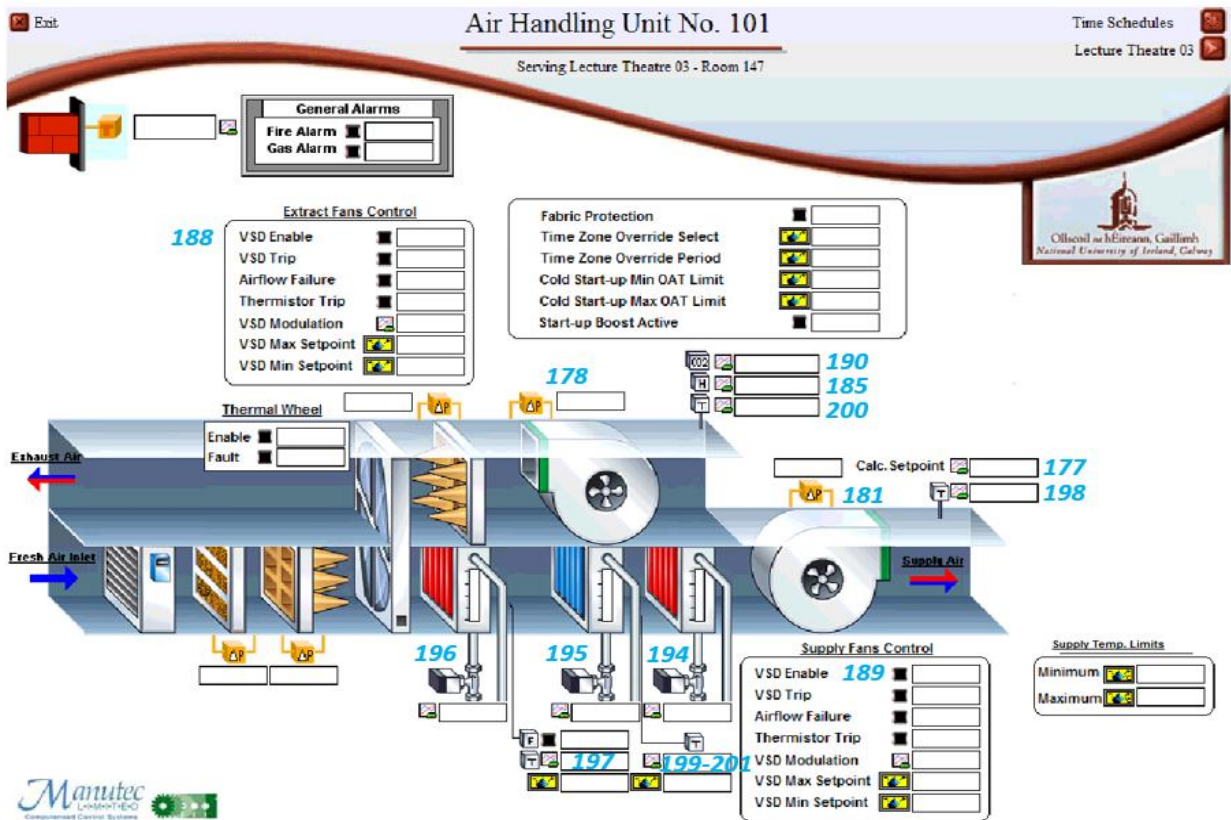


Ilustración 70. Detalle del excel con el esquema gráfico AHU101

El patrón de nombrado junto con algunos ejemplos de instancias principales se muestra en la Tabla 36:

URI Base:	https://www.uva.es/gii/tfg/angnava/ontology
Patrón	
Para localizaciones:	http://{URI base}#NUIG_Engineering_Building_{nombre}
Para controles y dispositivos en la habitación sobre la que actúa la AHU101:	http://{URI base}#NUIG_G047_{nombre}
Para los dispositivos, controles, localizaciones, servicios y medidas de la AHU101:	http://{URI base}#NUIG_AHU101_{Actuator Component Ctrls Meter Position Service }_{nombre}
Ejemplos	
http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_Engineering_Building http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_Engineering_Building_Ground_Floor http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_AHU101_Component_Cooling_Coil http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_AHU101_Component_Extract_Air_Fan http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_G047_Lecture_Theater_3_Room_147_AHU101_Ctrls http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_G047_Lecture_Theater_3_Engineering_Building	

Tabla 36. Instanciación de elementos de la ontología (Parte 1)

Se procede por tanto a instanciar todas las partes de la AHU101 y el edificio mediante las clases reutilizadas, entrando ya esta parte en la fase de implementación. Todos los elementos de la ontología se pueden consultar en el [Apéndice C](#) o en su versión navegable en HTML en la documentación en el gitlab del proyecto, consultar [Apéndice A](#). Como resultado final del proceso de modelado del sistema para la AHU101, la ontología final obtenida contiene los siguientes elementos:

- Classes (clases): 42
- Object Properties (relaciones): 14
- Data Properties (atributos): 2
- Individuals (instancias): 105

En cuanto a la instanciación de cada medida resultado del mapeo desarrollado en el Apartado 8.3, en la Ilustración 40 del Apartado 3.3 se refleja la estructura de SAREF para cada instancia de **“Measurement”**. Las instancias quedan definidas siguiendo el patrón de nombrado siguiente de la Tabla 37, donde también se muestran algunos ejemplos. Se puede ampliar la información sobre las mediciones en el Capítulo 9.

URI BASE:	https://www.uva.es/gii/tfg/angnava/ontology
PATRÓN	
http://{URI base}# Measure_{“id”}_{nombre}_{“dateTime”}	
EJEMPLOS	
HTTPS://SAREF.ETSI.ORG/CORE/MEASURE_1048511_AHU101_HOURS_RUN_2018_12_04T04_00_00_000_01_00 HTTPS://SAREF.ETSI.ORG/CORE/MEASURE_1048513_MCC02__FIRE_ALARM_2018_12_04T04_02_00_000_01_00 HTTPS://SAREF.ETSI.ORG/CORE/MEASURE_1048516_AHU101_CALC_SUPPLY_SETPT_2018_12_04T04_05_00_000_01_00 HTTPS://SAREF.ETSI.ORG/CORE/MEASURE_1048516_AHU101_SAF_VSD_SPEED_2018_12_04T04_05_00_000_01_00 HTTPS://SAREF.ETSI.ORG/CORE/MEASURE_1048574_TCV_101_1_MAIN_HVALVE_2018_12_04T05_03_00_000_01_00 HTTPS://SAREF.ETSI.ORG/CORE/MEASURE_1048574_LECTURE_THEATRE_3_AVG_CO2_2018_12_04T05_03_00_000_01_00	

Tabla 37. Instanciación de elementos de la ontología (Parte 2)

A la hora de generar el documento RDF con el mapeado de las medidas, la clase **“Measurement”** se añade a las 42 ya utilizadas. Además, por cada fila de datos, estamos instanciando las medidas de 29 dispositivos y cada instancia generará:

- 2 Object Property: **“isMeasuredIn”** y **“relatesToProperty”**
- 2 Data Property: **“hasValue”** y **“hasTimeStamp”**

La instanciación de estas medidas se hace en automático por el programa. Se puede ampliar la información sobre las mismas en el Capítulo 9.

8.4.3. Diagrama general de la ontología

La Ilustración 71 muestra un detalle sobre el diagrama general de la ontología. Los círculos contienen el nombre del elemento y su número de instancias. Los distintos colores diferencian el prefijo utilizado para los elementos, siendo el azul claro el de los elementos de la ontología, el morado SAREF4BLDG, azul oscuro SAREF y por último rojo para GEO.

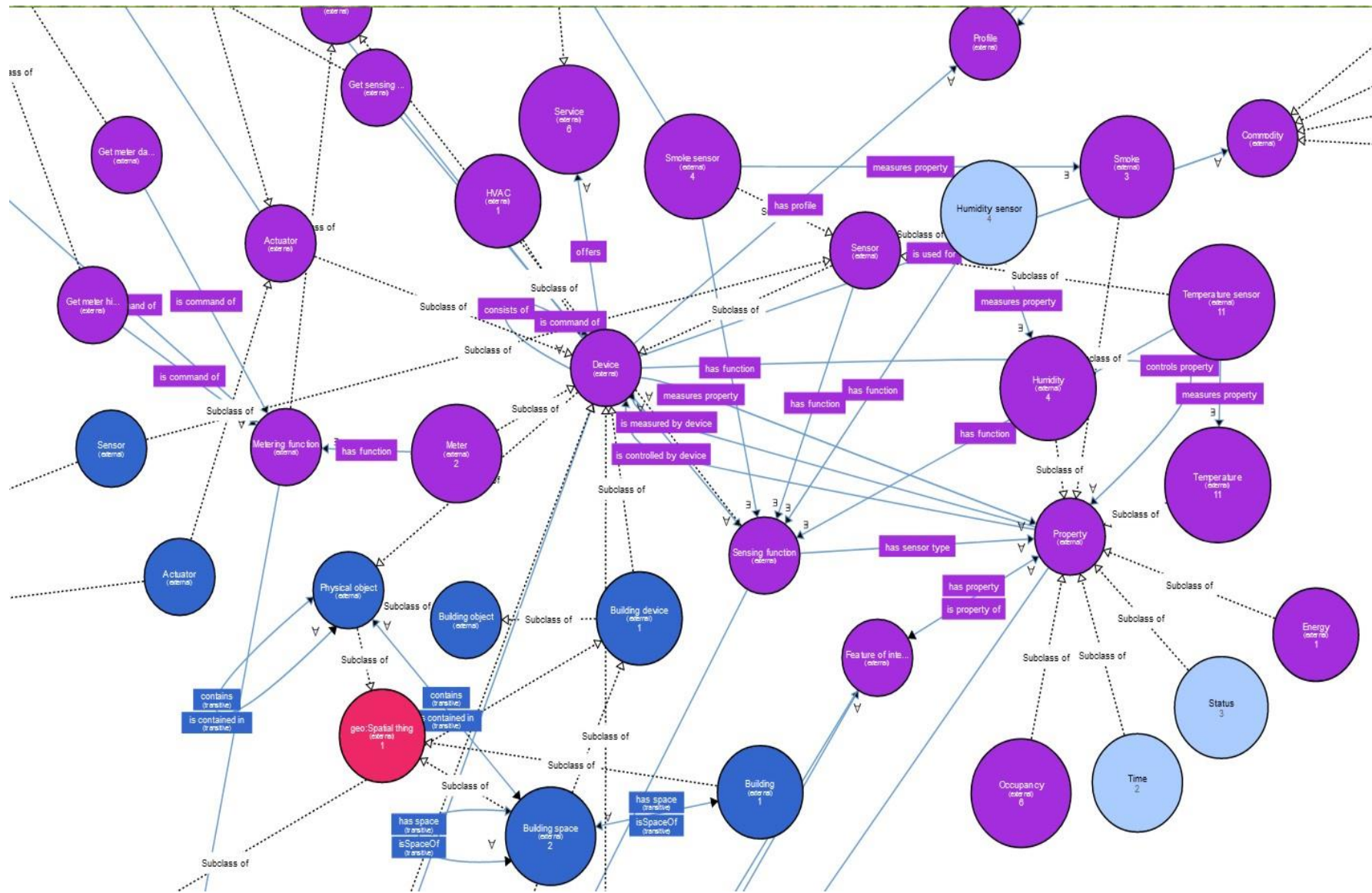


Ilustración 71. Detalle del diagrama general para la ontología, elaborado con WebVOWL

8.5. Implementación

Cada una de las tecnologías utilizadas tienen desarrollo en los sistemas operativos más comunes, se ha optado por Microsoft Windows 10 de 64 bit al ser uno de los más distribuidos, y, por tanto, las descripciones están orientadas a la instalación sobre dicha plataforma. Las indicaciones para la instalación del entorno de desarrollo de la ontología, de programación y de pruebas se describe en el [Apéndice D](#).

8.5.1. Diagrama de actividades

Previo a la implementación de las actividades que se crearán para dar resolución a los objetivos del proyecto, se muestra en el diagrama UML³⁵ de la Ilustración 72 una visión general de dichas actividades y como se coordinan entre sí.

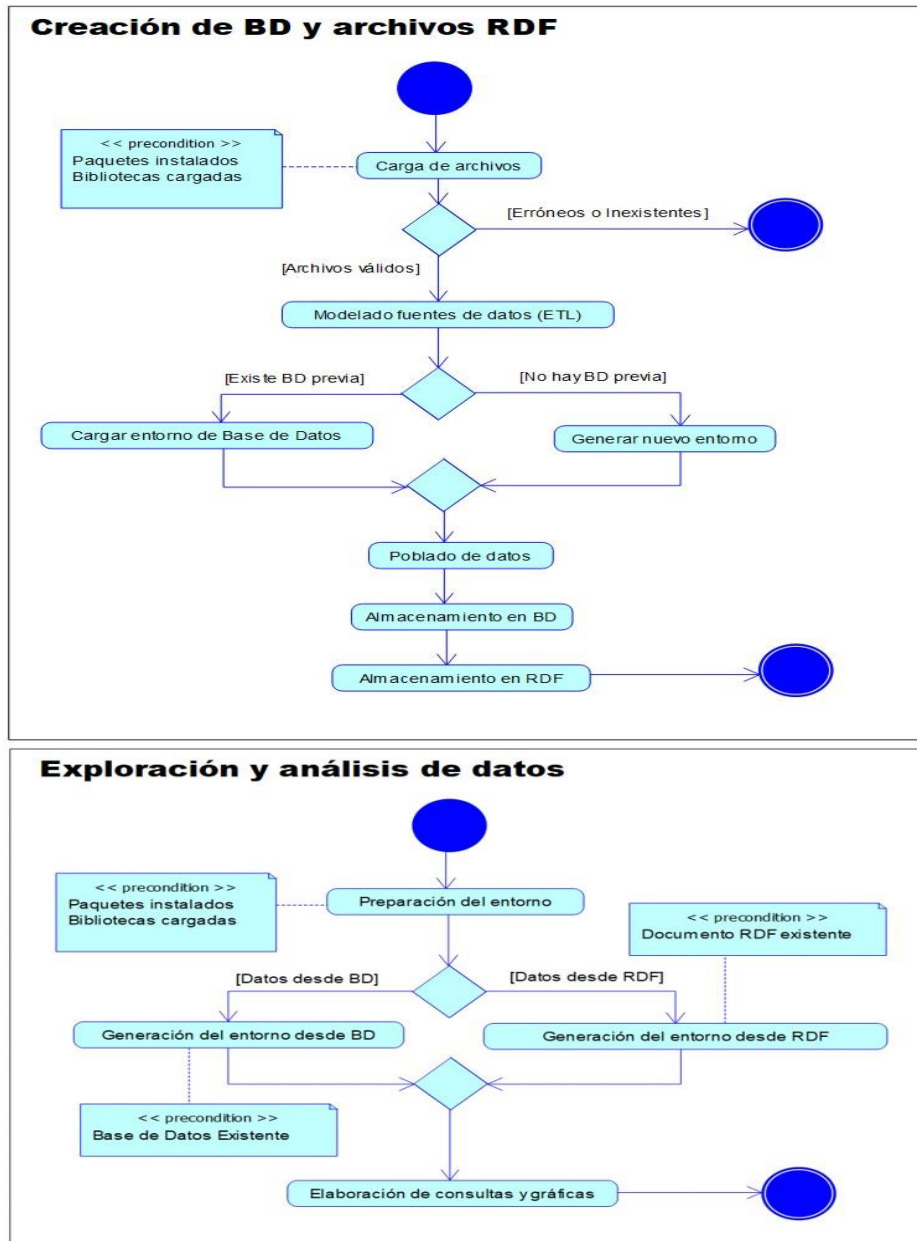


Ilustración 72. Diagrama de actividades

³⁵ UML: Lenguaje Unificado de Modelado (*Unified Modeling Language*), es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema https://en.wikipedia.org/wiki/Unified_Modeling_Language

8.5.2. Implementación del tratamiento de los NOR

Con el servidor Jupyter Notebook abierto en nuestro navegador, accedemos al directorio “TFG” y al notebook generado que contine el proyecto “**proyecto_angnava.ipynb**” y pulsamos sobre el mismo para abrirlo. Una vez abierto en el navegador, nos situaremos en el último bloque para comenzar a implementar el tratamiento de los datos.

Lectura de las fuentes de datos

Iniciaremos el código con la lectura de los archivos CSV de las fuentes en un script amigable de widget que emula el administrador de archivos de Windows y permite múltiple selección mediante un botón “**botón**” y la clase Python de gestión “**SelectFilesButton**”. El código se muestra a continuación:

```

### Lectura de datos crudos desde csv ###
class SelectFilesButton(widgets.Button):
    def __init__(self):
        super(SelectFilesButton, self).__init__()
        self.layout=Layout(width='initial')
        self.description = "Seleccionar Archivos"
        self.icon = "square-o"
        self.style.button_color = "lightgreen"
        self.style.button_color = "lightgreen"
        self.on_click(self.select_files)
    @staticmethod
    def select_files(b):
        root = Tk()
        root.withdraw()
        root.call('wm', 'attributes', '.', '-topmost', True)
        b.files = filedialog.askopenfilename(multiple=True, filetypes = (('csv files','*.csv'),))
        if b.files:
            b.description = "Archivos Seleccionados"
            b.icon = "check-square-o"
            b.style.button_color = "lightblue"
            print(f"Ficheros leídos: {len(b.files)}", *b.files, sep="\n")
        else:
            b.description = "Seleccionar Archivos"
            b.icon = "square-o"
            b.style.button_color = "orange"

boton = SelectFilesButton()
display(boton)

```

Proceso ETL

Se procede a la creación y ejecución de un *clúster* en local de Spark , como ya se ha indicado en varias partes del proyecto, la inclusión de Spark permitirá futuros desarrollos, tanto distribuidos, como multifuente. Con el *clúster* en funcionamiento, almacenamos los datos ya cargados y comenzamos la etapa ELT de

tratamiento comentada en el Apartado 8.3. El algoritmo mostrará por pantalla algunos informes sobre el número de datos leídos y los descartes.

```
#Carga de Spark
appName = "Ejemplo_01"
master = 'local[*]'
spark = SparkSession.builder.master(master).appName(appName).getOrCreate()

with open("schema.json") as f:
    custom_schema = StructType.fromJson(json.load(f))

#Carga de archivo y limpieza de datos
df0 = spark.read.csv( list(boton.files),
                    header=True,
                    schema=custom_schema,
                    mode="DROPMALFORMED",
                    timestampFormat="yyyy-MM-dd HH:mm:ss" )

print("Registros leídos: "+str(df0.count()))
#Eliminación de duplicados generales
df1 = df0.drop_duplicates()
print("Registros diferentes: "+str(df1.count()))

# Eliminación de duplicados por columnas clave
df2 = df1.drop_duplicates(["id","ReadingTime"])
print("Registros diferentes por 'id' y 'ReadingTime': "+str(df2.count()))
df3 = df2.dropna()
print("Registros con valores (columnas) faltantes: "+str(df2.count()-df3.count()))
# Tratamiento de las variables logicas
df = df3.withColumn("HS_101_1_SAF_Enable_DATALOG15",
                  df0["HS_101_1_SAF_Enable_DATALOG15"].cast(IntegerType()))\
        .withColumn("MCC02__Fire_Alarm_DATALOG29",
                  df0["MCC02__Fire_Alarm_DATALOG29"].cast(IntegerType()))\
        .withColumn("HS_101_2_EAF_Enable_DATALOG16",
                  df0["HS_101_2_EAF_Enable_DATALOG16"].cast(IntegerType()))

# Ordenamiento de datos
df=df.orderBy(df.ReadingTime.asc())
# Transformacion a json con diccionario
df_list_of_jsons = df.toJSON().collect()
df_list_of_dicts = [json.loads(x) for x in df_list_of_jsons]
df_json = json.dumps(df_list_of_dicts)
#paramos contexto spark
spark.stop()
```

Algoritmo ad-hoc y población de los datos




Por último, se procede al recorrido de los datos y la inclusión de las medidas en la ontología mediante el motor de procesamiento de la biblioteca Owlready2, implementando el algoritmo descrito en el Apartado 8.3. En este punto se detectó un pequeño error en la biblioteca Owlready2 a la hora de tratar la estructura “*Timestamp*” para las marcas temporales que obliga a asignar a la propiedad “*hasTime*” de diferente

manera si el dato carga desde la base de datos SQLite3 o si es de nueva creación. Para subsanar este inconveniente se creó una variable booleana “*newsystem*” que permite discriminar entre los dos tipos de asignaciones dependiendo de si la fuente de datos de entrada es de nueva creación o cargada desde la base de datos.

```
# Recorremos la lista de valores, por cada dato/fila procesamos en la ontología
for x in df_list_of_dicts:
    for key,value in x.items():
        #eliminamos en nombre cabecera de archivos DATALOG
        head, sep, tail = key.partition('_DATALOG')
        strname = "Measure_"+str(x['id'])+"_"+head+"_"+re.sub('[-:~+]', '_',x['ReadingTime'])
        #no procesamos estas entradas no son componentes
        if head=="id" or head=="ReadingTime": continue
        #no procesamos esta entrada al no tener uso
        elif head=="GF_East_Rads_Energy_Acc_Total": continue
        elif len(world.search(iri =strname))==0: #si no existe la medida
            measure = core.Measurement(strname) #creamos Measurement
            #aplicamos Annotations y Data Property
            measure.label = "Measure "+re.sub('_', ' ',head)+' '+x['ReadingTime']
            measure.HasValue.append(value)
            dt=datetime.strptime(x['ReadingTime'], "%Y-%m-%dT%H:%M:%S.%f%z")
            if newsystem: measure.hasTimestamp.append(dt)
            else: measure.hasTimestamp=dt
            #propiedad a la que pertenece
            propiedad = world.search(iri = "*NUIG_AHU101_Ctrls_"+head+"*")[0]
            measure.relatesToProperty.append(propiedad)
            if propiedad.is_a[0].label[0]=='Temperature':
                measure.isMeasuredIn.append( world.search(iri = "*CelsiusDegrees" )[0] )
            elif propiedad.is_a[0].label[0]=='Humidity':
                measure.isMeasuredIn.append(world.search(iri = "*PercentageRH" )[0] )
            elif propiedad.is_a[0].label[0]=='Occupancy':
                measure.isMeasuredIn.append( world.search(iri = "*Percentage" )[0] )
            elif propiedad.is_a[0].label[0]=='Smoke':
                measure.isMeasuredIn.append( world.search(iri = "*PartsPerMillion" )[0] )
            elif propiedad.is_a[0].label[0]=='Status':
                measure.isMeasuredIn.append(world.search(iri = "*Boolean" )[0] )
            else: #tipo Time
                measure.isMeasuredIn.append( world.search(iri = "*Hour" )[0] )
```

8.5.3. Implementación de la ontología

Con el *framework* Protégé vamos a proceder a crear todos los elementos del modelo de la ontología. Lo primero que hay que configurar es el espacio de URIs sobre el que se identificarán los elementos, las ontologías relacionadas y las importaciones. Todas estas declaraciones formarán parte de la cabecera del documento (ver Apartado 2.5.4.2 sobre la estructura del *header* en OWL2). También son importantes las anotaciones generales sobre la ontología como son: creador, descripción, versión, licencia, etc. Para proceder, una vez abierto Protégé nos situaremos en la primera pestaña superior “*Active ontology*” y

rellenamos los campos marcados para “**Ontology IRI**” y “**Ontology Version IRI**”. En el recuadro central de “**Annotations**” añadimos las etiquetas deseadas mediante el desplegable  con sus correspondientes datos. Finalmente, en la parte inferior, tenemos que añadir las importaciones de SAREF y SAREF4BLDG en la pestaña “**Ontology imports**”, mediante el desplegable  y en la pestaña de “**Ontology Prefixes**” mediante el botón  definir los prefijos y sus URI relacionadas. Estos procesos los podemos ver en la Ilustración 73 y la Ilustración 74.

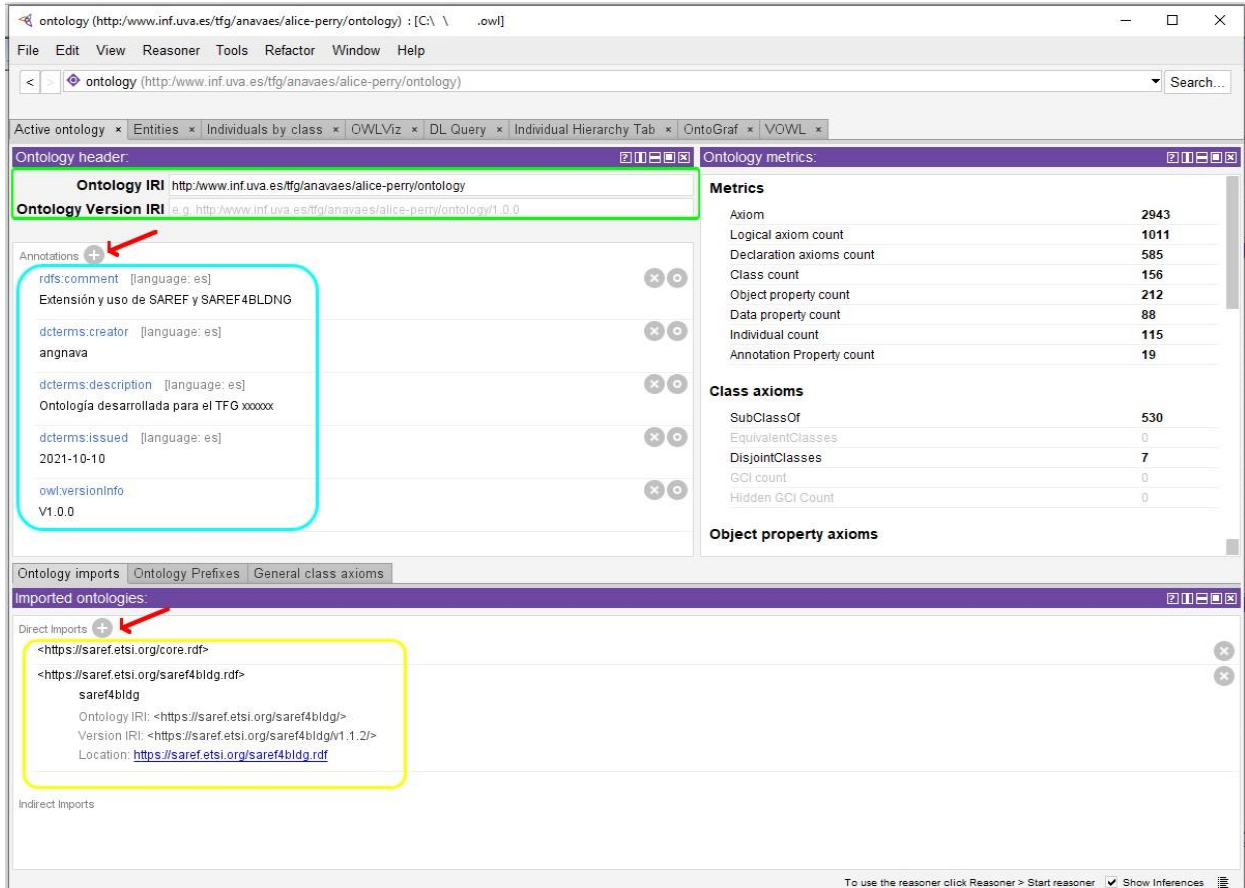


Ilustración 73. Protégé: creación de URIs, anotaciones generales e importaciones

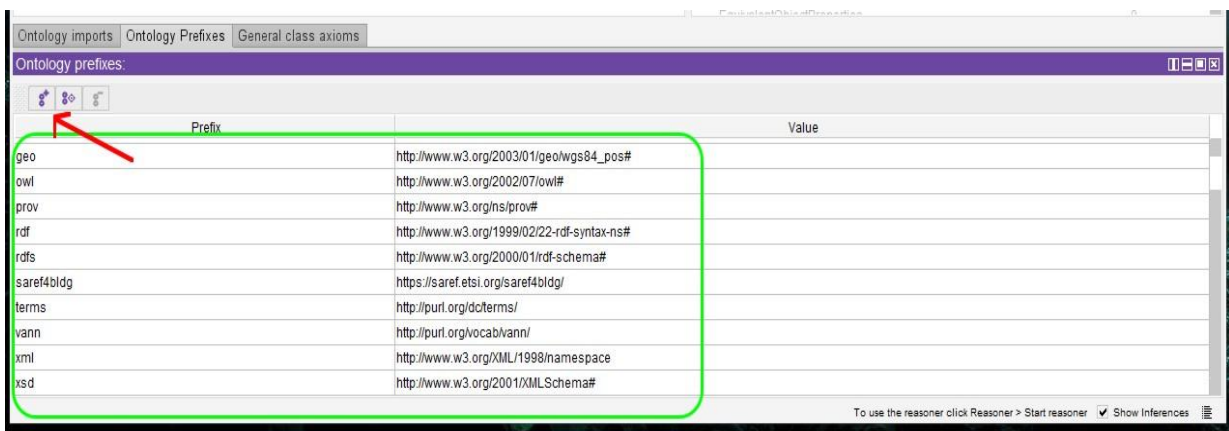


Ilustración 74. Protégé: creación de prefijos

Una vez importadas las ontologías SAREF y SAREF4BLDG, podemos observar su taxonomía de clases en la pestaña superior “**Entities**” subpestaña “**Classes**”: Ilustración 75. Igualmente encontraremos las

relaciones en la subpestaña “**Object properties**” y las propiedades en “**Data properties**”. Para crear las instancias o individuos también encontramos aquí la subpestaña “**Individuals**”.

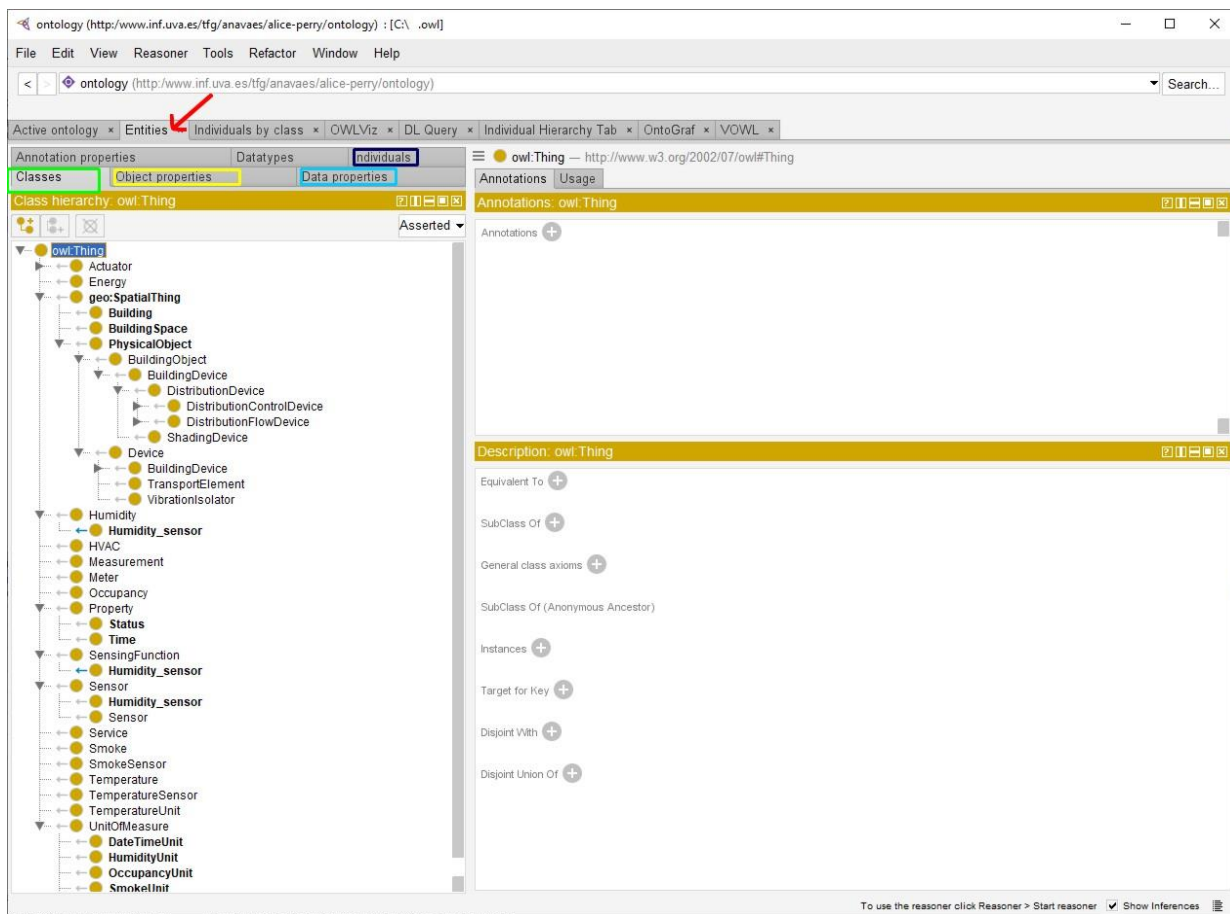


Ilustración 75. Protégé: pestaña “**Entities**” y árbol de clases

Sobre la subpestaña “**Individuals**”, pulsaremos sobre el icono superior para crear una instancia. Se abrirá una ventana para introducir su nombre, si incluimos espacios se convertirán en guion bajo en automático. Automáticamente nos la asociará a la IRI base de nuestra ontología en la ventana inferior, esto se puede modificar más adelante si fuera necesario, Ilustración 76.

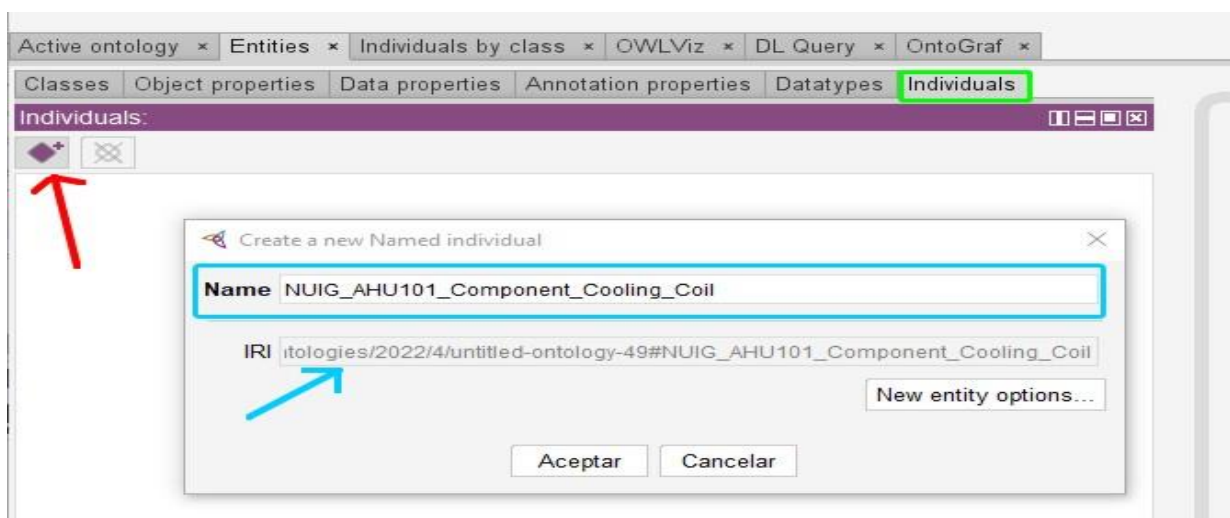


Ilustración 76. Protégé: instanciar un individuo (Parte 1)

Una vez creada la instancia, nos carga a la izquierda del recuadro central. Ahora a la derecha del recuadro, hay que asociarla al tipo de clase que pertenece, sus relaciones y propiedades. Pulsando sobre los correspondientes botones de añadir **+** aparecerán desplegables para seleccionar *classes*, *object properties*, *data properties* o *annotations*. Podemos ver un ejemplo con algunos ya instanciados y la ventana para añadir una *data property* desplegada en la Ilustración 77.

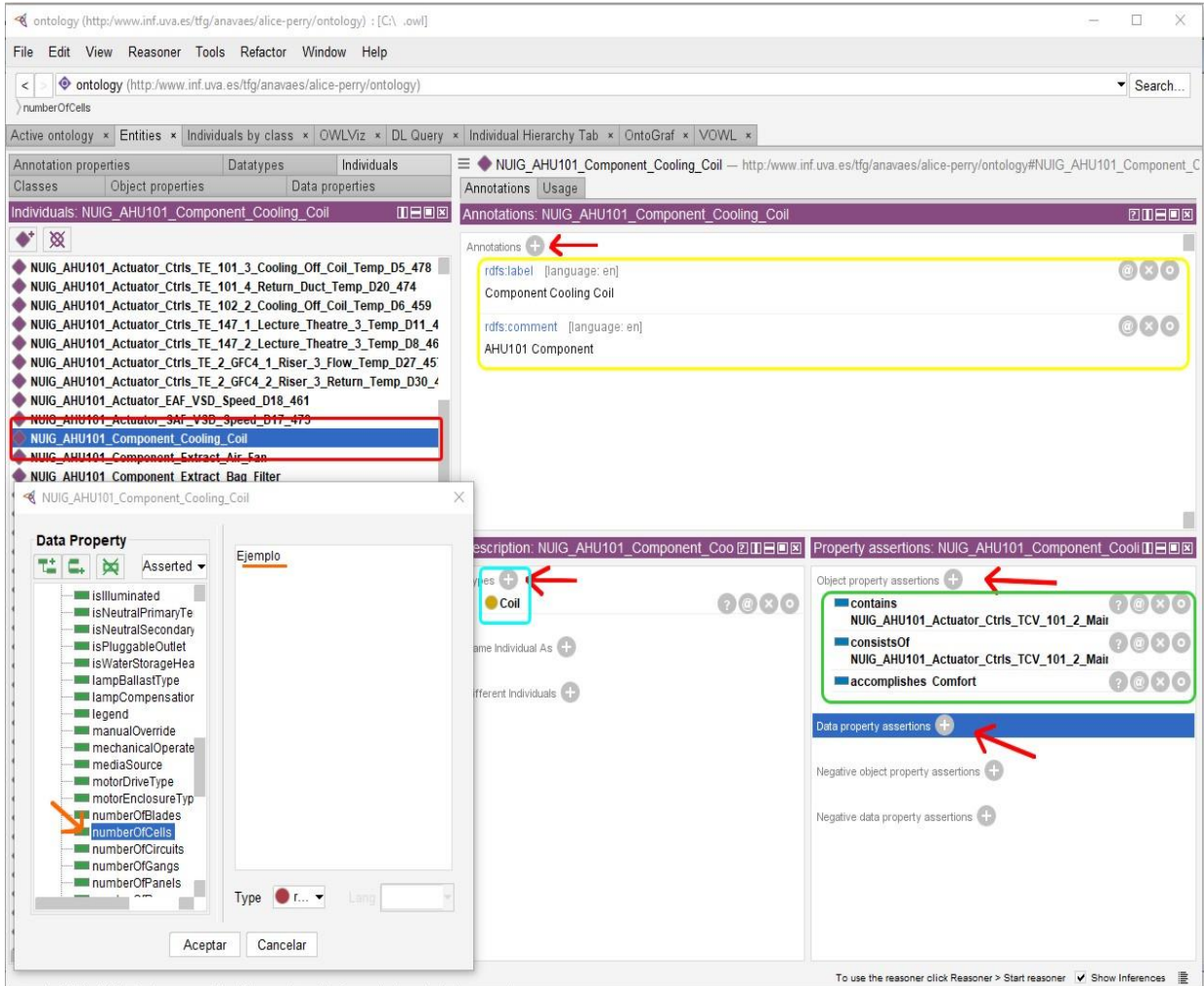


Ilustración 77. Protégé: instanciar un individuo (Parte 2)

Una vez implementados todos los elementos, guardamos la ontología en formato OWL-RDF desde el menú superior en **File** luego **Save as...**, en el desplegable dejamos **RDF/XML Syntax** damos a aceptar y en la ventana siguiente nos dejará elegir el nombre del archivo y el directorio de grabación. Dejamos el mismo directorio del proyecto, y de nombre **ontologia.rdf**.

8.5.4. Implementación de los datos RDF y consultas SPARQL

La biblioteca utilizada Owlready2, Apartado 7.10, además de permitir salvar los datos en una base de datos SQLite3 mantiene en memoria la estructura del grafo de la ontología, lo que permite realizar, tanto un volcado en fichero con formato OWL-RDF como consultas SPARQL. En el momento de carga inicial de la ontología solamente se tiene en memoria las instancias del modelado del archivo **ontologia.rdf**, cuando se ejecuta el código para el tratamiento de los NOR, Apartado 8.5.2, se van acumulando las medidas al grafo en memoria y a los volcados sobre la base de datos. Por tanto, si queremos forzar un volcado en fichero OWL-RDF tenemos la instrucción:

```
world.save(file = "datos.rdf", format = "rdxml")
```

Y para cualquier consulta SPARQL podemos crearla como cadena de texto y lanzarla sobre el motor que proporciona la biblioteca con el comando "list". En el ejemplo siguiente devuelve el máximo de una medición y realiza una gráfica:

```
#Consulta
result = list (world.sparql("""

PREFIX : <http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#>
PREFIX core: <https://saref.etsi.org/core/>
PREFIX s4bldg: <https://saref.etsi.org/saref4bldg/>

SELECT ?t (MAX(?v) as ?mx) (AVG(?v) as ?av)
WHERE {
  ?s core:relatesToProperty :NUIG_AHU101_Ctrls_CO_147_1_Lecture_Theatre_3_CO2_D10_462 .
  ?s core:hasValue ?v .
  ?s core:hasTimestamp ?t .
}
ORDER BY ?t

"""))

display(result);

#Gráfica
plt.plot(result[0][0].strftime("%d/%m/%Y, %H:%M:%S"),result[0][1],'bP',label="Maximo")
plt.show()
```

En la Tabla 39 y la Ilustración 83 del Apartado 9.3. se pueden observar otras cadenas de consulta SPARQL y la gráfica generada por una de ellas.

8.6. Validación

Para la validación de la ontología se ha empleado la herramienta online [OOPS!](#) Descrita en el Apartado 7.4. Es una aplicación web independiente de cualquier entorno de desarrollo de ontologías, con la que se pueden diagnosticar posibles "*pitfall*" lo que se puede traducir como escollos o trampas en el diseño del modelado y obtener recomendaciones sobre qué acciones tomar para resolverlos. Básicamente es una herramienta que valida ontologías sobre una serie de patrones de diseño y recomendaciones, y ofrece unas recomendaciones. Las recomendaciones ofrecidas hay que revisarlas detenidamente ya que pueden carecer de validez o no ser importantes para la ontología validada. En la Ilustración 78 se observan los resultados obtenidos:

Evaluation results

It is obvious that not all the pitfalls are equally important; their impact in the ontology will depend on multiple factors. For this reason, each pitfall has an importance level attached indicating how important it is. We have identified three levels:

- **Critical** 🚨 : It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- **Important** 🟡 : Though not critical for ontology function, it is important to correct this type of pitfall.
- **Minor** 🟢 : It is not really a problem, but by correcting it we will make the ontology nicer.

[Expand All] | [Collapse All]

Results for P04: Creating unconnected ontology elements.	1 case Minor 🟢
Results for P08: Missing annotations.	2 cases Minor 🟢
Results for P11: Missing domain or range in properties.	273 cases Important 🟡
Results for P13: Inverse relationships not explicitly declared.	195 cases Minor 🟢
Results for P22: Using different naming conventions in the ontology.	ontology* Minor 🟢
Results for P24: Using recursive definitions.	4 cases Important 🟡
Results for P30: Equivalent classes not explicitly declared.	1 case Important 🟡
Results for P32: Several classes with the same label.	3 cases Minor 🟢
Results for P41: No license declared.	ontology* Important 🟡

According to the highest importance level of pitfall found in your ontology the conformace badge suggested is "Important pitfalls" (see below). You can use the following HTML code to insert the badge within your ontology documentation:

Want to help?

- Suggest new pitfalls
- Provide feedback

Documentation:

- Pitfall catalogue
- User guide
- Technical report

Related papers:

- IJISWIS 2014
- EKAW 2012
- ESWC 2012 Demo
- Ontoqual 2010
- CAEPIA 2009

Web services:

- REST Web Service

Developed by:

Ilustración 78. Resultado de la evaluación de la ontología en OOPS!

Se procederá, por tanto, a revisar cada uno de los escollos reportados y a la revisión de las recomendaciones ofrecidas para verificar su aplicación en la ontología. La Tabla 38 presenta en su primera columna el nivel del escollo, con valores entre: crítico, importante o menor. En la segunda columna, el tipo de escollo que facilita OOPS!, en la tercera se muestra el número de casos encontrados por cada tipo de escollo. La última columna muestra la descripción del escollo y la solución aplicada para resolverlo, si se considera necesaria dentro del marco de desarrollo del proyecto.

Nivel	Tipo de Escollo	Núm. Casos	Descripción y Solución
Important	P11	273	<p>Descripción:</p> <div data-bbox="547 309 1425 658" style="border: 1px solid #ccc; padding: 5px;"> <p>Results for P11: Missing domain or range in properties. 273 cases Important</p> <p>Object and/or datatype properties without domain or range (or none of them) are included in the ontology.</p> <ul style="list-style-type: none"> • This pitfall appears in the following elements: > https://saref.etsi.org/core/isUsedFor > https://saref.etsi.org/core/isAbout > https://saref.etsi.org/core/hasTypicalConsumption > https://saref.etsi.org/core/isAccomplishedBy > https://saref.etsi.org/core/consistsOf > https://saref.etsi.org/core/hasPrice > https://saref.etsi.org/core/accomplishes </div> <p>Solución: nos está detectando la falta de definición del dominio y rango para elementos no instanciados o instanciados parcialmente, de las ontologías SAREF y SAREF4BLDG. Se resolvería con la eliminación de estos elementos de la solución final propuesta en el proyecto dejando solamente aquellos que forman parte del mapeado de medidas. Se toma la decisión de no hacer nada al respecto ya que se prioriza el requisito de permitir desarrollos futuros de la ontología que los necesitarán. Mientras que no se realicen inferencias complejas que utilicen estos elementos no definidos, no supondrá ninguna pérdida de funcionalidad, además, las inferencias están fuera del ámbito del TFG.</p>
Important	P24	4	<p>Descripción:</p> <div data-bbox="547 1084 1425 1500" style="border: 1px solid #ccc; padding: 5px;"> <p>Results for P24: Using recursive definitions. 4 cases Important</p> <p>An ontology element (a class, an object property or a datatype property) is used in its own definition. Some examples of this would be: (a) the definition of a class as the enumeration of several classes including itself; (b) the appearance of a class within its owl:equivalentClass or rdfs:subClassOf axioms; (c) the appearance of an object property in its rdfs:domain or range rdfs:range definitions; or (d) the appearance of a datatype property in its rdfs:domain definition.</p> <ul style="list-style-type: none"> • This pitfall appears in the following elements: > https://saref.etsi.org/core/Profile > https://saref.etsi.org/saref4bldg/BuildingSpace > https://saref.etsi.org/saref4bldg/PhysicalObject > https://saref.etsi.org/core/Device </div> <p>Solución: el escollo descrito detecta la autoinclusión de un elemento en su propia definición, un inconveniente habitual en las relaciones que describen subclases al contener recursividad y ser potencialmente problemática su finalización. Revisando los elementos resaltados, son las propias definiciones de SAREF y SAREF4BLDG las que llevan a esta situación: un dispositivo (<i>Device</i>) puede contener otros dispositivos, un dispositivo tiene un perfil (<i>hasProfile</i>) pero al contener otros dispositivos estos a su vez contienen otros perfiles...Lo mismo con edificios y objetos en edificios (<i>BulidingSpace</i>, <i>PhysicalObject</i>). Se trata de definiciones propias de los estándares base ya fundamentadas en los mismos, por lo que no procede ninguna acción. Además, como ocurre con el apartado anterior, no influyen para el ámbito de la ontología y los resultados a analizar, por lo que no requieren mayor atención.</p>

Important	P30	1	<p>Descripción:</p> <div data-bbox="544 232 1426 506" style="border: 1px solid #ccc; padding: 5px;"> <p>Results for P30: Equivalent classes not explicitly declared. 1 case Important</p> <p>This pitfall consists in missing the definition of equivalent classes (<code>owl:equivalentClass</code>) in case of duplicated concepts. When an ontology reuses terms from other ontologies, classes that have the same meaning should be defined as equivalent in order to benefit the interoperability between both ontologies.</p> <ul style="list-style-type: none"> • The following classes might be equivalent: › https://saref.etsi.org/core/Time, https://saref.etsi.org/core/Meter, http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#Time </div> <p>Solución: está detectando clases equivalentes. Efectivamente como se generan los datos en un RDF externo con la URI base de la ontología, hace falta vincular dichas implementaciones con las clases de SAREF y SAREF4BLDG del archivo con las definiciones del modelo. Se incluyen por tanto definiciones de <code>owl:equivalentClass</code> en el RDF que genera la aplicación tal y como recomienda la herramienta.</p>
Important	P41	n/a	<p>Descripción:</p> <div data-bbox="544 790 1426 947" style="border: 1px solid #ccc; padding: 5px;"> <p>Results for P41: No license declared. ontology* Important</p> <p>The ontology metadata omits information about the license that applies to the ontology.</p> <p>*This pitfall applies to the ontology in general instead of specific elements.</p> </div> <p>Solución: es una advertencia sobre la omisión de la etiqueta sobre licencia de uso. Esta ontología no va a ser publicada y su uso se restringe al TFG, por lo que no se aplicará ninguna acción al respecto.</p>
Minor	P04	1	<p>Descripción:</p> <div data-bbox="544 1131 1426 1310" style="border: 1px solid #ccc; padding: 5px;"> <p>Results for P04: Creating unconnected ontology elements. 1 case Minor</p> <p>Ontology elements (classes, object properties and datatype properties) are created isolated, with no relation to the rest of the ontology.</p> <ul style="list-style-type: none"> • This pitfall appears in the following elements: › http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing </div> <p>Solución: efectivamente, esta clase y su instanciación, que define las coordenadas espaciales del edificio, no se enlazan con ningún elemento más de la ontología, ya que los requisitos se centran en las medidas y no en las posiciones físicas de los elementos. Es un dato que se deja como muestra de instanciación para futuros desarrollos que sí incluyan localizaciones.</p>
Minor	P08	2	<p>Descripción:</p> <div data-bbox="544 1597 1426 1848" style="border: 1px solid #ccc; padding: 5px;"> <p>Results for P08: Missing annotations. 2 cases Minor</p> <p>This pitfall consists in creating an ontology element and failing to provide human readable annotations attached to it. Consequently, ontology elements lack annotation properties that label them (e.g. <code>rdfs:label</code>, <code>lemon:LexicalEntry</code>, <code>skos:prefLabel</code> or <code>skos:altLabel</code>) or that define them (e.g. <code>rdfs:comment</code> or <code>dc:description</code>). This pitfall is related to the guidelines provided in [5].</p> <ul style="list-style-type: none"> • The following elements have no <code>rdfs:label</code> defined: › http://www.w3.org/2003/01/geo/wgs84_pos#longitude › http://www.w3.org/2003/01/geo/wgs84_pos#latitude </div> <p>Solución: la herramienta nos insta a poner una etiqueta con un “nombre entendible en lenguaje humano” para saber que dato se guarda. En este caso, aunque el nombre es suficientemente descriptivo y el dato no se va a utilizar, se añaden las anotaciones <i>label</i> con “<i>longitud</i>” y “<i>latitud</i>” a las <i>Data properties</i> de la ontología.</p>

Minor	P13	195	<p>Descripción:</p> <p>Results for P13: Inverse relationships not explicitly declared. 195 cases Minor</p> <p>This pitfall appears when any relationship (except for those that are defined as symmetric properties using <code>owl:SymmetricProperty</code>) does not have an inverse relationship (<code>owl:inverseOf</code>) defined within the ontology.</p> <ul style="list-style-type: none"> • OOPS! has the following suggestions for the relationships without inverse: <ul style="list-style-type: none"> > https://saref.etsi.org/core/isControlledByDevice could be inverse of https://saref.etsi.org/core/measuresProperty > https://saref.etsi.org/core/isMeasuredByDevice could be inverse of https://saref.etsi.org/core/measuresProperty • Sorry, OOPS! has no suggestions for the following relationships without inverse: <ul style="list-style-type: none"> > https://saref.etsi.org/saref4bldg/flowResistanceMin > https://saref.etsi.org/saref4bldg/horizontalSpacing > https://saref.etsi.org/saref4bldg/compressorSpeed <p>Solución: se nos facilita una lista de relaciones que pueden ser inversas entre sí. En el primer bloque tenemos el caso similar a la P24, esas relaciones vienen establecidas por el estándar, y no se van a modificar. En el segundo bloque se repite circunstancia, añadiendo las indicaciones del P11: son relaciones sin definir que no afectan al proyecto y se mantienen para desarrollos futuros.</p>
Minor	P22	n/a	<p>Descripción:</p> <p>Results for P22: Using different naming conventions in the ontology. ontology* Minor</p> <p>The ontology elements are not named following the same convention (for example CamelCase or use of delimiters as "-" or "_"). Some notions about naming conventions are provided in [2].</p> <p>*This pitfall applies to the ontology in general instead of specific elements.</p> <p>Solución: existen diferencias en el nombrado de los elementos, mezclando el estándar CamelCase con los guiones. La ontología es producto de la fusión de dos estándares que importan a su vez dos ontologías, es normal que esto suceda, no se toma ninguna acción ya que requiere algo imposible: cambiar los estándares. Tampoco afecta a la ontología en ningún caso.</p>
Minor	P32	3	<p>Descripción:</p> <p>Results for P32: Several classes with the same label. 3 cases Minor</p> <p>Two or more classes have the same content for natural language annotations for naming, for example the <code>rdfs:label</code> annotation. This pitfall might involve lack of accuracy when defining terms.</p> <ul style="list-style-type: none"> • The following classes contains the same label, maybe they should be replaced by one class with several labels or might be equivalent classes: <ul style="list-style-type: none"> > https://saref.etsi.org/core/Time, http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#Time > https://saref.etsi.org/saref4bldg/Sensor, https://saref.etsi.org/core/Sensor > https://saref.etsi.org/core/Actuator, https://saref.etsi.org/saref4bldg/Actuator <p>Solución: Nos está detectando que varias clases tienen la misma etiqueta de nombre, por lo que una lectura “humana” de la ontología puede llevar a confundir elementos si solamente se hace por <i>label</i> (nombre). En el caso de <i>#Time</i>, se resolverá con la solución del P30. Con respecto a <i>Sensor</i> y <i>Actuator</i>, en el modelo de la AHU101 se les trata por igual al nombrarlos, se puede incluir una marca que los diferencia en el nombre, pero se decide seguir el nombrado de la documentación y no tocarlo.</p>

Tabla 38. Descripción y solución a escollos localizados en OOPS!

Al ser la ontología una adaptación reducida del dominio de SAREF y SAREF4BLDG, junto con los plugin incluidos para la ayuda en la sintaxis que proporciona Protégé con herramientas de desarrollo, no se ha encontrado ningún escollo que no tuviera suficiente justificación o que obligara a grandes cambios en la ontología.

Capítulo 9

9. Exploración y análisis de los datos RDF

Este capítulo amplía la información descrita en el Apartado 8.3 describiendo el mapeado de las medidas de los sensores y comprobando su funcionalidad mediante la realización de varias consultas SPARQL complementadas con un entorno gráfico.

9.1. Mapeado de las medidas en JSON

Inicialmente se parte de un archivo CSV con la primera fila como cabecera y el resto con datos. Tomaremos de ejemplo la línea de datos, fila 2, de la Ilustración 79. Se marcan los siguientes datos sobre el archivo y sus valores:

- **Id:** identificador, en azul oscuro
- **ReadingTime:** hora de lectura de los valores, en azul claro
- **HE_147_1_Lecture_Theatre_3_Humidity_DATALOG12:** humedad relativa de la sala G47 posición 1, en amarillo
- **TE_2_GFC4_1_Riser_3_Flow_Temp_DATALOG27:** temperatura del flujo de agua del calentador GFC4 en la AHU, en verde

	A	B	C	D	E	F	G	H	I	J	K	L
1	id,ReadingTime,HS_101_1_SAF_Enable_DATALOG15,HE_147_1_Lecture_Theatre_3_Humidity_DATALOG12,TE_2_GFC4_1_Riser_3_Flow_Temp_DATALOG27,HI											
2	1048511,2018-12-04 04:00:00,0.0,35.57564926147461,22.677305221557617,39.36426544189453,19.554208755493164,25.399761199951172,0.0,512.8112182											
3	1048512,2018-12-04 04:01:00,0.0,35.59758377075195,22.66918182373047,39.219970703125,19.55238914489746,25.399824142456055,0.0,512.74926757812											
4	1048513,2018-12-04 04:02:00,0.0,35.519203186035156,22.65974998474121,39.213287353515625,19.54937171936035,25.40226173400879,0.0,512.29760742											
5	1048514,2018-12-04 04:03:00,0.0,35.511714935302734,22.651147842407227,39.31456756591797,19.54812240600586,25.406675338745117,0.0,511.0401000											
6	1048515,2018-12-04 04:04:00,0.0,35.4631233215332,22.642351150512695,39.289337158203125,19.54644012451172,25.41153907775879,0.0,513.239318847											
7	1048516,2018-12-04 04:05:00,0.0,35.464576721191406,22.6337833404541,39.25542449951172,19.544395446777344,25.41747283935547,0.0,511.368560791											
8	1048517,2018-12-04 04:06:00,0.0,35.43080520629883,22.623332977294922,39.2889518737793,19.542984008789062,25.423480987548828,0.0,511.37307739											
9	1048518,2018-12-04 04:07:00,0.0,35.3514404296875,22.618316650390625,39.3408088684082,19.542333602905273,25.425111770629883,0.0,511.291778564											

Ilustración 79. Fila con datos en CSV

El resultado del proceso ETL sobre los datos de entrada, nos da una correspondencia de filas válidas de los CSV en cadenas JSON, que para el dato seleccionado da como resultado la mostrada en la Ilustración 80. Se marcan con los mismos colores los datos seleccionados que ahora forman parte de una estructura de diccionario siendo la clave el nombre de la columna.

Cada una de estas filas es tratada por el programa de reingeniería convirtiendo las claves con sus valores en una medida de cada sensor al que corresponden.


```

{
  "id":1048511
  "ReadingTime":"2018-12-04T04:00:00.000+01:00"
  "HS_101_1_SAF_Enable_DATALOG15":0
  "HE_147_1_Lecture_Theatre_3_Humidity_DATALOG12":35.57564926147461
  "TE_2_GFC4_1_Riser_3_Flow_Temp_DATALOG27":22.677305221557617
  "HE_101_1_Return_Duct_Humidity_DATALOG13":39.36426544189453
  "TE_102_2_Cooling_Off_Coil_Temp_DATALOG6":19.554208755493164
  "TE_2_GFC4_2_Riser_3_Return_Temp_DATALOG30":25.399761199951172
  "AHU101_EAF_VSD_Speed_DATALOG18":0
  "CO_147_1_Lecture_Theatre_3_CO2_DATALOG10":512.8112182617188
  "TE_101_1_Frost_Coil_Temp_DATALOG3":9.233405113220215
  "HE_147_2_Lecture_Theatre_3_Humidity_DATALOG9":36.640018463134766
  "TE_147_2_Lecture_Theatre_3_Temp_DATALOG8":19.349262237548828
  "CO_147_2_Lecture_Theatre_3_CO2_DATALOG7":462.03070068359375
  "LPHW_2_GFC4_Riser_3_IFM_DATALOG21":112.1678695678711
  "TCV_101_2_Main_CValve_DATALOG25":99
  "TE_147_1_Lecture_Theatre_3_Temp_DATALOG11":19.461599349975586
  "AHU101_Calc_Supply_Setpt_DATALOG22":18
  "TCV_101_3_Frost_Coil_HValve_DATALOG14":0
  "TE_101_2_Supply_Duct_Temp_DATALOG4":13.640135765075684
  "AHU101_SAF_VSD_Speed_DATALOG17":0
  "TE_101_4_Return_Duct_Temp_DATALOG20":17.224287033081055
  "TCV_101_1_Main_HValve_DATALOG23":15.378217697143555
  "MCC02__Fire_Alarm_DATALOG29":0
  "Lecture_Theatre_3_Avg_Temp_DATALOG19":18.678354263305664
  "TE_101_3_Cooling_Off_Coil_Temp_DATALOG5":10.553695678710938
  "HS_101_2_EAF_Enable_DATALOG16":0,"AHU101_Hours_Run_DATALOG31":12786
  "Lecture_Theatre_3_Avg_CO2_DATALOG26":487.42095947265625
  "117_Lecture_Theatre_3_Av__Humidity_DATALOG28":37.19330978393555
  "GF_East_Rads_Energy_Acc_Total_DATALOG2":324529.375
  "AHU101_Hours_Free_Cool_DATALOG32":2126 }

```

Ilustración 80. Fila del CSV en formato JSON

9.2. Transformación a RDF

Para poblar el RDF hay que saber qué tipo de dispositivo estamos midiendo, y de ahí obtener la información para saber las unidades de medida que vamos a instanciar y vincular correctamente la medida a dicho dispositivo. Cada medida generará por tanto una instancia con el siguiente esquema en formato OWL-RDF, Ilustración 81:

```

<owl:NamedIndividual rdf:about="{URI de la medida}">
  <rdf:type rdf:resource="https://saref.etsi.org/core/Measurement"/>
  <core:isMeasuredIn rdf:resource="{URI de la unidad de medida}">
  <core:relatesToProperty rdf:resource="{URI del dispositivo que tomó la medida}">
  <core:hasTimestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime"> {Reading_Time} </core:hasTimestamp>
  <core:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">{valor del campo} </core:hasValue>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">{nombre para la medida}</rdfs:label>
</owl:NamedIndividual>

```

Ilustración 81. Esquema OWL-RDF para cada medida

Trasladando esto a los dos ejemplos elegidos:

```

<!-- https://saref.etsi.org/core/Measure_1048511_HE_147_1_Lecture_Theatre_3_Humidity_2018_12_04T04_00_00_000_01_00 -->

<owl:NamedIndividual rdf:about="https://saref.etsi.org/core/Measure_1048511_HE_147_1_Lecture_Theatre_3_Humidity_2018_12_04T04_00_00_000_01_00">
  <rdf:type rdf:resource="https://saref.etsi.org/core/Measurement"/>
  <core:isMeasuredIn rdf:resource="http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#PercentageRH"/>
  <core:relatesToProperty rdf:resource="http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_AHU101_Ctrls_HE_147_1_Lecture_Theatre_3_Humidity_D12_456"/>
  <core:hasTimestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2018-12-04T04:00:00+01:00</core:hasTimestamp>
  <core:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">35.57564926147461</core:hasValue>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Measure HE 147 1 Lecture Theatre 3 Humidity 2018-12-04T04:00:00.000+01:00</rdfs:label>
</owl:NamedIndividual>

<!-- https://saref.etsi.org/core/Measure_1048511_TE_2_GFC4_1_Riser_3_Flow_Temp_2018_12_04T04_00_00_000_01_00 -->

<owl:NamedIndividual rdf:about="https://saref.etsi.org/core/Measure_1048511_TE_2_GFC4_1_Riser_3_Flow_Temp_2018_12_04T04_00_00_000_01_00">
  <rdf:type rdf:resource="https://saref.etsi.org/core/Measurement"/>
  <core:isMeasuredIn rdf:resource="http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#CelsiusDegrees"/>
  <core:relatesToProperty rdf:resource="http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#NUIG_AHU101_Ctrls_TE_2_GFC4_1_Riser_3_Flow_Temp_D27_457"/>
  <core:hasTimestamp rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2018-12-04T04:00:00+01:00</core:hasTimestamp>
  <core:hasValue rdf:datatype="http://www.w3.org/2001/XMLSchema#decimal">22.677305221557617</core:hasValue>
  <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Measure TE 2 GFC4 1 Riser 3 Flow Temp 2018-12-04T04:00:00.000+01:00</rdfs:label>
</owl:NamedIndividual>

```

Ilustración 82. Ejemplo de medidas en OWL-RDF

9.3. Análisis de los datos: consultas SPARQL y gráficas

Vamos a realizar un análisis del conjunto de datos generados en RDF y validar si son capaces de responder a una serie de preguntas de competencia (CQs) planteadas en el ORSD descrito en el Apartado 6.1.9, para esto se utilizan consultas sobre SPARQL. Emplearemos las capacidades de la biblioteca Owready2 (Apartado 7.10) y su base de datos SQLite3 integrada para que sirvan de motor de consultas sobre SPARQL.

En la aplicación se lanzarán las consultas para las CQ MD_PC06, MD_PC07 y MD_PC08 cuyas preguntas se pueden ver en la Tabla 39, junto con su consulta SAPRQL asociada:

MD_PC06 ¿Cuál fue el histórico de valores para la temperatura media del aire en la sala?

PREFIX : <<http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#>>
PREFIX core: <<https://saref.etsi.org/core/>>
PREFIX s4bldg: <<https://saref.etsi.org/saref4bldg/>>

```

SELECT ?v ?t
WHERE {
  ?s core:relatesToProperty :NUIG_AHU101_Ctrls_Lecture_Theatre_3_Avg_Temp_D19_477 .
  ?s core:hasValue ?v .
  ?s core:hasTimestamp ?t .
}
ORDER BY ?t

```

MD_PC07 ¿Cuál fue el histórico de valores de la humedad media del aire en la sala entre las fechas 2018-12-04T04:00:00+01:00y 2018-12-04T05:03:00+01:00?

PREFIX : <<http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#>>

PREFIX core: <<https://saref.etsi.org/core/>>

PREFIX s4bldg: <<https://saref.etsi.org/saref4bldg/>>

SELECT ?v ?t

WHERE {

?s core:relatesToProperty :[NUIG_AHU101_Ctrls_117_Lecture_Theatre_3_Av_Humidity_D28_482](#) .

?s core:hasValue ?v .

?s core:hasTimestamp ?t .

FILTER(?t >= '2018-12-04T04:00:00+01:00'^^xsd:dateTime) .

FILTER(?t <= '2018-12-05T05:03:00+01:00'^^xsd:dateTime) .

}

ORDER BY ?t

MD_PC08 ¿Cuál fue el histórico con valor máximo y medio de la concentración de CO2-Posición 1 en la sala?

PREFIX : <<http://www.inf.uva.es/tfg/anavaes/alice-perry/ontology#>>

PREFIX core: <<https://saref.etsi.org/core/>>

PREFIX s4bldg: <<https://saref.etsi.org/saref4bldg/>>

SELECT ?v ?t

WHERE {

?s core:relatesToProperty :[NUIG_AHU101_Ctrls_CO_147_1_Lecture_Theatre_3_CO2_D10_462](#) .

?s core:hasValue ?v .

?s core:hasTimestamp ?t .

}

ORDER BY ?t

SELECT ?t (MAX(?v) as ?mx) (AVG(?v) as ?av)

WHERE {

?s core:relatesToProperty :[NUIG_AHU101_Ctrls_CO_147_1_Lecture_Theatre_3_CO2_D10_462](#) .

?s core:hasValue ?v .

?s core:hasTimestamp ?t .

}

ORDER BY ?t

SELECT ?t (MAX(?v) as ?mx) (AVG(?v) as ?av)

WHERE {

?s core:relatesToProperty :[NUIG_AHU101_Ctrls_CO_147_1_Lecture_Theatre_3_CO2_D10_462](#) .

?s core:hasValue ?v .

?s core:hasTimestamp ?t .

}

ORDER BY ?t

Tabla 39. CQ validadas sobre los datos RDF generados

En la Tabla 40, se puede observar un resultado parcial de la consulta MD_PC06, con cuyos resultados completos se elabora la gráfica de la Ilustración 83.

[[18.678354263305664,	datetime.datetime(2018,	12,	4,	4,	0,
tzinfo=datetime.timezone(datetime.timedelta(seconds=3600))],					
[18.67693328857422,	datetime.datetime(2018,	12,	4,	4,	1,
tzinfo=datetime.timezone(datetime.timedelta(seconds=3600))],					
[18.68168830871582,	datetime.datetime(2018,	12,	4,	4,	2,
tzinfo=datetime.timezone(datetime.timedelta(seconds=3600))],					
[18.659936904907227,	datetime.datetime(2018,	12,	4,	4,	3,
tzinfo=datetime.timezone(datetime.timedelta(seconds=3600))],					
[18.653871536254883,	datetime.datetime(2018,	12,	4,	4,	4,
tzinfo=datetime.timezone(datetime.timedelta(seconds=3600))],					
[18.68573570251465,	datetime.datetime(2018,	12,	4,	4,	5,
tzinfo=datetime.timezone(datetime.timedelta(seconds=3600))],					
[18.67418098449707,	datetime.datetime(2018,	12,	4,	4,	6,
tzinfo=datetime.timezone(datetime.timedelta(seconds=3600))],					
[18.660688400268555,	datetime.datetime(2018,	12,	4,	4,	7,
tzinfo=datetime.timezone(datetime.timedelta(seconds=3600))],					

Tabla 40. Resultado parcial de la consulta SPARQL de la MD_PC06

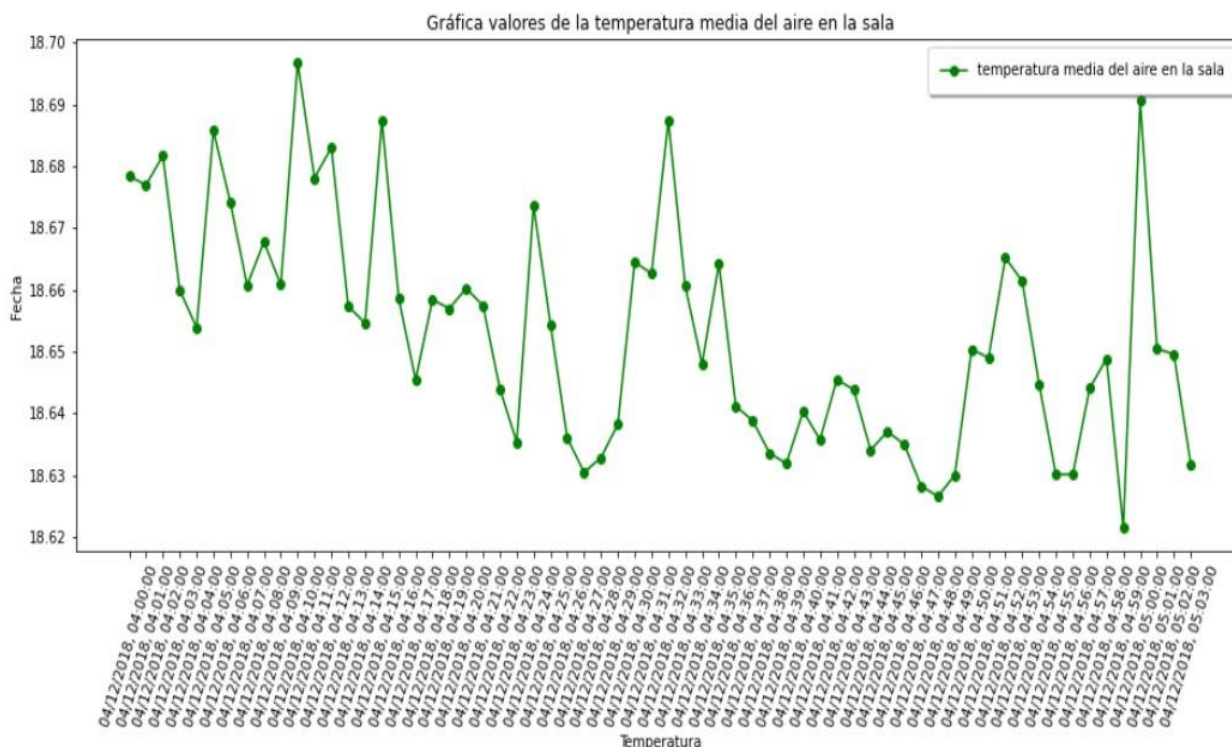


Ilustración 83. Grafica sobre datos RDF asociada a la PQ PC06

También se han realizado unas gráficas con estadísticas descriptivas básicas para validar el potencial de los datos para estudios más complejos. Algunas de estas gráficas son: un histograma sobre valores de temperatura en el sensor Temperatura Media en la Sala que se puede observar en la Ilustración 84 o unos diagramas de caja sobre valores de la temperatura para los sensores “Temperatura en Punto 1” y “Temperatura en Punto 2” de la sala, Ilustración 85.

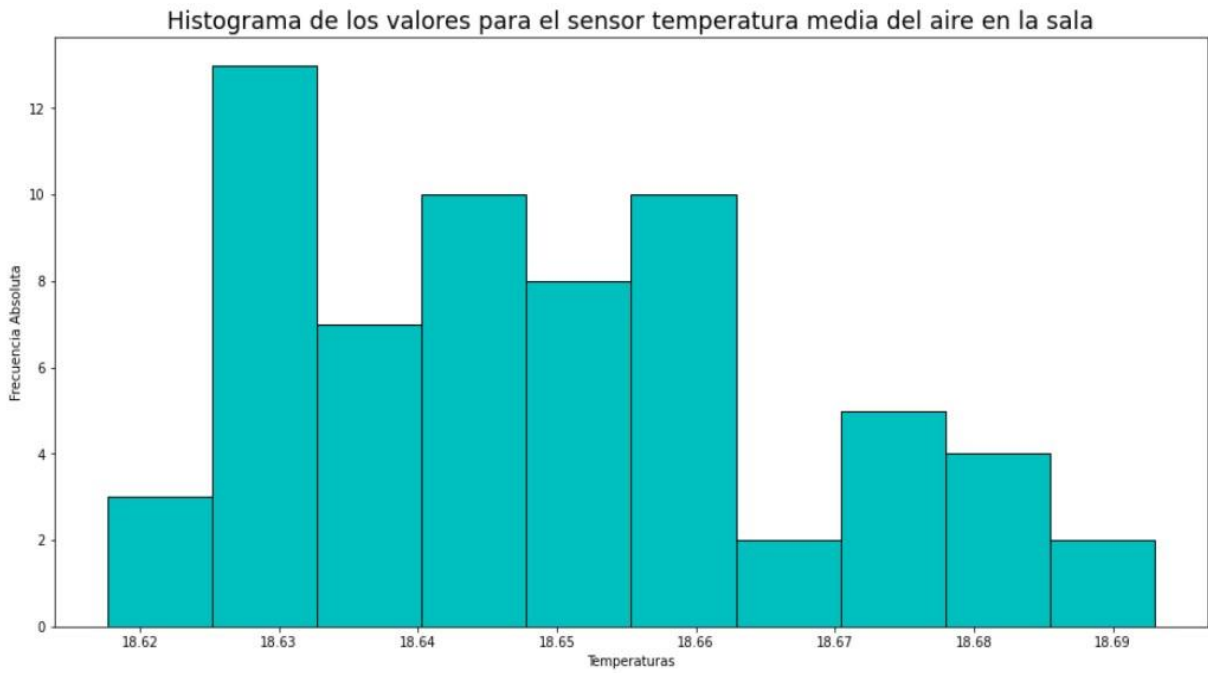


Ilustración 84. Histograma sobre datos RDF asociados a temperaturas en sala

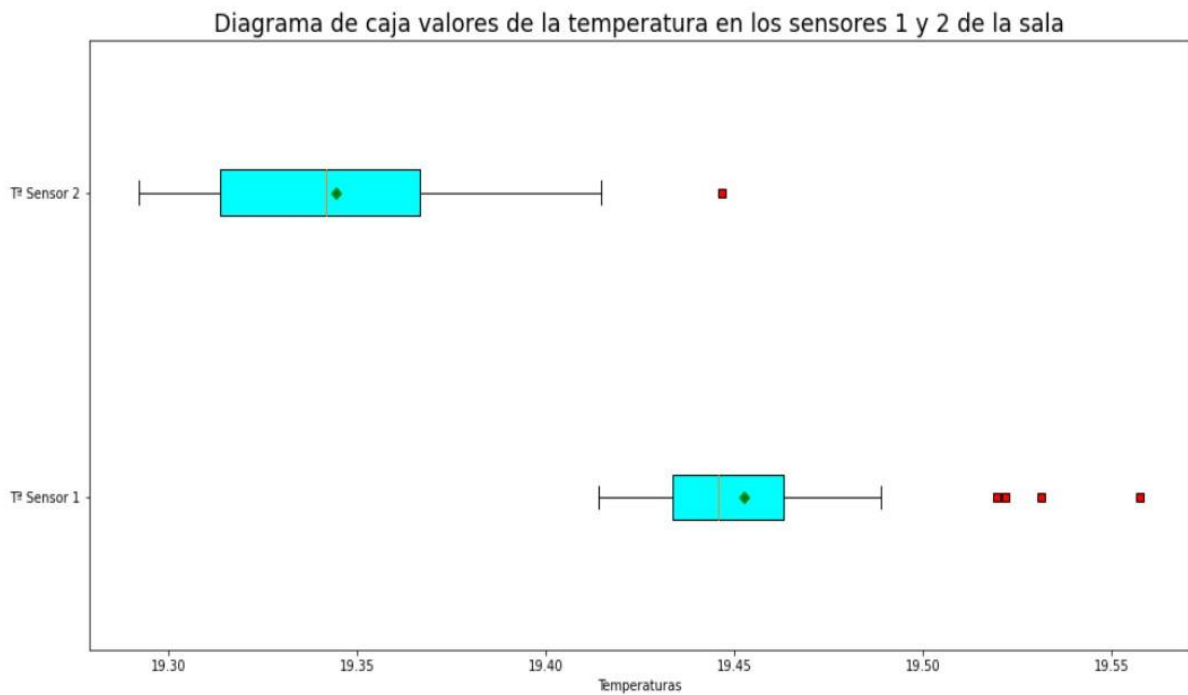


Ilustración 85. Diagrama de Caja sobre datos RDF asociados a temperaturas en sala

Capítulo 10

10. Seguimiento del proyecto

10.1. Introducción

Las primeras reuniones oficiales sobre este proyecto comenzaron en enero de 2021, aunque en noviembre y diciembre del año anterior se llevaron a cabo dos reuniones previas para tratar el alcance del TFG y por ello se realizaron distintos procesos de revisión y adquisición de información que se añaden a la etapa preliminar.

10.2. Fase preliminar de documentación y estudio

El proyecto comenzó con un conocimiento básico sobre ontologías adquirido mediante la realización de un trabajo para una de las asignaturas del plan de estudios, tal y como se comenta en el Apartado 1.1, estos conocimientos formaban un marco teórico que requería una nueva revisión para llevarlo a la práctica y, concretamente, asociarlo con el dominio de los climatizadores.

Uno de los grandes retos de documentación, debido al desconocimiento por parte del alumno en el campo de la Ingeniería Industrial y Arquitectura, ha sido la metodología BIM y su gran importancia y utilización a nivel mundial. La información y documentación encontrada ha sido amplia y organizada, denotando que es una tecnología madura ampliamente implantada.

Con respecto a la carga de trabajo, el esfuerzo mayor ha sido el dedicado a las ontologías. Aunque existe mucha información general sobre ontologías y datos enlazados, y más en los últimos años, donde han experimentado un gran auge, la documentación disponible es de carácter muy general y repetitivo, la mayoría de documentos se vinculan entre ellos y los ejemplos son poco instructivos, repetitivos y alejados de la realidad. Fue de gran ayuda partir del requisito de uso de SAREF ya que permitió conocer los desarrollos y trabajos de la Universidad Politécnica de Madrid a través del *Ontological Engineering Group*.

Para poder adquirir y afianzar la información, se han realizado pruebas menores con muchas de las aplicaciones y tecnologías utilizadas en el proceso, como Protégé o Apache Spark. No se realizó una medición del tiempo dedicado a estas tareas, además, parte de ese tiempo se engloba en el dedicado a la realización del trabajo para la asignatura ya mencionado o en tiempo de enriquecimiento personal y curricular.

En general, se ha estimado el esfuerzo para esta tarea en 200 horas, una carga de tiempo mayor que las posteriores, pero hay que tener en cuenta que uno de los objetivos fundamentales de este trabajo era su carácter de documentación y estudio sobre las materias, reflejado ampliamente en los Capítulos 2 y 3 del proyecto.

10.3. Plan de proyecto: fase de iniciación (19/01/2021-15/02/2021)

Esta fase de arranque del TFG ha ocupado alguna hora más de lo planificado, debido sobre todo a la documentación, decisiones sobre su estructuración y el software para gestionar las referencias bibliográficas. La fase preliminar de revisión también ayudó a reducir los tiempos de algunas tareas.

FASE 1 DE INICIACIÓN				
FASE	TAREA		TIEMPO ESTIMADO	TIEMPO INVERTIDO
Especificación de requisitos	1	Alcance	2	3
Planificación y desarrollo	1	Ciclo de vida y plan	2	1
Planificación y desarrollo	2	Escenarios		
Planificación y desarrollo	3	Actualización plan		
Planificación y desarrollo	4	Restricciones y recursos		
Especificación de requisitos	2,3	Destinatarios y usos	1	
Especificación de requisitos	4	Identificar requisitos	3	4
Especificación de requisitos	5	Requisitos funcionales	2	3
Especificación de requisitos	6	Validación requisitos	2	1
Especificación de requisitos	7	Priorización requisitos		
Especificación de requisitos	8	Términos y frecuencias	3	2
Especificación de requisitos	8-b	Elaboración de ORSD	2	2
Evaluación	1	Evaluación de la fase	1	2
Control	-	Control general		
Documentación	-	Documentación general	2	5
Total (horas)			20	23

Tabla 41. Tareas de la Fase 1

10.4. Fase de reutilización (16/02/2021-23/02/2021)

En esta fase se produjeron algunos desfases entre la estimación y la ejecución que, en una primera revisión, no parecían debidamente justificados. A la vista de estos resultados se decidió alargar la tarea de control. La conclusión alcanzada fue que se había estimado tiempo para la ejecución de procesos que realmente corresponden a tareas de la siguiente fase y no de la de reutilización.

FASE 2 DE REUTILIZACIÓN				
FASE	TAREA		TIEMPO ESTIMADO	TIEMPO INVERTIDO
Búsqueda y reutilización de ontologías	1	Búsqueda de ontologías de dominio	1	1
Búsqueda y reutilización de ontologías	2	Evaluación de ontologías de dominio		
Búsqueda y reutilización de ontologías	3	Selección de ontologías de dominio		
Búsqueda y reutilización de ontologías	4	Integración de ontologías de dominio	4	1
Reutilización de NOR	1	Búsqueda de NOR	2	1
Reutilización de NOR	2	Valoración de NOR		

Reutilización de NOR	3	Selección de recursos no ontológicos	3	1
Evaluación	2	Evaluación de la fase	1	2
Control	-	Control general		
Documentación	-	Documentación general	4	4
Total (horas)			15	10

Tabla 42. Tareas de la Fase 2

10.5. Fase de reingeniería (24/02/2021- 23/03/2021)

En esta fase ha ocurrido lo contrario que la anterior, al integrarse los procesos que se habían estimado para la fase anterior. Igualmente se dedicó algo más de tiempo a las tareas de evaluación y control para verificar que estaba todo conforme a la metodología

FASE 3 DE REINGENIERÍA				
FASE	TAREA		TIEMPO ESTIMADO	TIEMPO INVERTIDO
Reingeniería de NOR	1	Ingeniería inversa de NOR	2	3
Reingeniería de NOR	2a	Exploración de los NOR		
Reingeniería de NOR	2b	ETL sobre los NOR	7	8
Reingeniería de NOR	2c	Transformación de los NOR	6	7
Reingeniería de NOR	3	Ingeniería hacia adelante		
Evaluación	3	Evaluación de la fase	1	2
Control	-	Control general		
Documentación	-	Documentación general	4	4
Total (horas)			20	24

Tabla 43. Tareas de la Fase 3

10.6. Fase de diseño (24/01/2022-08/02/2022)

El desarrollo de esta fase ha sido más largo de lo planificado por dos motivos fundamentales. El primero, debido al tiempo transcurrido con la fase anterior, necesario para retomar el trabajo. El segundo, una carencia generalizada de recursos detectada en las aplicaciones para gráficos de ontologías que han alargado la tarea 3. Se ampliaron las tareas de evaluación y control para tener en cuenta estas circunstancias.

FASE 4 DE DISEÑO				
FASE	TAREA		TIEMPO ESTIMADO	TIEMPO INVERTIDO
Formalización de la ontología	1	Diseño de las URI	5	6
Formalización de la ontología	2	Vocabulario		
Formalización de la ontología	3	Diagrama general	2	4
Evaluación	4	Evaluación de la fase	1	2
Control	-	Control general		
Documentación	-	Documentación general	7	6
Total (horas)			15	18

Tabla 44. Tareas de la Fase 4

10.7. Fase de implementación (04/04/2022-04/05/2022)

Esta fase incluye la finalización de la memoria por lo que su carga de trabajo para las tareas de control, validación y sobre todo documentación es más grande.

FASE 5 DE IMPLEMENTACIÓN				
FASE	TAREA		TIEMPO ESTIMADO	TIEMPO INVERTIDO
Implementación	1	Instalación del entorno	4	3
Implementación	2	Tratamiento de datos	3	4
Implementación	3	Creación de la ontología	12	14
Implementación	4	Generación de RDF y consultas SPARQL	4	3
Validación	1	Validar la ontología en herramientas	3	2
Evaluación	5	Evaluación de la fase	2	3
Control	-	Control general		
Documentación	-	Documentación general	12	15
Total (horas)			40	44

Tabla 45. Tareas de la Fase 5

10.8. Resumen

La finalización del proyecto se ha expandido mucho más de lo esperado en la planificación original. Las dos fases finales se han aplazado y retomado, dejando dos periodos de inactividad que, por motivos personales de carácter familiar, impidieron mi dedicación al proyecto.

PLANIFICACIÓN REAL	
FASE	TIEMPO INVERTIDO
INICIACIÓN (plan de proyecto)	23
REUTILIZACIÓN	10
REINGENIERÍA	24
DISEÑO	18
IMPLEMENTACIÓN	44
Total (horas)	119

Tabla 46. Resumen de planificación real por horas

Con respecto a las horas invertidas, han superado las 310 horas estimadas por el plan de estudio haciendo un total de 319.

Capítulo 11

11. Conclusiones y líneas de trabajo futuras

En este último capítulo de la memoria se pretende realizar una reflexión final sobre todo el trabajo realizado en el marco de desarrollo del TFG y exponer las metas alcanzadas. Para ello se ha tenido en cuenta el conocimiento adquirido para su desarrollo, la interpretación de los resultados obtenidos y su contextualización especializada, dentro de las tecnologías actuales disponibles para el tratamiento de datos asociados al sistema de climatización estudiado.

11.1. Conclusiones

La línea de estudio y revisión sobre el estado actual de las ontologías y la Web Semántica ha sido completada con gran satisfacción personal. Como ya se ha indicado en algún punto del proyecto, el acceso a la información del Ontological Engineering Group de la Universidad Politécnica de Madrid abrió un camino para reforzar y ampliar los conocimientos, así como los desarrollos y metodologías aplicables al IoT en el contexto de los estándares europeos.

El desarrollo y modelado de la ontología sobre la AHU101, bloque central del TFG, también se ha finalizado y validado satisfactoriamente. El proceso de validación que aporta la herramienta OOPS! ha ofrecido recomendaciones que se han podido resolver o son irrelevantes en el contexto del proyecto. Se ha utilizado SAREF y su extensión SAREF4BLDG, siguiendo las pautas de la metodología propuesta por sus propios desarrolladores que permitirá su reutilización en cualquier proyecto futuro. También se han mapeado y validado con ejemplos y gráficas los datos en formato OWL-RDF, disponibles para compartirlos como Datos Enlazados con cualquier entidad.

La implementación de la aplicación en su entorno permite el fácil acceso y comprensión de las modificaciones que se realizan, así como el seguimiento de los datos en su proceso de transformación. Se dejan múltiples puntos de enganche para desarrollos futuros o complementarios, permitiendo recuperar el flujo de datos en cualquier momento. Se ha elegido *Spark* como herramienta para la lectura de datos al ser un software muy conocido en proyectos Big Data por su versatilidad en formatos de entrada (incluyendo *streaming*) y la posibilidad de trabajar de forma distribuida. *Spark* incluye el procesamiento de datos en paralelo y aporta herramientas para completar la fase ETL, algunas de las cuales han sido empleadas.

Como ya se adelantó en el Apartado 10.2 sobre la fase preliminar de estudio, el acceso a una información de calidad sobre ontologías y Datos Enlazados ha sido un gran inconveniente. La documentación disponible es de carácter muy general con documentos que se vinculan entre ellos y ejemplos poco instructivos. Muchas herramientas de acceso libre no han tenido continuidad o desarrollo o carecen de utilidad fuera del ámbito académico. La búsqueda de información útil y la prueba de herramientas resultaron de una complejidad mayor de lo esperado.

Se ha echado en falta también una mayor disponibilidad de datos para el mapeado que hubieran supuesto un mayor desarrollo de la parte de análisis y exploración sobre RDF, pero debido a las condiciones de seguridad y protección no ha sido posible disponer de ellos. Se considera, no obstante, suficiente el desarrollo realizado como para comprobar la capacidad del sistema.

Para finalizar este apartado, en cuanto a los objetivos personales, han sido todos alcanzados y superados.

11.2. Líneas de trabajo futuras

Como se ha indicado en el apartado anterior, la herramienta desarrollada en este proyecto es funcional, admite otros tipos de entrada de datos para su análisis y contine numerosos puntos de enganche a lo largo de su código para desarrollos paralelos del mismo, o ampliaciones futuras. La parte más importante del trabajo es el modelado de la AHU101 como ontología adaptada al estándar SAREF según el proceso de validación realizado, y está separado en un archivo tal y como recomienda la metodología, por lo que puede ser aplicado y reutilizado en cualquier proyecto ontológico futuro que requiera de este dominio de conocimiento.

Por supuesto, el área de conocimiento revisada es muy grande y se han quedado sin cubrir aspectos de desarrollo importantes que sería interesante abordar en futuros proyectos para ampliar el estudio realizado. Como posibles mejoras futuras realizables se destacan las siguientes:

- Inclusión de bases de datos sobre grafos o bases de datos NoSQL que admitan el motor de consulta SPARQL. Esta inclusión permitiría la generación de paneles para el análisis de datos directamente sobre el servidor o con conexiones a aplicaciones de visualización de datos que no se han podido desarrollar en el proyecto.
- Aplicar razonamiento sobre la base de conocimientos ontológicos y realizar un estudio de diversos razonadores y la utilidad de la información generada. La base de conocimientos ya está cargada en la aplicación por lo que quedaría cargar más datos y diseñar una batería de consultas.
- Ampliar el desarrollo del modelo a otros ámbitos de la AHU101, como el de las localizaciones, y completar las recomendaciones ofrecidas por la herramienta OOPS! al respecto
- Aprovechar las características de la tecnología para unir a esta base de conocimientos otras del mismo dominio como pueden ser otras unidades AHU del edificio, o de dominios relacionados como, por ejemplo, una conexión con un servicio meteorológico y aplicar consultas o razonamientos sobre ambos
- De manera más improbable por los requisitos de seguridad y tratamiento de datos, conectar el sistema al *Data Lake* del edificio en estudio y desarrollar un proyecto *Big Data* en tiempo real para verificar si el desarrollo actual es escalable

Bibliografía

- [1] D. Sánchez Martínez, «BIM : Hacia la alineación de intereses en la construcción», *Ing. Civ.*, p. 6, 2017, Accedido: ene. 28, 2022. [En línea]. Disponible en: <https://ingcivileng.com/2017/12/03/bim-hacia-la-alineacion-de-intereses-en-la-construccion/>.
- [2] C. Eastman, «An Outline of the Building Description System», *Carnegie-Mellon Univ., Pittsburgh, Pa. Inst. Phys. Planning.*, p. 23, 1974.
- [3] P. Shimonti, «BIM adoption around the world: how good are we?», *Geospatial World*, 2018. <https://www.geospatialworld.net/article/bim-adoption-around-the-world-how-good-are-we/> (accedido ene. 28, 2022).
- [4] BuildingSMART, «BuildingSMART - international home of openBIM», *BuildingSMART*, 2016. <https://www.buildingsmart.es/bim/qué-es/> (accedido dic. 10, 2021).
- [5] E. González, J. D. Piñeiro, J. Toledo, R. Arnay, y L. Acosta, «An approach based on the ifcOWL ontology to support indoor navigation», *Egypt. Informatics J.*, vol. 22, n.º 1, pp. 1-13, mar. 2021, doi: 10.1016/j.eij.2020.02.008.
- [6] buildingSMART International Limited, «Industry Foundation Classes 4.0.2.1», *buildingSMART International*, 2019. https://standards.buildingsmart.org/IFC/RELEASE/IFC4/ADD2_TC1/HTML/ (accedido ene. 15, 2022).
- [7] buildingSMART International Limited, «IfcAlarm», *buildingSMART International Limited*, 2019. https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC1/HTML/schema/ifcbuildingcontrolsdomain/lexical/ifcalarm.htm (accedido dic. 12, 2021).
- [8] Fundación Wikimedia, «ISO 10303», *wikipedia.org*. https://es.wikipedia.org/wiki/ISO_10303 (accedido ene. 15, 2022).
- [9] L. Cerrado *et al.*, «Enfoque para Implementación de Inspección Dimensional y Geométrica en Lazo Cerrado Basado en el Estándar ISO 10303 STEP/STEP-NC», *Conf. XIII Congr. Ibero-Americano Eng. Mecânica At Lisboa - Port.*, 2017, Accedido: ene. 31, 2022. [En línea]. Disponible en: <https://www.researchgate.net/publication/320720330>.
- [10] B. Kids, R. Kahn, L. Pouzin, y M. Andreessen, «Tim Berners-Lee», *World Wide Web Consort.*, pp. 13-16, 2013, Accedido: ene. 15, 2022. [En línea]. Disponible en: <https://www.w3.org/People/Berners-Lee/Overview.html>.
- [11] T. Berners-Lee, «The World Wide Web: Past, Present and Future», *World Wide Web Consortium (W3C)*, 1996. <https://www.w3.org/People/Berners-Lee/1996/ppf.html> (accedido ene. 15, 2022).
- [12] T. O'Reilly, «What Is Web 2.0 - O'Reilly Media», *O'Reilly Media, Inc.*, sep. 30, 2005. <https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html> (accedido ene. 15, 2022).
- [13] W. L. Hosch, «Web 2.0 | Definition & Examples | Britannica», *Encyclopedia Britannica*. <https://www.britannica.com/topic/Web-20> (accedido ene. 15, 2022).
- [14] J. L. Alonso, C. Carranza, P. Castells, M. Rico, y R. Lara, «Second International Semantic Web Conference (ISWC 2003) Posters and Demonstrations», 2003.
- [15] flatworldbusiness, «Digital Evolution |», <https://flatworldbusiness.wordpress.com>. <https://flatworldbusiness.wordpress.com/digital-evolution/> (accedido ene. 15, 2022).
- [16] T. Berners-Lee, J. Hendler, y O. Lassila, «The semantic web», *Sci. Am.*, vol. 284, pp. 1-4, may 2001, doi: 10.1038/SCIENTIFICAMERICAN0501-34.
- [17] T. Berners-Lee, «Tim Berners-Lee - Semantic Web», *World Wide Web Consortium (W3C)*. <https://www.w3.org/2000/Talks/0906-xmlweb-tbl/text.htm> (accedido ene. 15, 2022).
- [18] Semantic Web Activity Lead, «W3C Semantic Web Activity Homepage», *World Wide Web Consortium (W3C)*. <https://www.w3.org/2001/sw/> (accedido ene. 15, 2022).
- [19] T. Berners-Lee, «Semantic Web - XML2000 - slide "Architecture"», *Semantic Web - XML2000*. <https://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html> (accedido dic. 06, 2021).
- [20] John Sowa, «Semantics for Interoperable Systems», 2021. <https://www.jfsowa.com/ikl/> (accedido ene. 18, 2022).

- [21] Wikipedia, «Semantic Web Stack», *wikipedia.org*. https://en.wikipedia.org/wiki/Semantic_Web_Stack (accedido dic. 06, 2021).
- [22] The World Wide Web Consortium, «Data - W3C», *The Serial Librarian*, 2011. <https://www.w3.org/standards/semanticweb/data> (accedido feb. 12, 2022).
- [23] J. McCrae *et al.*, «The Linked Open Data Cloud», 2019. <https://lod-cloud.net/> (accedido feb. 12, 2022).
- [24] P. Hitzler, «A review of the semantic web field», *Communications of the ACM*, vol. 64, n.º 2. ACM PUB27 New York, NY, USA, pp. 76-83, ene. 25, 2021, doi: 10.1145/3397512.
- [25] M. Lucioni, «Universal phylogenetic tree in rooted form, showing the three domains. According to C. Woese *et al.* 1990», *wikipedia.org*, 2013. https://en.m.wikipedia.org/wiki/File:PhylogeneticTree,_Woese_1990.PNG (accedido ene. 19, 2022).
- [26] I. T. 46/SC 9, *ISO 2788:1986, Documentation -- Guidelines for the establishment and development of monolingual thesauri*. 2007.
- [27] T. R. Gruber, «A translation approach to portable ontology specifications», *Knowl. Acquis.*, vol. 5, n.º 2, pp. 199-220, 1993, doi: 10.1006/knac.1993.1008.
- [28] O. Corcho, M. Fernández-López, y A. Gómez-Pérez, «Ontological Engineering: What Are Ontologies and How Can We Build Them?», 2007.
- [29] A. Gómez-Pérez, M. Fernández-López, y O. Corcho, *Ontological engineering: with examples from the areas of knowledge management, e-commerce and the Semantic Web / Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho*. Springer, 2004.
- [30] I. Flores Vitelli, «Introducción al Razonamiento Sobre Ontologías», en *Lecturas en Ciencias de la Computación*, 2011, p. 29.
- [31] F. Baader, «The description logic handbook : theory, implementation, and applications», p. 555, 2003.
- [32] Wikipedia.org, «Ontology language - Wikipedia», *wikipedia.org*, 2022. https://en.wikipedia.org/wiki/Ontology_language (accedido nov. 04, 2021).
- [33] «VLDB Endowment Inc.» <https://vldb.org/> (accedido ene. 21, 2022).
- [34] E. Ukpe y S. M. F. D. S. Mustapha, «An Analysis of Six Standard Ontology Editing Tools for Capturing Entire Crop Processing», *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, n.º 1, pp. 18-24, 2016, Accedido: ene. 21, 2022. [En línea]. Disponible en: <https://sites.google.com/site/ijcsis/>.
- [35] «Software • Corporate Semantic Web • Department of Mathematics and Computer Science». <https://www.mi.fu-berlin.de/en/inf/groups/ag-csw/Research/Software/index.html> (accedido ene. 21, 2022).
- [36] E. Tonkin, H. D. Pfeiffer, y A. Hewson, «An evidence-based approach to collaborative ontology development», en *Proceedings of the International Symposium on Matching and Meaning Automated Development, Evolution and Interpretation of Ontologies - A Symposium at the AISB 2010 Convention*, 2010, pp. 39-41.
- [37] E. E. Barber *et al.*, «Metodologías para el diseño de ontologías Web», vol. 8327, n.º diciembre, pp. 13-36, 2018.
- [38] D. Stuart, *Practical Ontologies for Information Professionals*. Facet, 2016.
- [39] Y. Sure, S. Staab, y R. Studer, «On-To-Knowledge Methodology (OTKM)», en *Handbook on Ontologies*, Springer, Berlin, Heidelberg, 2004, pp. 117-132.
- [40] O. Corcho, M. Fernández-López, A. Gómez-Pérez, y A. López-Cima, «Building legal ontologies with METHONTOLOGY and WebODE», en *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2005, vol. 3369 LNAI, pp. 142-157, doi: 10.1007/978-3-540-32253-5_9.
- [41] M. Fernandez, A. Gómez-Pérez, y N. Juristo, «Methontology: from ontological art towards ontological engineering», en *Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering*, 1997, n.º May 2014, pp. 33–40, [En línea]. Disponible en: <http://speech.inesc.pt/~joana/prc/artigos/06c METHONTOLOGY from Ontological Art towards Ontological Engineering - Fernandez, Perez, Juristo - AAAI - 1997.pdf>.
- [42] NeOn Project, «NeOn Book NeOn Methodology in a Nutshell», *NeOn Project*, 2010. http://neon-project.org/nw/NeOn_Book.html (accedido feb. 05, 2022).
- [43] I. Jacobs y N. Walsh, «Architecture of the World Wide Web, Volume One», *World Wide Web Consort.*, n.º December, pp. 1-54, 2004, Accedido: ene. 22, 2022. [En línea]. Disponible en: <https://www.w3.org/TR/webarch/>.
- [44] W3C, «Extensible Markup Language (XML) 1.0 (Fifth Edition)», *The World Wide Web Consortium*, 2008. <https://www.w3.org/TR/REC-xml/> (accedido ene. 22, 2022).

- [45] Wikipedia.org, «List of XML markup languages», *Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/List_of_XML_markup_languages (accedido ene. 22, 2022).
- [46] W3C, «W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures», *Language*, 2012. <https://www.w3.org/TR/xmlschema11-1/> (accedido ene. 22, 2022).
- [47] RDF Working Group, «RDF - Semantic Web Standards», *W3C Recommendation February 2014*. 2014, Accedido: feb. 22, 2022. [En línea]. Disponible en: <https://www.w3.org/RDF/>.
- [48] D. Brickley y R. V Guha, «RDF Schema 1.1», *W3C Recomm.*, 2004, Accedido: feb. 22, 2022. [En línea]. Disponible en: <https://www.w3.org/TR/rdf-schema/>.
- [49] S. U. Deborah, L. McGuinness (Knowledge Systems Laboratory) y A. Frank van Harmelen (Vrije universiteit), «OWL Web Ontology Language Overview», *W3C Recomm.*, pp. 1-22, 2004, Accedido: ene. 23, 2022. [En línea]. Disponible en: <https://www.w3.org/TR/owl-features/>.
- [50] M. Krötzsch, F. Simančík, y I. Horrocks, «A description logic primer», en *Perspectives on Ontology Learning*, vol. 18, 2014, pp. 3-20.
- [51] W3C, «OWL 2 Web Ontology Language Quick Reference Guide (Second Edition)», n.º December, pp. 1-13, 2012, Accedido: ene. 23, 2022. [En línea]. Disponible en: <https://www.w3.org/TR/owl2-quick-reference/>.
- [52] M. Horridge y P. F. Patel-Schneider, «OWL 2 Web Ontology Language Manchester Syntax (Second Edition)», *W3C Work. Gr. Note*, n.º December 2012, pp. 1-10, 2012, Accedido: feb. 23, 2022. [En línea]. Disponible en: <https://www.w3.org/TR/owl2-manchester-syntax/>.
- [53] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, y G. Carothers, «RDF 1.1 Turtle», *W3C Recommendation*, 2014. <https://www.w3.org/TR/turtle/> (accedido feb. 23, 2022).
- [54] M. Hori, P. F. Patel-schneider, y L. Technologies, «OWL Web Ontology Language XML Presentation Syntax», *Structure*, n.º June, pp. 1-31, 2003, Accedido: ene. 23, 2022. [En línea]. Disponible en: <https://www.w3.org/TR/owl-xmlsyntax/>.
- [55] M. Ahmed, «The Web Ontology Language (WOL) - Technologies In Industry 4.0», *Technologies In Industry 4.0*, 2021. <https://www.technologiesinindustry4.com/2021/07/the-web-ontology-language-wol.html> (accedido ene. 28, 2022).
- [56] M. Poveda-Villalón, A. Gómez-Pérez, y M. C. Suárez-Figueroa, «OOPS! (OntOlogy Pitfall Scanner!)», *Int. J. Semant. Web Inf. Syst.*, vol. 10, n.º 2, pp. 7-34, 2014, doi: 10.4018/ijswis.2014040102.
- [57] P. V Biron y A. Malhotra, «XML Schema Part 2: Datatypes Second Edition», *W3C Recommendation*. 2004, Accedido: ene. 24, 2022. [En línea]. Disponible en: <https://www.w3.org/TR/xmlschema-2/>.
- [58] E. P. Hommeaux y A. Seaborne, «SPARQL Query Language for RDF», *W3C Recomm.* <http://www.w3.org/TR/rdf-sparql-query/>, 2008, Accedido: feb. 23, 2022. [En línea]. Disponible en: <https://www.w3.org/TR/rdf-sparql-query/>.
- [59] Stardog Union, «Learn SPARQL | Stardog Documentation Latest», 2022. <https://docs.stardog.com/tutorials/learn-sparql> (accedido ene. 23, 2022).
- [60] M. D. Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosf, «SWRL: A Semantic Web Rule Language Combining OWL and RuleML», 2016. Accedido: ene. 23, 2022. [En línea]. Disponible en: <https://www.w3.org/Submission/SWRL/>.
- [61] H. P. Alesso y C. F. Smith, «Semantic Web Rule Language», en *Thinking on the Web*, 2006, pp. 161-167.
- [62] R. and A.-C. E. American Society of Heating, 2008 *Ashrae Handbook: HVAC Systems and Equipment, I-P Edition*. 2008.
- [63] S. Tsemekidi Tzeiranaki *et al.*, *Energy consumption and energy efficiency trends in the EU-28 for the period 2000-2016*. 2018.
- [64] US Department of Energy, «Frequently Asked Questions (FAQs) - U.S. Energy Information Administration (EIA)», *Eia*, 2019. <https://www.eia.gov/tools/faqs/faq.php?id=1174&t=1> (accedido feb. 01, 2022).
- [65] S. Dr.Ghosh, «Evaporative Cooling in Greenhouses | Cooling India Monthly Business Magazine on the HVACR Business | Green HVAC industry | Heating, Ventilation, Air conditioning and Refrigeration News Magazine Updates, Articles, Publications on HVACR Business Industry |», *Cooling India*, 2019. <https://www.coolingindia.in/strategies-for-ameliorating-energy-efficiency-in-hvac/> (accedido dic. 12, 2021).
- [66] «Building-Energy-Consumption-Chart - BlueHat Mechanical». [https://bluehatmechanical.com/save-on-the-biggest-use-of-power-in-your-building/building-energy-consumption-chart/#lightbox\[postimages\]/0](https://bluehatmechanical.com/save-on-the-biggest-use-of-power-in-your-building/building-energy-consumption-chart/#lightbox[postimages]/0) (accedido feb. 01, 2022).

- [67] cleantech group, «Heating, Ventilation and Air Conditioning (HVAC) Systems, the Next Building Load to Optimize | Cleantech Group». <https://www.cleantech.com/heating-ventilation-and-air-conditioning-hvac-systems-the-next-building-load-to-optimize/> (accedido feb. 01, 2022).
- [68] Ashrae, «Thermal Environmental Conditions for Human Occupancy», *ANSI/ASHRAE Stand. 55-2004*, vol. 2004, p. 30, 2004, Accedido: dic. 12, 2021. [En línea]. Disponible en: www.ashrae.org.
- [69] Official Journal of the European Union, «COMMISSION REGULATION (EU) No 1253/2014», *L 337/8*, 2014. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.337.01.0008.01.ENG (accedido feb. 12, 2022).
- [70] E. L. P. Europeo *et al.*, «Directiva (UE) 2018/844 del Parlamento Europeo y del Consejo de 30 de mayo de 2018 por la que se modifica la Directiva 2010/31/UE relativa a la eficiencia energética de los edificios y la Directiva 2012/27/UE relativa a la eficiencia energética», *Ars Iuris Salmant.*, vol. 6, n.º 2, pp. 253-254, 2019.
- [71] C. C. S. Raymond, «EEIP: Optimización energética mediante inteligencia artificial (IA) en sistemas de climatización», *Eficiencia Energética en Procesos Industriales Organization*, 2022. <https://ee-ip.org/es/article/optimizacion-energetica-mediante-inteligencia-artificial-ia-en-sistemas-de-climatizacion-6324> (accedido ene. 20, 2022).
- [72] Climagold, «OPTIMA | Clima Gold». <https://en.climagold.com/project/optima/> (accedido feb. 01, 2022).
- [73] Wikipedia, «Air handler - Wikipedia», *en.wikipedia.org*, 2020. https://en.wikipedia.org/wiki/Air_handler (accedido dic. 10, 2021).
- [74] Entropic, «Data Centre air handling units (AHUs) typical features -». <https://entropic.ie/air-handling-units-ahus-typical-features> (accedido ene. 09, 2022).
- [75] ISO, «ISO 16739-1:2018 - Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries — Part 1: Data schema», *ISO*, 2018. <https://www.iso.org/standard/70303.html> (accedido ene. 16, 2022).
- [76] V. Bazjanac y T. Maile, «IFC HVAC interface to EnergyPlus - A case of expanded interoperability for energy simulation», en *Proceedings of Building Simulation 2004. First National Conference of IBPSA-USA*, 2004, pp. 31-37, Accedido: ene. 12, 2022. [En línea]. Disponible en: <https://www.osti.gov/biblio/840312>.
- [77] M. Marini, C. C. Mastino, R. Baccoli, y A. Frattolillo, «“BIM and PLANT SYSTEMS: A SPECIFIC ASSESSMENT”», en *Energy Procedia*, 2018, vol. 148, pp. 623-630, doi: 10.1016/j.egypro.2018.08.150.
- [78] A. Ludovic, A. Ku Leuven, R. Klein, K. U. Leuven, A. Andriamamonjy, y D. Saelens, «An auto-deployed model-based fault detection and diagnosis approach for Air Handling Units using BIM and Modelica An anthropological inquiry by design towards improving indoor air quality within hospital settings View project Flemish Cities in Transition-», 2018, doi: 10.1016/j.autcon.2018.09.016.
- [79] P. Pauwels y W. Terkaj, «EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology», *Autom. Constr.*, vol. 63, pp. 100-133, mar. 2016, doi: 10.1016/j.autcon.2015.12.003.
- [80] buildingSMART, «ifcOWL ontology (IFC4_ADD2_TC1)», 2019. https://standards.buildingsmart.org/IFC/DEV/IFC4/ADD2_TC1/OWL/index.html (accedido ene. 24, 2022).
- [81] S. Antipolis, «ETSI releases 3 new ontology specifications for Smart Cities, Industry 4.0 and Smart Agriculture». 2019, Accedido: feb. 24, 2022. [En línea]. Disponible en: <https://www.etsi.org/newsroom/press-releases/1620-2019-06-etsi-releases-3-new-ontology-specifications-for-smart-cities-industry-4-0-and-smart-agriculture>.
- [82] European Standards Organization (ETSI), «SAREF Portal». <https://saref.etsi.org/> (accedido ene. 24, 2022).
- [83] ETSI, «TS 103 410-3 - V1.1.2 - SmartM2M; Extension to SAREF; Part 3: Building Domain», vol. 1, pp. 1-38, 2020, Accedido: feb. 24, 2022. [En línea]. Disponible en: <https://portal.etsi.org/TB/ETSIDeliverableStatus.aspx>.
- [84] W. Li, G. Tropea, A. Abid, A. Detti, y F. Le Gall, «Review of standard ontologies for the web of things», 2019, doi: 10.1109/GIOTS.2019.8766377.
- [85] M. D. F. Baonza, A. Pérez, y B. Villazón, «NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse», 2008.
- [86] A. Gómez-Pérez, E. Motta, y M. Carmen Suárez-Figueroa, «The NeOn Ontology Engineering Methodology Handbook», 2009, Accedido: ene. 30, 2022. [En línea]. Disponible en:

- <http://ontologydesignpatterns.org/>.
- [87] A. G. and M. C. S.- Figueroa, «Neon methodology for building ontology networks: a Scenario- Based Methodology», *Demetra EOOD*, n.º February, pp. 1-18, 2009, [En línea]. Disponible en: <http://kmi.open.ac.uk/events/sssw08/presentations/Gomez Perez-NeOn-Methodology-OntologySpecification-v3.pdf>.
- [88] V. Presutti *et al.*, «D2.5.1: A Library of Ontology Design Patterns: reusable solutions for collaborative design of networked ontologies.», 2006. Accedido: feb. 06, 2022. [En línea]. Disponible en: http://neon-project.org/deliverables/WP2/NeOn_2008_D2.5.1.pdf.
- [89] F. Priyatna y B. Villazón-Terrazas, «Building ontologies by using re-engineering patterns and R2RML mappings», en *CEUR Workshop Proceedings*, 2012, vol. 929, pp. 109-120, Accedido: may 09, 2022. [En línea]. Disponible en: <http://ontologydesignpatterns.org/wiki/Submissions:ReengineeringODPs>.
- [90] PMI, *PMBOK Guide | Project Management Institute*. 2017.
- [91] J. B. Lamy, «Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies», *Artif. Intell. Med.*, vol. 80, pp. 11-28, jul. 2017, doi: 10.1016/j.artmed.2017.07.002.

Apéndice A

Contenido de la memoria

Los contenidos de la carpeta en la dirección *gitlab* de la Escuela de Ingeniería Informática asociada a este proyecto:

https://gitlab.inf.uva.es/angnava/tfg_informatica/

Con la siguiente estructura y contenidos:

- Carpeta **TFG** :
 - **proyecto_angnava.ipynb**: código fuente del proyecto
 - **DelAHU101_2018.csv**, **DelAHU101_2019.csv**: archivos csv con los datos
 - **NUIG_variables_MR1_MP5_AHU101.xlsx**: excel con la información sobre los sensores de los datos y su distribución en la AHU101
 - **NUIG_variables_MR1_MP5_AHU101-comentado.xlsx**: anotaciones de expertos en el dominio sobre el tipo de variables en el excel anterior
 - **schema.json**: metadatos de los archivos csv
 - **ontologia.rdf**: archivo OWL-RDF con la ontología desarrollada
 - **saref.rdf**: ontología SAREF en formato XML-RDF
 - **saref4bldg.rdf**: ontología SAREF4BLDG en formato XML-RDF
 - **resultados.rdf**: ejemplo de datos creados con el programa, en formato XML-RDF
 - **ontology.sqlite3**: BD de ejemplo creada con el programa, en formato sqlite3

- Carpeta **Memoria**:
 - **memoria.pdf**: la presente memoria en formato PDF
 - **ontologia-lode.html**: versión HTML en formato estándar de la ontología desarrollada
 - **traza-cuaderno.html**: versión HTML con la traza de ejecución del programa
 - Carpeta **lode**: archivos para visualizar correctamente ***ontologia-lode.html***

Apéndice B











Reutilización de elementos

A continuación, se detallan todos los elementos (clases ● , propiedades ■ y relaciones ■) reutilizados de las ontologías SAREF, SAREF4BLDG o sus importaciones GEO y TIME en la ontología desarrollada.

Elemento	URI (con prefijo)	Ontología
● AirToAirHeatRecovery	https://saref.etsi.org/saref4bldg/AirToAirHeatRecovery	SAREF4BLDGS
● Alarm	https://saref.etsi.org/saref4bldg/Alarm	SAREF4BLDGS
● Building	https://saref.etsi.org/saref4bldg/Building	SAREF4BLDGS
● BuildingDevice	https://saref.etsi.org/saref4bldg/BuildingDevice	SAREF4BLDGS
● BuildingSpace	https://saref.etsi.org/saref4bldg/BuildingSpace	SAREF4BLDGS
● Coil	https://saref.etsi.org/saref4bldg/Coil	SAREF4BLDGS
● Controller	https://saref.etsi.org/saref4bldg/Controller	SAREF4BLDGS
● Device	https://saref.etsi.org/core/Device	SAREF
● Energy	https://saref.etsi.org/core/Energy	SAREF
● Fan	https://saref.etsi.org/saref4bldg/Fan	SAREF4BLDGS
● Filter	https://saref.etsi.org/saref4bldg/Filter	SAREF4BLDGS
● FlowInstrument	https://saref.etsi.org/saref4bldg/FlowInstrument	SAREF4BLDGS
● FlowTreatmentDevice	https://saref.etsi.org/saref4bldg/FlowTreatmentDevice	SAREF4BLDGS
● Function	https://saref.etsi.org/core/Function	SAREF
● geo:SpatialThing	http://www.w3.org/2003/01/geo/wgs84_pos#SpatialThing	GEO

● Humidity	https://saref.etsi.org/core/Humidity	SAREF
● HVAC	https://saref.etsi.org/core/HVAC	SAREF
● Meter	https://saref.etsi.org/core/Meter	SAREF
● Measurement	https://saref.etsi.org/core/Measurement	SAREF
● Occupancy	https://saref.etsi.org/core/Occupancy	SAREF
● PhysicalObject	https://saref.etsi.org/saref4bldg/PhysicalObject	SAREF4BLDGS
● Property	https://saref.etsi.org/core/Property	SAREF
● SensingFunction	https://saref.etsi.org/core/SensingFunction	SAREF
● Sensor	https://saref.etsi.org/core/Sensor	SAREF
● Service	https://saref.etsi.org/core/Service	SAREF
● Smoke	https://saref.etsi.org/core/Smoke	SAREF
● SmokeSensor	https://saref.etsi.org/core/SmokeSensor	SAREF
● SwitchingDevice	https://saref.etsi.org/saref4bldg/SwitchingDevice	SAREF4BLDGS
● Task	https://saref.etsi.org/core/Task	SAREF
● Temperature	https://saref.etsi.org/core/Temperature	SAREF
● TemperatureSensor	https://saref.etsi.org/core/TemperatureSensor	SAREF
● TemperatureUnit	https://saref.etsi.org/core/TemperatureUnit	SAREF
● Time	https://saref.etsi.org/core/Time	SAREF
● UnitaryControlElement	https://saref.etsi.org/saref4bldg/UnitaryControlElement	SAREF4BLDGS
● UnitOfMeasure	https://saref.etsi.org/core/UnitOfMeasure	SAREF
● Valve	https://saref.etsi.org/saref4bldg/Valve	SAREF4BLDGS
■ consistsOf	https://saref.etsi.org/core/consistOf	SAREF
■ contains	https://saref.etsi.org/saref4bldg/contains	SAREF4BLDGS
■ controlsProperty	https://saref.etsi.org/core/controlsProperty	SAREF
■ geo:location	http://www.w3.org/2003/01/geo/wgs84_pos#location	GEO
■ hasFunction	https://saref.etsi.org/core/hasFunction	SAREF
■ hasSpace	https://saref.etsi.org/saref4bldg/hasSpace	SAREF4BLDGS
■ isContainedIn	https://saref.etsi.org/saref4bldg/isContainedIn	SAREF4BLDGS
■ isControlledByDevice	https://saref.etsi.org/core/isControlledByDevice	SAREF

APÉNDICE B: REUTILIZACIÓN DE ELEMENTOS

 isMeasuredByDevice	https://saref.etsi.org/core/isMeasuredByDevice	SAREF
 isMeasuredIn	https://saref.etsi.org/core/isMeasuredIn	SAREF
 isOfferedBy	https://saref.etsi.org/core/isOfferedBy	SAREF
 isSpaceOf	https://saref.etsi.org/saref4bldg/isSpaceOf	SAREF4BLDGS
 measuresProperty	https://saref.etsi.org/core/measuresProperty	SAREF
 offers	https://saref.etsi.org/core/offers	SAREF
 relatesToMeasurement	https://saref.etsi.org/core/relatesToMeasurement	SAREF
 relatesToProperty	https://saref.etsi.org/core/relatesToProperty	SAREF
 hasValue	https://saref.etsi.org/core/hasValue	SAREF
 hasTimestamp	https://saref.etsi.org/core/hasValue	SAREF

Apéndice C

Ontología AHU101 Alice Perry

A continuación, se detallan todos los elementos (clases ●, propiedades ■, relaciones ■ e instancias ■) de la ontología. Se emplean en las URI para abreviar los prefijos:



















































owl: <http://www.w3.org/2002/07/owl#>
 geo: http://www.w3.org/2003/01/geo/wgs84_pos#
 saref: <https://saref.etsi.org/core/>
 s4bldg: <https://saref.etsi.org/saref4bldg/>
 core: <https://www.uva.es/qii/tfg/anqnav/ontology>

















































































Elemento	URI (con prefijo)	rdf:comment	rdfs:domain	rdfs:range	owl:inverseOf
● owl:Thing					
● AirToAirHeatRecovery					
● Alarm					
● Building	s4bldng:Buiding	A building represents a structure that provides shelter for its occupants or contents and stands in one place...			
● BuildingDevice					
● BuildingSpace					
● Coil	s4bldng:Coil	A coil is a device used to provide heat transfer between non-mixing media...			

● Controller	saref:Controler	A controller is a device that monitors inputs and controls outputs within a building automation system...
● DateTimeUnit		The class of units of measure for date and time
● Duct_Segment		A Duct Segment type is used to define the common properties of a duct segment that may be applied to many occurrences of that type. A duct segment is used to typically join two sections of duct network.
● Energy		A saref:Property related to some measurements that are characterized by a certain value measured in an energy unit...
● Fan		A fan is a device which imparts mechanical work on a gas. A typical usage of a fan is to induce airflow in a building services air distribution system.
● Filter		A filter is an apparatus used to remove particulate or gaseous matter from fluids and gases.
● FlowInstrument		A flow instrument reads and displays the value of a particular property of a system at a point, or displays the difference in the value of a property between two points...
● FlowTreatmentDevice		The distribution flow element FlowTreatmentDevice defines the occurrence of a device typically used to remove unwanted matter from a fluid, either liquid or gas, and typically participates in a flow distribution system.
● geo:SpatialThing		Anything with spatial extent, i.e. size, shape, or position. e.g. people, places, bowling balls, as well as abstract areas like cubes.
● Humidity		A saref:Property related to some measurements that are characterized by a

	certain value that is measured in a humidity unit
● Humidity_sensor	
● HumidityUnit	The class of units of measure for date and time
● HVAC	Heating, Ventilation and Air Conditioning (HVAC) device that provides indoor environmental comfort. A saref:HVAC is typically used to accomplish saref:Comfort.
● Meter	A device built to accurately detect and display a quantity in a form readable by a human being. Further, a device of category saref:Meter that performs a saref:MeteringFunction.
● Occupancy	A saref:Property related to some measurements that are characterized by a certain value (saref:hasValue property) that is measured in a unit of measure for occupancy
● OccupancyUnit	The class of units of measure for occupancy
● owl:Nothing	
● PhysicalObject	Any Object that has a proper space region. (Definition extracted from DUL ontology)
● SensingFunction	A function that allows to transmit data from sensors, such as measurement values (e.g., temperature) or sensing data (e.g., occupancy)
● Sensor	A device that detects and responds to events or changes in the physical environment such as light, motion, or temperature changes...
● Service	A service is a representation of a function to a network that makes the function discoverable, registerable, remotely controllable by other devices in the network...
● Smoke	A saref:Property related to some measurements that are characterized by a

	certain value that is measured in a unit of measure for smoke
● SmokeSensor	A sensor that performs the saref:SensingFunction and the saref:EventFunction, and is used for the purpose of sensing a property of type saref:Smoke....
● SmokeUnit	The class of units of measure for smoke
● Status	Extension: Property related to some measurements that are characterized by a certain value measured in a status or position units
● StatusUnit	The class of units of measure for statuses
● SwitchingDevice	A switch is used in a cable distribution system (electrical circuit) to control or modulate the flow of electricity...
● Temperature	A saref:Property related to some measurements that are characterized by a certain value that is measured in a temperature unit (degree_Celsius, degree_Fahrenheit, or degree_kelvin)
● TemperatureSensor	A sensor that is used for the purpose of sensing a property of type saref:Temperature. A saref:TemperatureSensor is typically used to saref:accomplish saref:Comfort.
● TemperatureUnit	The unit of measure for temperature
● Time	A class that allows to specify the time concept.
● UnitaryControlElement	A unitary control element combines a number of control components into a single product, such as a thermostat or humidistat...
● UnitOfMeasure	he unit of measure is a standard for measurement of a quantity, such as a Property...

 Valve	A valve is used in a building services piping distribution system to control or modulate the flow of the fluid.			
 accomplishes	A relationship between a certain entity (e.g., a device) and the task it accomplishes			 isAccomplishesBy
 consistsOf	A relationship indicating a composite entity that consists of other entities...			
 contains	A relation between a physical space and the objects located in such space.			 isContainedIn
 controlsProperty	A relationship specifying the property that can be controlled by a certain device	 Device	 Property	
 geo:location				
 hasFunction	A relationship identifying the function of a device	 Device	 Function	
 hasSpace				 isSpaceOf
 isContainedIn				 contains
 isControlledByDevice		 Property	 Device	
 isMeasuredByDevice		 Property	 Device	
 isOfferedBy		 Service	 Device	 offers
 isSpaceOf				 hasSpace
 measuresProperty		 Device	 Property	
 offers		 Device	 Service	 isOfferedBy
 retlatesToProperty		 Measurement	 Property	 retlatesToMeasurement
 hasTimestamp				
 hasValue				
 geo:latitude				
 geo:longitude				
URI (con prefijo)		Clase instanciada		Relaciones
 core:Boolean		 StatusUnit		
 core:CelsiusDegrees		 TemperatureUnit		
 core:Hour		 DateTimeUnit		

 core:NUIG_AHU101_Actuator_Ctrls_117_Lecture_Theatre_3_Av_Humidity_D28_482	 Humidity_sensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_AHU101_Calc_Supply_Setpt_D22_470	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_CO_147_1_Lecture_Theatre_3_CO2_D10_462	 SmokeSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_CO_147_2_Lecture_Theatre_3_CO2_D7_466	 SmokeSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_HE_101_1_Return_Duct_Humidity_D13_458	 Humidity_sensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_HE_147_1_Lecture_Theatre_3_Humidity_D12_456	 Humidity_sensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_HE_147_2_Lecture_Theatre_3_Humidity_D9_464	 Humidity_sensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_Lecture_Theatre_3_Avg_CO2_D26_481	 SmokeSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_Lecture_Theatre_3_Avg_Temp_D19_477	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_LPHW_2_GFC4_Riser_3_IFM_D21_467	 Valve	 controlsProperty
 core:NUIG_AHU101_Actuator_Ctrls_TCV_101_1_Main_HVValve_D23_475	 Valve	 controlsProperty
 core:NUIG_AHU101_Actuator_Ctrls_TCV_101_2_Main_CVValve_D25_468	 Valve	 controlsProperty
 core:NUIG_AHU101_Actuator_Ctrls_TCV_101_3_Frost_Coil_HVValve_D14_471	 Valve	 controlsProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_101_1_Frost_Coil_Temp_D3_463	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_101_2_Supply_Duct_Temp_D4_472	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_101_3_Cooling_Off_Coil_Temp_D5_478	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_101_4_Return_Duct_Temp_D20_474	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_102_2_Cooling_Off_Coil_Temp_D6_459	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_147_1_Lecture_Theatre_3_Temp_D11_469	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_147_2_Lecture_Theatre_3_Temp_D8_465	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_2_GFC4_1_Riser_3_Flow_Temp_D27_457	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_Ctrls_TE_2_GFC4_2_Riser_3_Return_Temp_D30_460	 TemperatureSensor	 measuresProperty
 core:NUIG_AHU101_Actuator_EAF_VSD_Speed_D18_461	 FlowInstrument	 controlsProperty
 core:NUIG_AHU101_Actuator_SAF_VSD_Speed_D17_473	 FlowInstrument	 controlsProperty
 core:NUIG_AHU101_Component_Cooling_Coil	 Coil	 consistOf  contains  accomplishes
 core:NUIG_AHU101_Component_Extract_Air_Fan	 Fan	 consistOf  contains  accomplishes

core:NUIG_AHU101_Component_Extract_Bag_Filter	Filter	consistOf contains accomplishes
core:NUIG_AHU101_Component_Extract_Fans_Control	Controller	accomplishes
core:NUIG_AHU101_Component_GFC4_Riser	BuildingDevice	consistOf contains accomplishes
core:NUIG_AHU101_Component_Post_Heating_Coil	Coil	consistOf contains accomplishes
core:NUIG_AHU101_Component_Pre_Heating_Coil	Coil	consistOf contains accomplishes
core:NUIG_AHU101_Component_Supply_Air_Fan	Fan	consistOf contains accomplishes
core:NUIG_AHU101_Component_Supply_Bag_Filter	Filter	consistOf contains accomplishes
core:NUIG_AHU101_Component_Supply_Fans_Control	Controller	accomplishes
core:NUIG_AHU101_Component_Supply_Panel_Filter	Filter	consistOf contains accomplishes
core:NUIG_AHU101_Component_Thermal_Wheel	AirToAirHeatRecovery	accomplishes
core:NUIG_AHU101_Ctrls_117_Lecture_Theatre_3_Av_Humidity_D28_482	Humidity	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_AHU101_Calc_Supply_Setpt_D22_470	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_AHU101_EAF_VSD_Speed_D18_461	Occupancy	isControlledByDevice
core:NUIG_AHU101_Ctrls_AHU101_Hours_Free_Cool_D32_484	Time	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_AHU101_Hours_Run_D31_480	Time	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_AHU101_SAF_VSD_Speed_D17_473	Occupancy	
core:NUIG_AHU101_Ctrls_CO_147_1_Lecture_Theatre_3_CO2_D10_462	Smoke	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_CO_147_2_Lecture_Theatre_3_CO2_D7_466	Smoke	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_HE_101_1_Return_Duct_Humidity_D13_458	Humidity	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_HE_147_1_Lecture_Theatre_3_Humidity_D12_456	Humidity	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_HE_147_2_Lecture_Theatre_3_Humidity_D9_464	Humidity	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_HS_101_1_SAF_Enable_D15_455	Status	isControlledByDevice
core:NUIG_AHU101_Ctrls_HS_101_2_EAF_Enable_D16_479	Status	isControlledByDevice
core:NUIG_AHU101_Ctrls_Lecture_Theatre_3_Avg_CO2_D26_481	Smoke	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_Lecture_Theatre_3_Avg_Temp_D19_477	Temperature	isMeasuredByDevice

core:NUIG_AHU101_Ctrls_LPHW_2_GFC4_Riser_3_IFM_D21_467	Occupancy	isControlledByDevice
core:NUIG_AHU101_Ctrls_MCC02__Fire_Alarm_D29_476	Status	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_MCC02__Gas_Alarm	SmokeSensor	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TCV_101_1_Main_HVAlve_D23_475	Occupancy	isControlledByDevice
core:NUIG_AHU101_Ctrls_TCV_101_2_Main_CVAlve_D25_468	Occupancy	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TCV_101_3_Frost_Coil_HVAlve_D14_471	Occupancy	isControlledByDevice
core:NUIG_AHU101_Ctrls_TE_101_1_Frost_Coil_Temp_D3_463	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TE_101_2_Supply_Duct_Temp_D4_472	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TE_101_3_Cooling_Off_Coil_Temp_D5_478	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TE_101_4_Return_Duct_Temp_D20_474	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TE_102_2_Cooling_Off_Coil_Temp_D6_459	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TE_147_1_Lecture_Theatre_3_Temp_D11_469	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TE_147_2_Lecture_Theatre_3_Temp_D8_465	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TE_2_GFC4_1_Riser_3_Flow_Temp_D27_457	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Ctrls_TE_2_GFC4_2_Riser_3_Return_Temp_D30_460	Temperature	isMeasuredByDevice
core:NUIG_AHU101_Engineering_Building	HVAC	contains isContainedIn hasSpace isSpaceOf consistOf offers
core:NUIG_AHU101_Exhaust_Air_Outlet	Duct_Segment	
core:NUIG_AHU101_Fresh_Air_Inlet	Duct_Segment	
core:NUIG_AHU101_Meter_Ctrls_AHU101_Hours_Free_Cool_D32_484	Meter	measuresProperty
core:NUIG_AHU101_Meter_Ctrls_AHU101_Hours_Run_D31_480	Meter	measuresProperty
core:NUIG_AHU101_Position_1_SAF_Enable_D15_455	SwitchingDevice	controlsProperty
core:NUIG_AHU101_Position_2_EAF_Enable_D16_479	SwitchingDevice	controlsProperty
core:NUIG_AHU101_Position_Ctrls_MCC02__Fire_Alarm_D29_476	Alarm	offers measuresProperty
core:NUIG_AHU101_Position_Ctrls_MCC02__Gas_Alarm	Alarm	offers measuresProperty
core:NUIG_AHU101_Position_Extract_Bag_Filter_Differential_Pressure_Switch	SwitchingDevice	
core:NUIG_AHU101_Position_Supply_Bag_Filter_Differential_Pressure_Switch	SwitchingDevice	
core:NUIG_AHU101_Position_Supply_Panel_Filter_Differential_Pressure_Switch	SwitchingDevice	
core:NUIG_AHU101_Return_Duct	Duct_Segment	contains

core:NUIG_AHU101_Section_After_Cooling_Coil	Duct_Segment	contains
core:NUIG_AHU101_Section_After_Pre_Heating_Coil	Duct_Segment	contains
core:NUIG_AHU101_Service_Air_Quality_CO2_Control	Service	isOfferedBy
core:NUIG_AHU101_Service_Filter_Monitoring	Service	isOfferedBy
core:NUIG_AHU101_Service_Frost_Monitoring	Service	isOfferedBy
core:NUIG_AHU101_Service_General_Alarms	Service	isOfferedBy
core:NUIG_AHU101_Service_Relative_Humidity_Control	Service	isOfferedBy
core:NUIG_AHU101_Service_Temperature_Control	Service	isOfferedBy
core:NUIG_AHU101_Service_Thermistor_Trip_Monitoring	Service	isOfferedBy
core:NUIG_AHU101_Supply_Air_Outlet	Duct_Segment	contains
core:NUIG_Alice_Perry_Building_Location	geo:SpatialThing	geo:longitude geo:latitude
core:NUIG_Engineering_Building	Building	hasSpace geo:location
core:NUIG_G047_Lecture_Theater_3_Engineering_Building	BuildingSpace	contains isContainedIn hasSpace isSpaceOf
core:NUIG_G047_Lecture_Theater_3_Room_147_AHU101_Ctrls	UnitaryControlElement	contains accomplishes consistOf isSpaceOf
core:NUIG_G047_Lecture_Theater_3_Room_147_AHU101_Ctrls_Air_Quality_Controls	Controller	consistOf accomplishes
core:NUIG_G047_Lecture_Theater_3_Room_147_AHU101_Ctrls_General_Alarms	Controller	consistOf accomplishes
core:NUIG_G047_Lecture_Theater_3_Room_147_AHU101_Ctrls_Humidity_Controls	Controller	consistOf accomplishes
core:NUIG_G047_Lecture_Theater_3_Room_147_AHU101_Ctrls_Temperature_Settings	Controller	consistOf accomplishes
core:NUIG_GF_East_Rads_Energy_Acc_Total	Energy	
core:NUIG_Ground_Floor_Engineering_Building	BuildingSpace	contains isContainedIn hasSpace
core:PartsPerMillion	SmokeUnit	
core:Percentage	OccupancyUnit	
core:PercentageRH	HumidityUnit	

Apéndice D

Instalación del entorno

Instalación del entorno para el desarrollo de la ontología

Para poder desarrollar la ontología es necesario descargar el software [Protégé](https://protege.stanford.edu/products.php) de su página web oficial <https://protege.stanford.edu/products.php>, podemos acceder a su última versión 5.5.0, lanzada el 14 de marzo de 2019 para Windows directamente en el botón de descarga, no es necesario registrarse en la ventana emergente que aparece, con darle a declinar se inicia la descarga del software en automático. En cualquier caso, los desarrolladores dan tres opciones de uso: vía web, previo registro, y con aplicación desarrollada en Java. Esta aplicación requiere de JVM³⁶ pero por defecto se baja junto con el propio entorno un encapsulado que podemos utilizar sin modificar las versiones JRE³⁷ de nuestro equipo.

Una vez descargado el archivo comprimido **Protege-5.5.0-win.zip** podemos descomprimirlo en el lugar de nuestra elección, se genera una carpeta llamada **“Protege-5.5.0”**, en cuyo interior tenemos un árbol de carpetas y archivos con la aplicación. Para ejecutarla con su propio entorno java, ejecutaremos el archivo de comandos “run.bat” que creará el entorno y abrirá la aplicación por primera vez. En su primera carga aparecerá una ventana emergente para la instalación de plugin, la descartaremos en el botón **“Not Now”**. Podemos evitar que salte en cada reinicio mediante el *check* habilitado para ello. El aspecto final debe ser el mostrado en la Ilustración 86.

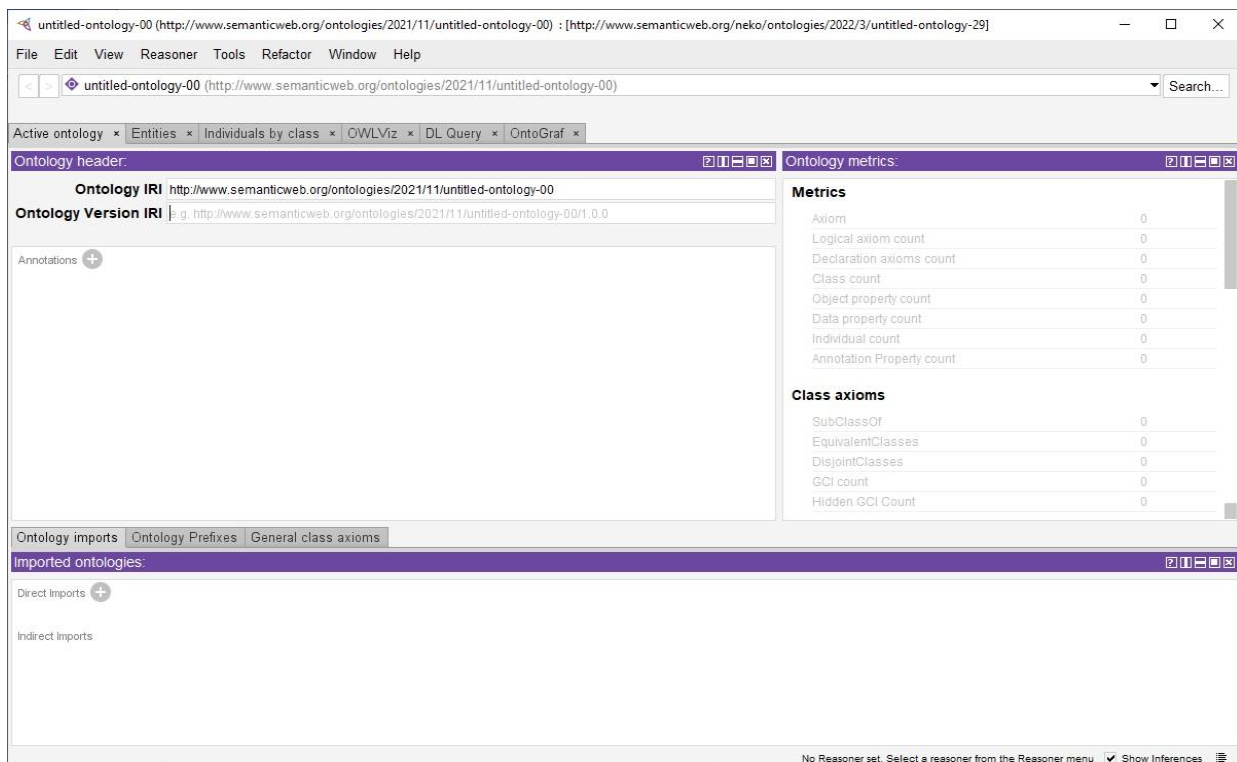


Ilustración 86. Pantalla inicial de Protégé

³⁶ JVM: Java Virtual Machine, https://es.wikipedia.org/wiki/M%C3%A1quina_virtual_Java

³⁷ JRE: Java Runtime Environment, <https://www.ibm.com/es-es/cloud/learn/jre>

Instalación del entorno de programación y pruebas

Para instalar el entorno de programación es necesario tener Java instalado, se puede hacer desde su web <https://www.oracle.com/java/technologies/downloads/> . Posteriormente hay que descargar el software [Anaconda](https://www.anaconda.com/products/distribution) de su página web oficial <https://www.anaconda.com/products/distribution>, accediendo a su última versión Anaconda3 2021.11(64-bit) directamente para Windows en el botón habilitado, que además indica que incluye Python 3.9. El software se descargará automáticamente desde el link descargando el archivo de autoinstalación “**Anaconda3-2021.11-Windows-x86_64.exe**” en el equipo. Una vez descargado lo ejecutamos y seguimos el proceso que se puede ver en la Ilustración 87 pulsamos sobre “**Next >**” para aceptar el proceso de instalación, a continuación, los términos de la licencia en “**I Agree**”. En la siguiente ventana aceptaremos la instalación solo para nuestro usuario “**Just Me (recommended)**” y continuamos en “**Next >**”, para elegir el directorio de instalación, se dejará el que figura por defecto y se dará a “**Next >**”. Nos preguntará sobre el registro en el PATH de Windows y sobre declarar el software como editor por defecto para Python, marcamos ambas casillas y damos a “**Install**”. Una vez completada la instalación, confirmaremos “**Next >**” dos veces, desmarcamos el inicio de los tutoriales y finalizamos el proceso de instalación de Anaconda.

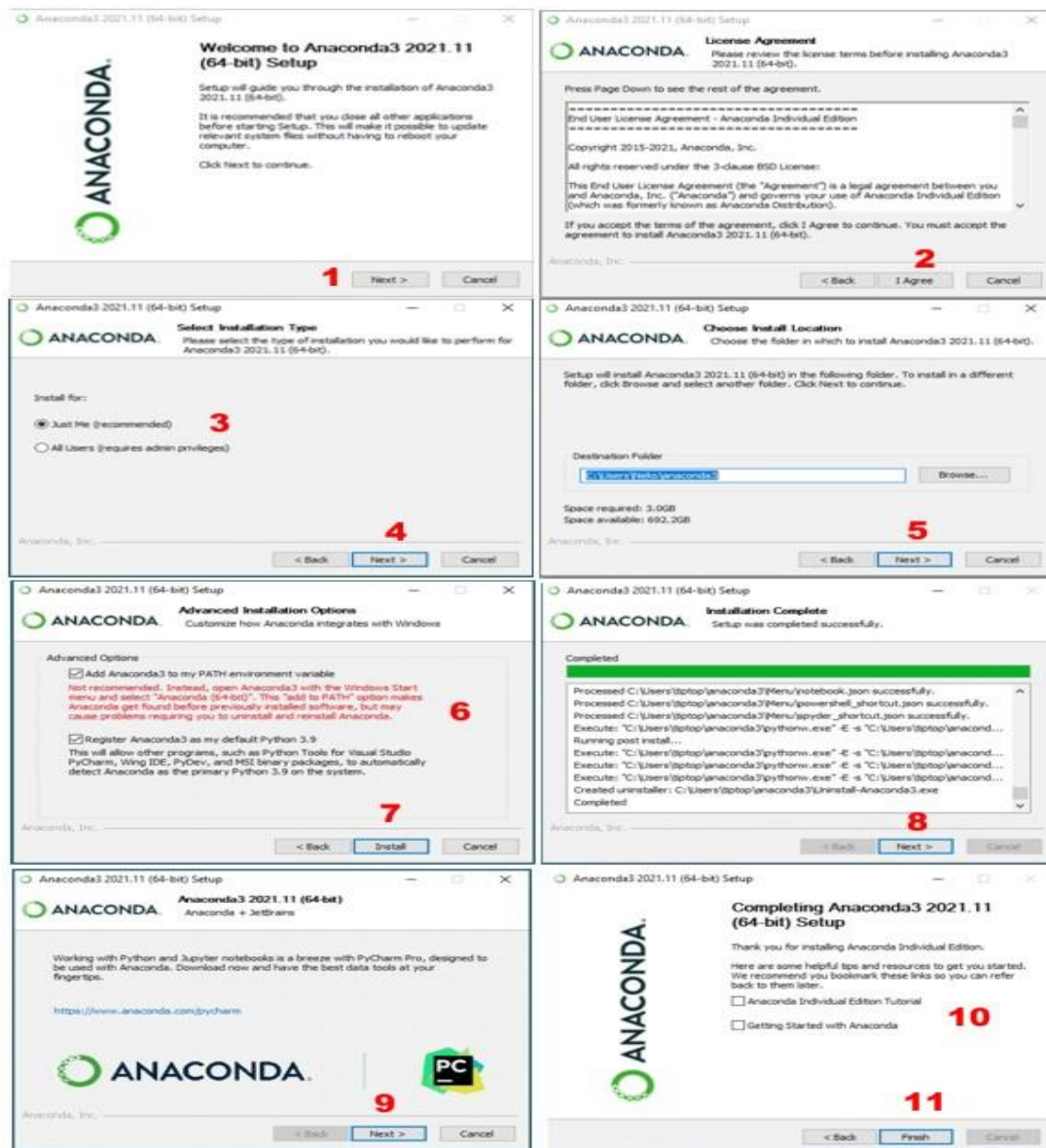


Ilustración 87. Pasos para la instalación de Anaconda

El siguiente paso es configurar el entorno de Python y crear el notebook para poder implementar la aplicación. Buscamos en el menú de inicio, en los programas instalados “**Anaconda3 (64-bit)**” y dentro pulsamos sobre “**Jupyter Notebook**”. Tras aparecer una ventana de comandos tipo MS-DOS que ejecuta el servidor de la aplicación, se abrirá en nuestro navegador la interfaz del programa.

La interfaz de Jupyter Notebook tiene el aspecto de la Ilustración 88 donde se listan los directorios disponibles en el punto de instalación elegido para la aplicación, pero nunca directorios de orden superior.

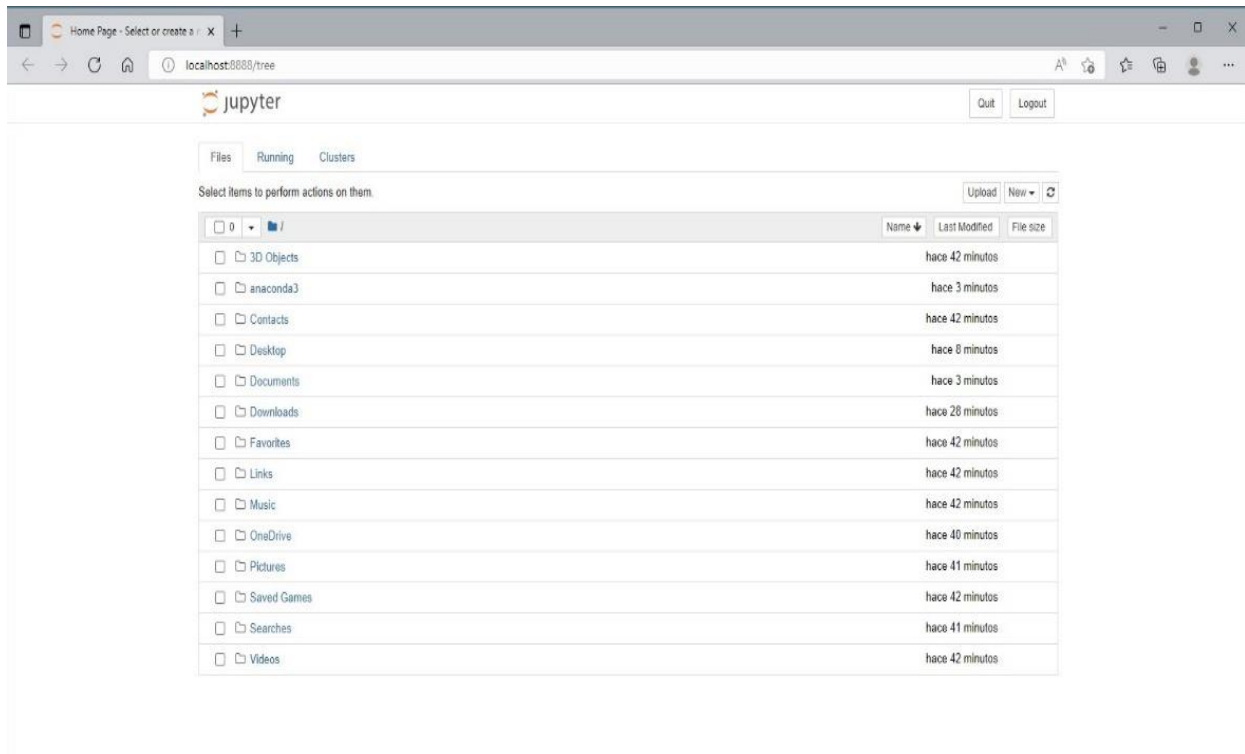


Ilustración 88. Interfaz Jupyter Notebook

Crearemos un directorio para desarrollar nuestro *notebook* y contener todos los archivos utilizados en el proyecto, y un *notebook* de trabajo. Para ello desde la interfaz, en la esquina superior derecha, pulsamos sobre “**New**” y después “**Folder**”, entraremos en el nuevo directorio creado y repetiremos “**New**” y después “**Python 3 (ipykernel)**” para crear el archivo, Ilustración 89.

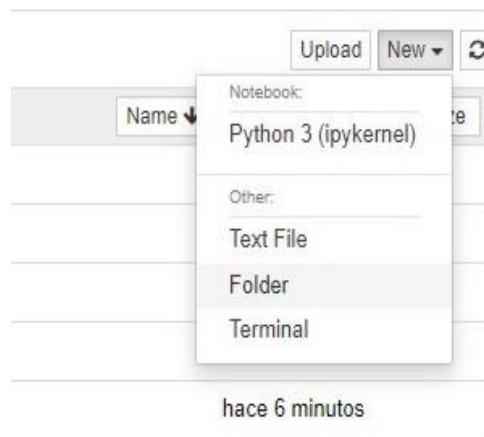


Ilustración 89. Jupyter Notebook: crear directorio y notebook

Para cambiar los nombres del directorio y el archivo, debemos seleccionarlos en el check de su izquierda y se habilitarán unos botones en la parte superior izquierda del menú para “**Rename**”, “**Move**”, entre otros, y por último el icono de una papelera que es “**Delete**”. Al pulsar sobre “**Rename**” nos dejará modificar el nombre con una pantalla emergente. El directorio lo nombraremos “**TFG**” y el notebook “**proyecto_angnava**”.

Por último, solo queda cargar los paquetes de Python que necesitamos para desarrollar el proyecto. Lo haremos desde el notebook, una vez abierto, escribiendo los siguientes comandos sobre el cuadrado que se habilita en la ventana del navegador, que en su borde superior izquierdo figura “**In []**”, de *input* o entrada:

```
!pip install pyspark
!pip install owlready2
```

Una vez escrito, con el cuadrado seleccionado, pulsaremos al botón superior “**Run**” y esperaremos a que se carguen las bibliotecas. Observaremos que a la izquierda se modifica el input por “**In [*]**”, el asterisco indica ejecución. Cuando finalice se le asignará un número de entrada, en este caso lo normal sería “**In [1]**”. Procederemos a comentar todo el bloque para que no sobrecargue futuras ejecuciones ya que este proceso solamente lo necesitamos una vez. Para ello comentamos todo el bloque comenzándolo con tres comillas, o en el desplegable superior donde figura “**Code**” lo pasamos a “**Raw NBConvert**”.

El paso siguiente es importar las bibliotecas necesarias para la ejecución del programa. Sobre el siguiente cuadro escribimos y ejecutamos la carga de las mismas:

```
### Carga de bibliotecas ###
import matplotlib.pyplot as plt
import os.path
from ipywidgets import widgets, Layout
from IPython.display import display
from tkinter import Tk, filedialog
from pyspark.sql import SparkSession
from pyspark.sql.types import *
from pyspark.sql.functions import *
import json
from owlready2 import *
```