MODELO ONTOLÓGICO DE DEPURACIÓN PARA ESPECIFICACIONES DE CASOS DE USO

MOD-CAS



TESIS PARA OPTAR AL TÍTULO DE:

ESPECIALISTA EN INGENIERÍA DE SOFTWARE

PRESENTA:

DIEGO ALEJANDRO GRAJALES RODRÍGUEZ

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

BOGOTÁ, D.C. 2016

MODELO ONTOLÓGICO DE DEPURACIÓN PARA ESPECIFICACIONES DE CASOS DE USO

MOD-CAS



TESIS PARA OPTAR AL TÍTULO DE:

ESPECIALISTA EN INGENIERÍA DE SOFTWARE

PRESENTA:

DIEGO ALEJANDRO GRAJALES RODRÍGUEZ

DIRECTOR: PHD. SANDRO BOLAÑOS

REVISOR: MAGISTER J.J. MEZA

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

BOGOTÁ, D.C. 2016

Dedicatoría

A mis padres, familia, compañeros y en especial a mi tio Raphael Grajales que siempre me impulsa a seguir estudiando …

Agradecimientos

- Agradecimiento especial a la Especialización en Ingeniería de Software de la Universidad Distrital Francisco Jose De Caldas por haberme dado la oportunidad de realizar este postgrado.

- A los profesores que compartieron su conocimiento conmigo y con mis compañeros.

- Al ingeniero JJ Meza por sus valiosos aportes como revisor en el planteamiento y el desarrollo de la investigación.

- Al ingeniero Sandro Bolaños por haberme apoyado en la idea de realizar esta investigación y haberme guiado durante todo el proyecto.

Nota de aceptación

_____

_____

_____

_____

_____

Firma del Director del Proyecto

_____

Firma del Revisor del Proyecto

Bogotá, 17 de Enero de 2016

Índice general

Índice de figuras **13**

1. Proyecto **1**

1.1. Formulación del Problema 1

1.1.1. Sistematización del Problema 1

1.2. Objetivo General 2

1.3. Objetivos Específicos 2

1.4. Hipótesis 2

1.5. Justificación 3

1.6. Metodología 4

1.6.1. Tipo de Estudio 4

1.6.2. Método de Investigación 5

1.6.3. Fuentes y Técnicas para la recolección de Información 5

1.6.4. Tratamiento de la información 5

2. Marco de Referencia **7**

2.1. Marco Teórico 7

2.1.1. Arquitectura Empresarial 7

2.1.2. Errores En Una Especificación De Caso De Uso . . . 9

2.1.3. Diseño de Ontologías 12

2.1.4. Propuesta para diseño de ontologías 14

2.1.5. Pasos del proceso de desarrollo de ontologías 14

2.1.6. Límites del alcance de la ontología 17
2.1.7. Trabajos relacionados con Ontologías y análisis de requerimientos 17
2.1.8. Frameworks y Aplicaciones para el trabajo con Ontologías 18
2.2. Marco Conceptual 20
2.2.1. Framework de Arquitectura Empresarial TOGAF . . 20
2.2.2. Lenguaje Archimate 22
2.2.3. Casos De Uso Y Especificaciones De Casos De Uso 23
2.2.4. Visión Formal De Un Caso De Uso 24
2.2.5. Ontologías 25
2.2.6. Lenguaje de Ontologías Web (OWL) 25
2.2.7. Origen de OWL 26

3. Definición de la Arquitectura Empresarial 29
3.1. Contextualización del proyecto 29
3.2. Advanced Software Tools 30
3.2.1. Misión 30
3.2.2. Visión 30
3.2.3. Principios 30
3.2.4. Estructura Organizacional 32
3.2.5. Productos 32

4. Capa de Negocio 33
4.1. Definición 33
4.2. Punto de Vista de Organización 33
4.3. Punto de Vista de Cooperación de Actor 35
4.4. Punto de Vista de Función de Negocio 36
4.5. Punto de Vista de Proceso de Negocio 38
4.6. Punto de Vista de Cooperación de Proceso de Negocio . . . 40

4.7. Punto de Vista de Producto 41

5. Capa de Aplicaciones 43

5.1. Definición 43
5.2. Punto de Vista de Comportamiento de Aplicación 43
5.3. Punto de Vista de Cooperación de Aplicación 45
5.4. Punto de Vista de Estructura de Aplicación 46
5.5. Punto de Vista de Uso de Aplicación 47

6. Nivel de Infraestructura 49

6.1. Definición 49
6.2. Punto de Vista de Infraestructura 49
6.3. Uso de Infraestructura 51
6.4. Organización e Implementación 52
6.5. Punto de Vista de Estructura de Información 54
6.6. Punto de Vista de Realización del Servicio 56
6.7. Punto de Vista de Capas 57

7. Motivación 59

7.1. Definición 59
7.2. Punto de Vista de Stakeholder 59
7.3. Punto de Vista de Realización de Objetivos 60
7.4. Punto de Vista de Contribución 62
7.5. Punto de Vista de Principios 63
7.6. Punto de Vista de Realización de Requerimientos 64
7.7. Punto de Vista de Motivación 66
7.8. Punto de vista de Proyecto 67
7.9. Punto de Vista de Migración 68
7.10. Punto de Vista de Migración e Implementación 69

8. Arquitectura de bajo nivel **71**
8.1. Contexto 71
8.2. Patrones Creacionales 72
8.2.1. Método Fabrica 72
8.2.2. Prototipo 73
8.3. Patrones Estructurales 74
8.3.1. Puente 74
8.3.2. Componente 76
8.3.3. Fachada 77
8.3.4. Proxy 78
8.4. Patrones de Comportamiento 79
8.4.1. Comando 79
8.4.2. Mediador 80
8.4.3. Visitador 81

9. Ontología MOD-CAS **83**
9.1. Dominio y alcance de la ontología MOD-CAS 83
9.2. Términos clave de la ontología MOD-CAS 85
9.3. Definición de las clases y jerarquía de clases de la ontología MOD-CAS 86
9.3.1. Jerarquía de Clases: 87
9.4. Definición de las propiedades y relaciones de la ontología MOD-CAS 89
9.5. Modelos de la Ontología MOD-CAS para identificar los errores propuestos 91
9.5.1. Tamaño del caso de uso 91
9.5.2. La complejidad de la estructura del caso de uso . . . 92
9.5.3. Inconsistencias de lenguaje 94
9.5.4. Descomposición del caso de uso 94
9.5.5. Elementos faltantes 95

9.5.6. Flujos incorrectos 96
9.5.7. Ausencia de pares adyacentes 96
9.5.8. Errores de actores no mencionados 97
9.5.9. Inconsistencias en numeración de los pasos 97
9.5.10. La condición para acceder a un flujo alternativo no es clara 99
9.6. Análisis de Casos de Uso usando la Ontología MOD-CAS . 100
9.6.1. Tamaño del caso de uso 100
9.6.2. La complejidad de la estructura del caso de uso . . . 101
9.6.3. Inconsistencias de lenguaje 104
9.6.4. Descomposición del caso de uso 106
9.6.5. Elementos faltantes 107
9.6.6. Flujos incorrectos 108
9.6.7. Ausencia de pares adyacentes 111
9.6.8. Errores de actores no mencionados 112
9.6.9. Inconsistencias en numeración de los pasos 115
9.6.10. La condición para acceder a un flujo alternativo no es clara 115

10. Conclusiones **117**
10.1. Conclusiones 117
10.2. Trabajos futuros 118

A. Anexos **119**
A.1. Caso de Uso de Prueba 119
A.1.1. Caso de Uso Número CU001: 119
A.2. Ontología MOD-CAS 121

Bibliografía **137**

Índice de figuras

2.1. Lenguajes sobre los que se construye OWL[Bec15] 26

3.1. Organigrama 32

4.1. Metamodelo Vista de Organización 33
4.2. Vista de Organización 34
4.3. Metamodelo Vista de Cooperación de Actor 35
4.4. Cooperación de Actor 35
4.5. Metamodelo Punto de Vista de Función de Negocio 36
4.6. Función de Negocio 1 37
4.7. Función de Negocio 2 37
4.8. Metamodelo Vista Proceso de Negocio 38
4.9. Proceso de Negocio 39
4.10. Metamodelo Punto de Vista Proceso de Negocio 40
4.11. Proceso de Negocio 40
4.12. Metamodelo Vista de Producto 41
4.13. Vista de Producto 42

5.1. Metamodelo Comportamiento de Aplicación 43
5.2. Comportamiento de Aplicación 44
5.3. Cooperación de Aplicación 45
5.4. Cooperación de Aplicación 45
5.5. Metamodelo Estructura de Aplicación 46
5.6. Estructura de Aplicación 46

5.7. Metamodelo Uso de Aplicación 47
5.8. Uso de Aplicación 48

6.1. Metamodelo de Infraestructura 49
6.2. Infraestructura 50
6.3. Metamodelo de Uso de Infraestructura 51
6.4. Uso de Infraestructura 51
6.5. Metamodelo de Organización e Implementación 52
6.6. Organización e Implementación 53
6.7. Metamodelo de Estructura de Información 54
6.8. Estructura de Información 54
6.9. Metamodelo de Realización del Servicio 56
6.10. Realización del Servicio 56
6.11. Vista de Capas 57

7.1. Metamodelo Vista de Stakeholder 59
7.2. Vista de Stakeholder 60
7.3. Metamodelo Vista de Realización de Objetivos. Fuente [Archimate 2.0][Diseñado con Coloso] 60
7.4. Vista de Realización de Objetivos. Fuente: [Autor] 61
7.5. Metamodelo de Vista de Contribución 62
7.6. Vista de Contribución 62
7.7. Metamodelo de Vista de Principios 63
7.8. Vista de Principios 63
7.9. Metamodelo de Realización de Requerimientos 64
7.10. Realización de Requerimientos 64
7.11. Vista de Contribución 65
7.12. Metamodelo de Vista de Motivación 66
7.13. Vista de Motivación 66
7.14. Metamodelo de Vista de Proyecto 67

7.15. Vista de Proyecto 67
7.16. Metamodelo de Vista de Migración 68
7.17. Vista de Migración 68
7.18. Metamodelo de Vista de Migración e Implementación 69
7.19. Vista de Migración e Implementación 69

8.1. Estructura de la aplicación MOD-CAS 71
8.2. Metamodelo de Método Fabrica 72
8.3. Método Fabrica 72
8.4. Metamodelo de Prototipo 73
8.5. Prototipo 73
8.6. Metamodelo de Puente 74
8.7. Puente Ontología 74
8.8. Puente Razonador 75
8.9. Puente Analizador Semántico 75
8.10. Metamodelo de Componente 76
8.11. Componente 76
8.12. Metamodelo de Fachada 77
8.13. Fachada 77
8.14. Metamodelo de Proxy 78
8.15. Proxy 78
8.16. Metamodelo de Comando 79
8.17. Comando 79
8.18. Metamodelo de Mediador 80
8.19. Mediador 80
8.20. Metamodelo de Visitador 81
8.21. Visitador 82

9.1. Herencia de clases de la ontología MOD-CAS 87
9.2. Pasos del escenario básico del CU001 Fuente: [Autor] 100

9.3. Total de Pasos del escenario básico del CU001. Fuente: [Autor] 101
9.4. Pasos con una sola acción en el CU001. Fuente: [Autor] . . . 102
9.5. Pasos con mas de una acción en el CU001. Fuente: [Autor] . 102
9.6. Cantidad de pasos con una sola acción en el CU001 Fuente: [Autor] 103
9.7. Cantidad de pasos con mas de una acción en el CU001. Fuente: [Autor] 103
9.8. Paso 6 del CU001 modelado en la ontolgía MOD-CAS Fuente: [Autor] 104
9.9. Resultado de la composición de los pasos del caso de uso CU001. Fuente: [Autor] 105
9.10. Número de pasos de cada escenario alternativo del caso de uso CU001. Fuente: [Autor] 106
9.11. Precondiciones del caso de uso CU001. Fuente: [Autor] . . . 107
9.12. Poscondiciones del caso de uso CU001. Fuente: [Autor] . . . 107
9.13. Recursos utilizados en el caso de uso CU001. Fuente: [Autor] 108
9.14. Dependencia entre pasos. Fuente: [Autor] 109
9.15. Prerequisitos para alcanzar un paso. Fuente: [Autor] 109
9.16. Prerequisitos para alcanzar un paso del escenario alternativo 4a. Fuente: [Autor] 110
9.17. Actores que ejecutan cada paso del flujo básico. Fuente: [Autor] 111
9.18. Secuencia de pasos indicando actores. Fuente: [Autor] . . . 112
9.19. Pasos ejecutados por el Actor: El Comprador. Fuente: [Autor] 113
9.20. Consulta de sujetos que ejecutan acciones en el caso de uso. Fuente: [Autor] 114
9.21. Pasos con el mismo número. Fuente: [Autor] 115
9.22. Modelado de una condición para un escenario alternativo. Fuente: [Autor] 116

9.23. Recursos y Estados relacionados con los escenarios alternativos. Fuente: [Autor] 116

Capítulo 1

Proyecto

1.1. Formulación del Problema

¿Modelar una ontología de dominio que conceptualice las especificaciones de casos de uso con el fin de eliminar ambigüedades y errores de dicha especificación conseguirá mejorar la descripción del requerimiento funcional descrito por el caso de uso?

1.1.1. Sistematización del Problema

Para responder a la formulación del problema se deben plantear los siguientes cuestionamientos respecto a las variables enunciadas:

1. ¿Cómo modelar de forma genérica una ontología que represente las especificaciones de casos de uso?
2. ¿Cómo lograr que la ontología identifique las ambigüedades y errores que una especificación de caso de uso tiene?
3. ¿Cómo lograr que la ontología propuesta desambigüe y elimine errores de la especificación de caso de uso que se encontraron en el análisis?
4. ¿Cómo verificar que la ontología propuesta aporto efectivamente información a la especificación del caso de uso?

1.2. Objetivo General

Modelar una ontología de dominio que conceptualice especificaciones de casos de uso en lenguaje OWL y que tenga como finalidad desambigüar las especificaciones que se analicen usando esta ontología para mejorar la descripción del requerimiento que se encuentre plasmado en el caso de uso.

1.3. Objetivos Específicos

1. Modelar una ontología en lenguaje OWL que represente de forma genérica una especificación de casos de uso.

2. Diseñar los mecanismos que le permitan a la ontología identificar los errores y las ambigüedades de la especificación de caso de uso que se esta analizando.

3. Diseñar los mecanismos que le permitan a la ontología propuesta aportar información minimizando los errores y las ambigüedades presentes en la especificación del caso de uso.

4. Verificar que la ontología propuesta y los mecanismos propuestos en la ontología efectivamente logran aportar información al caso de uso.

1.4. Hipótesis

Mejorar la calidad de las especificaciones de casos de uso en un procesos de desarrollo de Software requiere de una ontología que permita eliminar las ambigüedades y errores en los casos de uso, logrando mejorar la calidad de los requerimientos especificados.

1.5. Justificación

La motivación de la investigación surge de una problemática existente en el levantamiento de requerimientos que es uno de los aspectos más importantes de los ciclos de vida del desarrollo de sistemas. Específicamente en proyectos en los que se utilicen casos de uso como herramienta para describir requerimientos.

Las especificaciones de casos de uso son una sucesión de interacciones entre los usuarios y el sistema, que deben tener como objetivo el alcance de un resultado o estado del sistema deseado por los usuarios.

Los casos de uso generalmente son escritos en un documento de texto que plantea secciones como flujos básicos, tablas de entradas, flujos alternativos, reglas de negocio etc.

A diferencia de muchos otros aspectos de diseño para sistemas de información como diagramas, prototipos, flujos, los Casos de Uso no poseen una herramienta con la que se puedan diseñar, en general se escriben usando lo un editor de texto.

Como todos los demás elementos de diseño de un sistema un Caso de Uso posee una descripción formal de sus objetivos dentro del desarrollo del sistema es decir una forma correcta de realizarse. Para muchos otros elementos de diseño las herramientas CASE facilitan la forma de escribirlos y evitan cometer errores que estén en contra de estas definiciones formales. Los editores de texto no permiten hacer este tipo de controles sobre un Caso de Uso, no están pensados para hacerlo.

Esta ausencia de una herramienta para la especificación de Casos de Uso deriva en recurrentes errores en la escritura de los mismos, lo que conlleva a materializar riesgos en múltiples etapas posteriores en las que este documento de caso de uso es insumo para tareas como codificación, diseño y ejecución de pruebas, mantenimiento etc.

Abordar la idea de realizar validaciones en un tema tan variable en su forma

requiere de una tecnología que sea flexible pero permita mirar el problema desde su generalidad.
Por esta razón se pensó en abordar la complejidad desde una tecnología de sistemas expertos, pues el modelo propuesto debería aportar de forma automática validaciones dentro de un dominio especifico del lenguaje de un sistema que está definido con casos de uso y estas características las puede aportar el diseño de un modelo ontológico de dominio usando un lenguaje amplio y rico para estas definiciones como lo es el lenguaje Web Ontology Language (OWL-DL).
La investigación aportaría un método práctico para validar especificaciones de caso de uso permitiendo identificar ambigüedades y errores de forma y contenido contribuyendo a mejorar la calidad y consistencia de estos artefactos y en general la de los proyectos de software.

1.6. Metodología

1.6.1. Tipo de Estudio

La investigación busca encontrar un método práctico para evaluar la calidad de las especificaciones de caso de uso. Partiendo de un problema claramente identificado que son los errores y las ambigüedades que se encuentran en las especificaciones de casos de uso, buscar la forma de identificarlos automáticamente, creando una herramienta para este fin.
El trabajo de investigación deberá estar en capacidad de estudiar y comprender las variables que componen el problema que son: errores y ambigüedades de especificaciones de casos de uso y tecnologías y técnicas que permitan identificar los diferentes errores de especificaciones de casos de uso.
Las características anteriormente enunciadas para la investigación circunscriben el trabajo en un tipo de estudio explicativo, que desea responder a una hipótesis causal compuesta de múltiples variables.

1.6.2. Método de Investigación

Con las variables que componen la hipótesis de investigación se busca crear una herramienta que identifique errores pero que sea capaz de responder en forma general a cualquier especificación de casos de uso y no a un caso de uso o tipo de caso de uso. De modo que se partirá de evidencias y conceptos específicos, concretos y teóricos para llegar al desarrollo de un modelo general para validación de casos de uso. La investigación se enmarca en un método inductivo para llegar al conocimiento necesario y cumplir el objetivo.
El trabajo de investigación debe seguir una metodología de desarrollo iterativo e incremental que permite realizar mayor control del avance del proyecto y aporta flexibilidad en el trabajo al poder realizar refinamientos de elementos de etapas anteriores en cada iteración ejecutada.

1.6.3. Fuentes y Técnicas para la recolección de Información

Dado que se conceptualizara el problema desde trabajos teóricos relacionados, las fuentes desde donde se recolectara información serán publicaciones en revistas, tesis, textos, manuales, la Web, entre otros. No se requiere realizar entrevistas o encuestas, las fuentes primarias de información relacionadas con la observación del problema se harán sobre especificaciones de casos de uso de ejemplos encontradas en la bibliografía propuesta.

1.6.4. Tratamiento de la información

Con el fin de validar los resultados de la investigación se deben realizar pruebas sobre las variables identificadas en el desarrollo y de este modo lograr una validez de los modelos propuestos. Se debe evaluar:

1. La capacidad que tiene el modelo ontológico para encontrar errores y ambigüedades en los casos de uso que se analicen.

2. La capacidad que tiene el modelo ontológico para aportar consistencia en los casos de uso evaluados.

Capítulo 2

Marco de Referencia

A continuación se presenta el marco teórico con los temas que permiten plantear la investigación y el marco conceptual que permite delimitar los conceptos y las variables de la investigación.

2.1. Marco Teórico

2.1.1. Arquitectura Empresarial

La arquitectura empresarial no es una arquitectura enfocada al diseño de sistemas de información unicamente, si bien los sistemas de información son importantes dentro de las organizaciones para lograr el exito y alcanzar objetivos es necesaria una vision mas amplia que incluya las estrategias de negocio y aspectos de la organización. Sobre todo alinear el uso de la tecnología con el negocio es uno de los principales intereses de los gerentes y administradores de TI.

La arquitectura empresarial cubre requerimientos y estrategias empresariales asi como procesos de negocio, aplicaciones e infraestructura de información y comunicación buscando una optima articulación entre todas estas diferentes facetas.[PD14]

Los esfuerzos de una arquitectura empresarial deben tener un enfoque de top-down antes que diseñar sistemas infraestructuras y comunicaciones. Sin

una visión holistica del negocio claramente articulada como manifestación del comportamiento del negocio cualquier desarrollo de sistemas de información tendera a suplir los requerimientos actuales de la organización pero fallaran al alinearse y cumplir con los requerimientos estratégicos del negocio.

Una arquitectura de forma general es usada para describir la utilidad de un conjunto de estructuras mostrando las relaciones entre todos estos elementos para todos los interesados.[PD14]
Una arquitectura empresarial se usa para describir las utilidades de un conjunto de estructuras empresariales para todos los interesados dentro de la organización mostrando las relaciones entre estas estructuras que pueden ser: arquitecturas de negocio, arquitecturas de información, arquitecturas de sistemas de información, arquitecturas de datos y arquitectura de la infraestructura de tecnología.
La clave para la excelencia de una organización esta en aprovechar apropiadamente las tendencias tecnológicas. Las compañías que tienen las mejores formas de identificar estas tendencias y pueden establecer sus estrategias de acuerdo con estas tendencias se comienzan a llamar Organizaciones Digitales y les permitirá sobrevivir, superar a sus competidores y mejorar continuamente.[Mil15]
Las tecnologías actuales y futuras le permiten a las compañías crear ventajas competitivas. El éxito de la compañía depende de que tan temprano se adopten esas tecnologías y la estrategia que usen para esta adopción. Las compañías deben ser capaces de transformarse digitalmente, esta transformación responde a cambios sociales para establecer una cultura organizacional de agilidad, innovación y empoderamiento.[AU15]
Dentro del top diez de las tendencias tecnológicas del portal de Garnet para el año 2016 se habla de la capacidad de obtener información de todos los medios posibles. La informacion siempre ha existido pero de forma aislada, incompleta, no disponible o inentendible. Los avances en herramientas semánticas

como bases de datos orientadas a grafos entre otras tecnologías emergentes de clasificación de datos y análisis de información brindaran significado a la gran cantidad de caótica información.[Woo16]

2.1.2. Errores En Una Especificación De Caso De Uso

Definir cuáles son los errores que se pueden cometer cuando se está especificando un caso de uso es el primer paso para diseñar la manera de identificarlos de forma automática. Un estudio empírico tomo una muestra de 360 ECU, identifico y clasifico los errores encontrados proponiendo una taxonomía de errores de ECU[PSS14]. La taxonomía definida en este estudio se compone de las siguientes categorías:

- **La baja orientación a los interesados:** Error que se puede medir observando si el caso de uso está escrito en un nivel técnico y no de negocio, si el caso de uso está escrito orientado al desarrollo y si el objetivo a alcanzar por el caso de uso es ambiguo.
- **Tamaño del caso de uso:** Se puede medir contando la cantidad de pasos que un caso de uso presenta en el escenario principal o normal y en el número de reglas de negocio propuestas en el caso de uso
- **La complejidad de la estructura del caso de uso:** Se puede medir en términos de qué tan complejas son las acciones atómicas, o las acciones que agrupan acciones atómicas, donde cada paso del caso de uso debería representar una sola acción y que tantas condiciones o decisiones contiene cada uno de estas acciones.
- **Inconsistencias de lenguaje:** La categoría de inconsistencias habla del uso de pronombres en la redacción de un caso de uso (el, ella, nosotros...) y de mezclar el nivel de abstracción de la ECU hablando en algunos pasos del negocio y en otros de los aspectos técnicos del sistema que se desea modelar.

- **Mala redacción o nivel pobre de redacción:** Se consideran como elementos de una mala redacción el uso de la voz pasiva, que es una figura gramatical en la que se puede entender que el sujeto no tiene responsabilidad en la acción, por el contrario una forma más asertiva de escribir una sentencia sería el uso de la figura de voz activa en la que el sujeto si se hace responsable de la acción y el uso de oraciones negativas en contraposición al uso de oraciones positivas.
- **Número de escenarios alternativos:** El número de escenarios alternativos a considerar debe ser el necesario, añadir un gran número de escenarios alternativos esperando abarcar toda la complejidad de un negocio puede resultar en más complejidad innecesaria.
- **La falta de precondiciones y postcondiciones:** La falta de precondiciones y pos condiciones no son requeridas en todos los casos de uso, pero son recomendadas.
- **Ausencia de pares adyacentes:** Los pares adyacentes fueron propuestos por Phalp et al. La idea es que cada interacción propuesta en el sistema tenga una respuesta.
- **Errores de actores no mencionados:** Los errores de actores no mencionados corresponden a que en la ECU se presente una gran cantidad de pasos sin que el actor sea mencionado.

Un estudio adicional realizado sobre 43 casos de uso especificados para una empresa de construcción y venta de automóviles revela otros tipos de errores adicionales a los enunciados en la taxonomía del punto anterior. El estudio demostró que la mayoría de los errores en las especificaciones corresponden a elementos faltantes y errores o ambigüedades lingüísticas. El estudio propone 12 tipos de errores clasificados en 7 categorías.[TIPO06]

1. **Completitud**:

- Elementos faltantes: Faltan elementos de la especificación como precondiciones, postcondiciones, actores, título etc.
- No se evidencia cumplir el objetivo del CU: El flujo principal de la especificación del caso de uso no logra el objetivo del caso de uso.

2. **Exactitud**:

- Flujos incorrectos: Hay errores en los pasos que se referencian en los flujos alternativos ó los flujos alternativos retornan a pasos incorrectos en el flujo principal.
- Acciones fuera del dominio del sistema: ¿Las acciones que ejecuta el caso de uso están relacionadas con el objetivo o con el dominio del problema?

3. **Consistencia**:

- Inconsistencias en numeración de los pasos: ¿La numeración de todos los pasos es consistente?
- Pasos o acciones irrelevantes: ¿Hay un paso por cada acción? ¿Todos los pasos son relevantes para conseguir el objetivo?
- Descomposición del caso de uso: ¿Hay flujos alternativos tan extensos que debieron modelarse como un caso de uso diferente?

4. **Legibilidad**:

- Mal uso de flujos alternativos: Se abusa de la cantidad de flujos alternativos para representar flujos normales muy complejos o extensos.

5. **Ambigüedad**:

- La condición para acceder a un flujo alternativo no es clara: ¿La condición para acceder al flujo alternativo está claramente especificada y en el lugar correcto?
- Errores lingüísticos: ¿Existen sinónimos, homónimos, pronombres, y referencias que son usadas innecesariamente?

6. **Nivel de detalle**:

   - El nivel de detalle del caso de uso es muy específico en cada paso o por el contrario es muy general

7. **Mal uso de las Precondiciones**:

   - Las precondiciones son enunciadas pero dentro de los pasos de la especificación son contradichas.

Ambos trabajos definen claramente una serie de errores que se pueden evidenciar en especificaciones de casos de uso. Los errores expuestos definen un marco de trabajo de las validaciones que se pueden generar en el desarrollo de la investigación al momento de analizar la especificación por medio del modelo ontológico a desarrollar.

2.1.3. Diseño de Ontologías

Las ontologías son importantes porque permiten a expertos en dominios expresar información de su campo de conocimiento como por ejemplo: Ontologías para categorizar contenidos de sitios web como Yahoo, ontologías para categorizar servicios y productos de Amazon.com, ontologías en el campo de la medicina como “snomed” y la red semántica “Unified Medical Language System”. Las ontologías definen un vocabulario común para investigadores que necesitan compartir información de un dominio.[NFN05] Las principales Razones para desarrollar ontologías son:

- Compartir conocimiento como una estructura de información entre personas o entre agentes de software: Las ontologías permiten extraer información a agentes de software, si varios sitios web relacionados usan una misma ontología podrían compartir de manera más fácil sus contenidos. Esta es una de las principales razones para desarrollar ontologías.

- Permitir la reutilización de conocimiento: Un grupo de investigadores podría reutilizar una ontología ya desarrollada y aplicarla a un dominio en el que se necesite. Las ontologías se pueden integrar a otras ontologías y se pueden extender (completar a partir de una ontología base).

- Explicar suposiciones de un dominio: Las ontologías permiten cambiar suposiciones de un dominio de conocimiento fácilmente si el conocimiento cambia. Las explicaciones del dominio de conocimiento son útiles para personas que deseen acercarse al significado de términos de un dominio.

- Separar conocimiento de un dominio del conocimiento operacional: Una ontología puede por ejemplo describir un procedimiento para configurar un producto y con esta ontología se puede implementar un programa que ejecute la configuración de un producto, luego dependiendo de cómo se alimente la ontología se pueden configurar diferentes productos, computadores, carros etc.

- Analizar el conocimiento de un dominio: El desarrollo de una ontología no es la meta. Diferentes métodos, aplicaciones y agentes de software pueden hacer uso de las ontologías como datos. Una aplicación podría crear sugerencias a partir de las estructuras definidas como ontología sobre un tema en específico.

2.1.4. Propuesta para diseño de ontologías

En la práctica desarrollar una ontología consta de:

1. Definir las clases de la ontología.
2. Organizar las clases en jerarquía taxonómica (subclases y super clases)
3. Definir los slots y describir los valores permitidos para los slots.
4. Llenar los valores de los slots para las instancias.

Aunque no existe una metodología correcta para desarrollar ontologías se puede abordar con un enfoque iterativo e incremental que pretende en cada iteración refinar la definición de la ontología propuesta completándola cada vez de forma más detallada pero siguiendo las reglas fundamentales presentadas a continuación[NFN05]:

1. No existe una forma correcta de modelar un dominio, diferentes ontologías pueden expresar un domino de forma correcta.
2. El desarrollo de una ontología debe ser iterativo.
3. Los conceptos que se definan en la ontología deben ser cercanos a los objetos modelados y las relaciones entre estos objetos en el dominio que se está modelando. Se puede definir los objetos como sustantivos y las relaciones como verbos en oraciones que definen el dominio.

2.1.5. Pasos del proceso de desarrollo de ontologías

La guía en cada iteración del desarrollo de la ontología es decidir para que se va a usar la ontología y que tan detallada o general se requiere si olvidar que la ontología debe representar un concepto del mundo real[NFN05]. El método propuesto consta de 7 pasos descritos a continuación:

1. Determinar el alcance de la ontología
   Se deben responder a las preguntas: ¿Cuál es el dominio de la ontología?, ¿Para qué se va a usar la ontología?, ¿Para qué tipo de preguntas la ontología deberá proveer respuestas?, ¿Quién usara y mantendrá la ontología? Una de las formas de limitar el alcance es realizar la lista de preguntas más completa posible que la ontología deberá responder, dichas preguntas servirán al final de proceso como pruebas de calidad de la ontología propuesta.

2. Considerar la reutilización de ontologías existentes.
   Las ontologías permiten ser reutilizadas, importando una ontología ya existente o extendiéndola en caso de ser necesario, las ventajas de reutilizar es que se pueden encontrar modelos de ontologías ya depurados que se adapten a las necesidades del dominio que se requiere expresar[NFN05]:

   En Internet se encuentran varios recursos con bibliotecas de ontologías como:

   Ontolingua: http://www.ksl.stanford.edu/software/ontolingua/

   DAML Ontology Library: http://www.daml.org/ontologies/

   Ontologías comerciales públicas: http://www.unspsc.org/,
   https://www.rosettanet.org/,
   http://www.dmoz.org/,
   http://webprotege.stanford.edu/

3. Enumerar los términos importantes para la ontología
   Se debe enumerar los términos importantes para un experto en el dominio y las características de cada uno de estos términos, independientemente de que se traten de clases, propiedades o individuos.

4. Definir las clases y la jerarquía de las clases: Se pueden usar 3 estrategias para definir las clases y su jerarquía para la ontología:

   - Top-down: Identificar los conceptos o clases más generales y posteriormente identificar las sub clases o conceptos más especializados.
   - Bottom-up: Comenzar con la definición de las clases más especializadas que representaran las hojas de la ontología y agrupar estas clases en conceptos más generales.
   - Proceso Combinado: Consiste en combinar las dos técnicas anteriores, identificando clases específicas, clases generales y encontrando clases intermedias que las relacionan jerárquicamente.

5. Definir las propiedades de las clases (slots):
   Consiste en describir la estructura interna de los conceptos identificados en el paso anterior, los conceptos necesitan de estas características para poder comenzar a responder a las preguntas que guían el diseño de la ontología.

6. Definir las facetas de los slots:
   Las facetas de los slots (propiedades de las clases) definen el dominio, el rango y tipos de datos del slot. Las facetas más comunes son:

   - Cardinalidad: Define cuantos valores puede tener el slot, en algunos lenguajes se puede definir cardinalidad mínima y máxima.
   - Tipos de Valores: Pueden ser Cadenas de caracteres, numéricos, boléanos, Enumeraciones, e Instance que admite la definición de relaciones entre individuos.
   - Dominios y Rangos: Las clases admitidas para los slot de tipo Instance son llamados rango de un slot. Las clases cuyas propiedades son descritas por un slot son llamadas dominio del slot.

7. Crear Instancias: En el último paso se crean las instancias individuales (individuos) de la ontología. Crear una instancia consiste en elegir una clase, crear un individuo y asignar valores a los slots de la clase a la que pertenece el individuo creado.

2.1.6. Límites del alcance de la ontología

Una ontología no debería tener toda la información posible de un dominio pero si la necesaria para la aplicación que se le quiere dar. Tampoco es necesario que contenga todos los posibles slots de las clases propuestas en la ontología y de igual manera no es necesario definir todas las relaciones existentes entre clases o individuos, ya que pueden aparecer datos irrelevantes para el problema que se desea modelar[NFN05].

2.1.7. Trabajos relacionados con Ontologías y análisis de requerimientos

Existen diferentes trabajos con enfoques en el análisis de requerimientos por medio de ontologías, sin embargo no se encontraron trabajos enfocados a mejorar la consistencia de especificaciones de casos de uso. Entre los trabajos encontrados cabe destacar: “Ontology Based Requirements Analysis:Lightweight Semantic Processing Approach”[KS05] ó “Análisis de requerimientos basados en ontologías: Aproximación de procesamiento semántico ligero” cuyo objetivo general es establecer un mapeo entre una lista de requerimientos de un sistema y un modelo ontológico de dominio que represente componentes semánticos de los requerimientos analizados. La investigación permitió realizar 3 tipos de análisis sobre la lista de requerimientos de un sistema:

1. Detectar inconsistencias.

2. Medir la calidad de los requerimientos especificados.

3. Predecir cambios en los requerimientos especificados.

Otro trabajo destacado es “Deriving Requirements Model from Textual Use Cases”[SRP+14] ó “Derivando modelos de requerimientos de especificaciones de casos de uso”, en este trabajo los autores buscaron por medio de análisis lingüísticos de las especificaciones de casos de uso crear dos modelos diferentes: Un diagrama en lenguaje BPEL (Business Process Execution Language) y una ontología de dominio OWL que represente el requerimiento especificado en el caso de uso. La investigación les permitió generar los diagramas BEPL y las ontologías perdiendo entre el 2 % y el 3 % de las acciones descritas en el texto del caso de uso. Esta investigación no tenía como objetivo verificar la consistencia ni aumentar la consistencia de la especificación analizada.

2.1.8. Frameworks y Aplicaciones para el trabajo con Ontologías

Existe un gran número de herramientas que permiten construir y hacer uso de las características de los modelos ontológicos. A continuación se enumeran algunos de los más relevantes y sus características.

- Framework de creación de ontologías: Protégé
  Protégé es una aplicación libre de código abierto mantenida por la comunidad compuesta de un conjunto de herramientas para construir modelos de dominio y aplicaciones expertas basadas en ontologías. Protégé está soportado por la comunidad académica, gobiernos y corporaciones privadas que lo usan para construir bases de conocimiento de diversas áreas como biomedicina, e-commerce y modelamiento organizacional. Protégé puede ser adaptado como un componente para construir modelos simples y complejos de ontologías. Y puede adaptarse para trabajar con otros componentes como motores de inferencia o razonadores[fBIR15]. Las principales características de Protégé son:

- Cumple con los estándares del W3C
- Interfaz de usuario personalizable
- Soporte para visualización gráfica de la ontología
- Refactorización de ontologías
- Interfase para integración con razonadores.
- Arquitectura adaptable con otras aplicaciones.
- Compatibilidad con la versión web de Protégé.

- Sistemas de búsquedas e inferencias: Pellete
  Pellete es un razonador opensource desarrollado por Mindswap. Es un razonador exclusivo para OWL DL e implementado en java, dispone de una API para poder ser utilizado directamente desde una aplicación Java[Llo07]. Ofrece una variedad de tareas de razonamiento y compatibilidad con otras herramientas de razonamiento como por ejemplo JENA.

2.2. Marco Conceptual

2.2.1. Framework de Arquitectura Empresarial TOGAF

The Open Group Architectural Framework TOGAF fue desarrollado por The Open Group (de ahí TOG) y su primera versión fue liberada en 1995 y su versión mas actual es la 9.1. TOGAF provee una visión pragmática de la arquitectura empresarial resaltando la centralización en el rol de la organización. Cualquier transformación de arquitectura requiere una cercana colaboración entre las diferentes personas involucradas en el desarrollo de la arquitectura deseada. El gobierno de la organización, los interesados del proyecto, administradores y el equipo que va a construir la arquitectura se incluyen en muchos de los temas tratados en el framework TOGAF.[PD14]
La colaboración entre estos actores y temas esta basada en un proceso organizado. El proceso es llamado ADM (Architecture Development Method) que provee una estructura para el desarrollo de proyectos de transformación de arquitectura donde la comunicación juega un rol esencial en cada uno de los estados del proceso ya que debe existir un común entendimiento de los objetivos de la arquitectura. Los medios usados para realizar esta comunicación efectiva pueden ser documentos, modelos etc. y deben ser claramente definidos y adaptados para cada uno de los participantes.[PD14]
El ADM se compone de 8 paso nombrados por letras de la A a la H comenzando por la determinación de los objetivo y estrategias para implementar la arquitectura hasta el mantenimiento y despliegue de la arquitectura construida en la ultima fase[PD14]. A continuación se enuncian cada una de las fases:

- **Fase A Visión:** En esta fase se establece el alcance de la arquitectura, la visión de la arquitectura y se identifican los interesados.
- **Fase B Arquitectura de Negocio:** Se identifica una línea base de la arquitectura de negocio y se propone una arquitectura de negocio que

se desea alcanzar. Se identifican las brechas entre la arquitectura base encontrada y el modelo propuesto. Se evalúa el impacto de la implementación y se propone una hoja de ruta de implementación.

- **Fase C Arquitectura de sistemas de información:** Se identifica una linea base de la arquitectura de sistemas de información y datos y se propone una arquitectura de sistemas de información que se desea alcanzar. Se identifican las brechas entre la arquitectura base encontrada y el modelo propuesto. Se evalua el impacto de la implementación y se propone una hoja de ruta de implementación.

- **Fase D Arquitectura de Tecnología:** Se identifica una linea base de la arquitectura de Tecnología y se propone una arquitectura de Tecnología que se desea alcanzar. Se identifican las brechas entre la arquitectura base encontrada y el modelo propuesto. Se evalua el impacto de la implementación y se propone una hoja de ruta de implementación.

- **Fase E Oportunidades y Soluciones:** Se propone la planeación y medios para realizar las entregas de los bloques de construcción identificados en las fases anteriores.

- **Fase F Planificación de la Migración:** Se establece una agenda de migración precisa y se constituyen los proyectos y su organización teniendo en cuenta objetivos y costos.

- **Fase G Gobierno de la Implementación:** Se establece la versión definitiva de la arquitectura y los contratos que soportan los proyectos. Se asegura que los proyecto de implementación se desarrollen de acuerdo a la arquitectura diseñada.

- **Fase H Gestión de Cambios de la Arquitectura:** Se realiza un proceso continuo de evaluación de cambios en la arquitectura propuesta.

2.2.2. Lenguaje Archimate

Una arquitectura es tipicamente desarrollada porque existen personas que tienen algun interes o necesidad que debe ser solucionada por sistemas de información en la organización. El rol del arquitecto es identificar estos intereses y refinar los requerimientos de los interesados desarrollando vistas de arquitectura que muestren como esos intereses van a ser suplidos.[PD14]

Una arquitectura es una descripción formal de un sistema de información organizada de forma que soporte la representación de la estructura y el comportamiento de las propiedades del sistema y su evolución.[PD14]

Para proveer una representación uniforme de diagramas que describan una arquitectura empresarial The Open Group desarrollo el lenguaje ArchiMate que ofrece una aproximación integrada para describir y visualizar los diferentes dominios de una arquitectura asi como sus relaciones y dependencias.

ArchiMate define tres capas principales cada una identificada con un color diferente y actualmente en la versión 2.0 se agrego una capa de motivación [Gro13]:

1. **Capa de Negocio:** Ofrece los productos y servicios que son realizados por los procesos y los actores de la organización para los clientes externos.

2. **Capa de Aplicación:** Presenta el soporte que debe tener la capa de Negocio con los servicios de aplicaciones que deben ser realizados por artefactos de software.

3. **Capa de Tecnología:** Ofrece los servicios de infraestructura necesarios para ejecutar las aplicaciones de la capa de Aplicación, realizados por computadores, hardware de comunicación etc.

4. **Capa de Motivación:** La extensión de motivación adiciona conceptos como objetivos, principios, requerimientos e interesados.

2.2.3. Casos De Uso Y Especificaciones De Casos De Uso

Los casos de uso son una herramienta ampliamente difundida para plasmar las necesidades o requerimientos de los usuarios a la hora de desarrollar un sistema[HPH11, KS05]. Existen muchas formas, plantillas o recomendaciones que definen cuales son los componentes de una especificación de un caso de uso. A diferencia de otros componentes de diseño de sistemas como diagramas UML, prototipos, pruebas etc. que usan herramientas específicamente construidas para su representación y que permiten realizar una validación de la forma que deben tener estos artefactos, la especificación de los casos de uso no cuenta con una herramienta propia ampliamente difundida o utilizada y a causa de esto generalmente se usa un editor de texto para su realización. Adicionalmente una de las ventajas de las especificaciones de casos de uso es que tienen la libertad de expresar las necesidades de los interesados en lenguaje natural. Las definiciones de los casos de uso pueden ser realizadas por personas que no tengan un conocimiento técnico sobre informática o programación ya que pueden expresar lo que necesitan del sistema pero no como debe ser diseñado y desarrollado. El lenguaje natural manejado en los casos de uso permite plasmar mucha de la complejidad del negocio que se está especificando de forma muy simple, sin embargo, esta ventaja también es origen de errores y ambigüedades que de no ser detectadas pueden causar problemas en posteriores etapas del ciclo de vida del sistema[PSS14, SRP$^+$14]. Para poder definir como identificar estos errores y ambigüedades en los casos de uso primero se debe analizar formalmente que es un caso de uso y cuales errores se presentan al momento de escribir un caso de uso. Estas definiciones despejan el camino para el diseño de un sistema de conocimiento sobre casos de uso que se convierta en un marco de referencia para la construcción o validación de cualquier caso de uso sea cual sea el negocio que se esté representando. En adelante el artículo se referirá a la especificación de un caso de uso de forma general como (E.C.U.)

2.2.4. Visión Formal De Un Caso De Uso

El concepto de caso de uso fue introducido por Ivar Jacobson en 1992. Un caso de uso debe representar la interacción entre el sistema que se está modelando y un actor ya sea una persona u otro sistema que esperan conseguir un objetivo que les aporte valor. Un caso de uso debe especificar un comportamiento por medio de una secuencia de pasos y variantes de esta secuencia principal. Los casos de uso (C.U.) deben representar el comportamiento del sistema pero no deben indicar como debe ser implementado[JR98].
Todo CU debe tener un nombre único para ser identificado y que exprese su idea principal, una serie de actores que van a interactuar con el sistema para obtener un resultado y una secuencia de pasos que identifican el inicio y el final de la interacción con los actores[SRP$^+$14].
Sin embargo estas definiciones de las partes de un caso de uso esconden muchos más conceptos detrás de sí y cada una se puede desglosar en sus componentes para realizar un análisis más completo.
En la Figura[CR98] se muestra un diagrama de clases UML definiendo un CU en el que se pueden encontrar Escenarios normales y alternativos, estos escenarios se componen de acciones que se pueden clasificar como acciones concurrentes, secuenciales, iterativas y alternativas. Las acciones pueden contener más acciones o ser de tipo atómico. Los escenarios alternativos son causados porque en el CU se presentan flujos de condición que generan un camino diferente al flujo normal. Finalmente toda acción atómica puede necesitar de parámetros representados como objetos y va a tener influencia desde o hacia un actor o un recurso como una base de datos, archivo o periférico en un computador.

2.2.5. Ontologías

El concepto de Ontología aplica en muchas áreas de conocimiento no solo para la ingeniería, en esta investigación se trabajara el concepto de Ontología para las áreas de sistemas expertos e inteligencia artificial. El concepto de Ontología fue propuesto por T.R. Gruber en 1993 y se define como "Una especificación explicita y formal sobre una conceptualización compartida"[CG11]. Una Ontología es un sistema de representación de conocimiento que es compartida y valida[MT14]. Una ontología debe tener una representación formal que le permita a un proceso automático ó software su interpretación y uso. Una ontología es una descripción explicita y formal de conceptos en un dominio de discurso a lo que se llama Clases, propiedades de cada concepto describiendo varias características y atributos del concepto (slots) llamados roles, propiedades o atributos, y restricciones sobre los slots: facetas también llamados restricciones de rol.
El conjunto de las Clases y los individuos y las relaciones entre ellos constituyen una base de conocimiento. Se puede decir que no es claro el punto para diferencia la ontología de la base de conocimiento que esta describe.[NFN05] Existen varios tipos de ontologías como ontologías de aplicación, ontologías de tarea, ontologías de alto nivel y ontologías de dominio. La ontología propuesta para la investigación es una ontología de dominio. La representación formal de una Ontología debe hacerse a través de un lenguaje estándar definido para el tipo de ontología que se está desarrollando.

2.2.6. Lenguaje de Ontologías Web (OWL)

El Lenguaje de Ontologías Web (OWL) está diseñado para ser usado en aplicaciones que necesitan procesar el contenido de la información en lugar de únicamente representar información para los humanos[Su5]. OWL es un

lenguaje de diseño de ontologías para la web semántica y está pensado para ser compatible con estándares web[Bec15].

2.2.7. Origen de OWL

OWL nace de la necesidad de permitir a las maquinas procesar automáticamente e integrar la información disponible en la web[Su5]. Esto corresponde a la aparición de la web semántica como la siguiente generación de tecnologías web después de los documentos HTML y de la segunda generación de las aplicaciones web dinámicas[Bec15].
La Web semántica se basará en la capacidad de XML para definir esquemas de etiquetas a medida y en la aproximación flexible de RDF para representar datos. El primer nivel requerido por encima de RDF para la Web semántica es un lenguaje de ontologías que pueda describir formalmente el significado de la terminología usada en los documentos Web. Si se espera que las máquinas hagan tareas útiles de razonamiento sobre estos documentos, se debía crear un lenguaje más completo que RDF, a esta necesidad responde OWL[Su5].

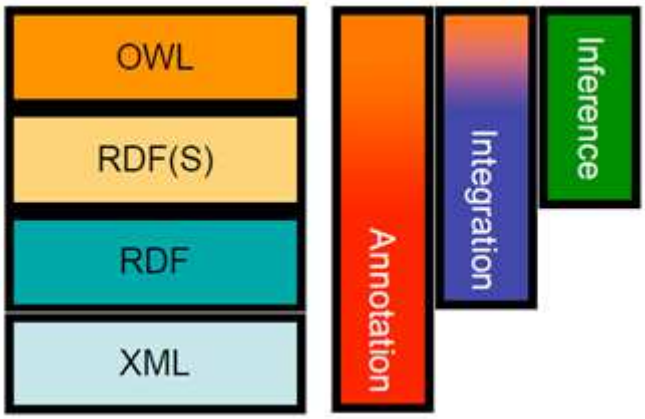


Figura 2.1 Lenguajes sobre los que se construye OWL[Bec15]

El World Wide Web Consortium (W3C) define 3 lenguajes estándar para representación de ontologías a saber: RDF, RDFS y OWL basados en XML[Bec15, Su5].

XML: eXtensible Markup Language en la web semántica proporciona una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas en el significado de estos documentos.
Resource Description Framework RDF: Es un lenguaje orientado a representar información de recursos en la Web. RDF permite que la información de la ontología sea procesada por aplicaciones y no solo pueda ser entendida por personas. RDF representa la información en XML y ordenado en tripletas que se componen de un sujeto, un predicado y un objeto. Aunque fue diseñado para representar semántica de sitios web se puede representar otras áreas de conocimiento.
RDF Schema: Es un vocabulario utilizado para describir propiedades y clases de recursos RDF, con una semántica para la generalización y jerarquización tanto de propiedades como de clases.
Web Ontology Language OWL: Es un lenguaje que extiende el RDF Figura 2, también representa la información en XML y puede ser interpretado por software. RDF tiene debilidades a la hora de describir detalles de los recursos web, describir rangos y dominios de restricciones, establecer cardinalidad entre los conceptos relacionados entre otras limitaciones, OWL da soluciona a estas limitaciones.

Existen tres subtipos de lenguaje OWL:

- OWL Lite es el lenguaje más básico y poco usado soportando restricciones simples y solo permite expresar cardinalidad de 0 a 1.

- OWL DL permite realizar razonamientos, no permite definir clases como una instancia, permite tener una semántica bien definida en la ontología y tiene una relación directa con un campo llamado Lógica Descriptiva (de ahí su nombre DL).

- OWL Full no impone restricciones en el uso de vocabulario que puede ser tan rico como lo permita RDF, a su vez permite expresar una clase como instancia entre otras capacidades, sin embargo no garantiza que todas las operaciones de razonamiento sobre la ontología puedan ser resueltas.[Bec15, Su5]

Capítulo 3

Definición de la Arquitectura Empresarial

3.1. Contextualización del proyecto

El objetivo general del proyecto es desarrollar un modelo ontológico que sea capaz de desambiguar las especificaciones de casos de uso, Sin embargo es útil para definir la arquitectura de un sistema que necesite o se beneficie del desarrollo de la ontología proponer un contexto empresarial que permita definir los requerimientos que debe satisfacer el desarrollo del proyecto.
El enfoque de arquitectura empresarial que se va a seguir es el propuesto por el Open Group en el framework Togaf Apoyándose en el lenguaje de definición de arquitectura Archimate.
A continuación se presentan una propuesta de empresa que se beneficia directamente del desarrollo de la ontología y por medio de puntos de vista cómo se logra definir un sistema que use la ontología a nivel comercial.

3.2. Advanced Software Tools

Advanced Software Tools es la idea de empresa propuesta que se puede beneficiar del desarrollo de una ontología para desambiguar especificaciones de casos de uso.

3.2.1. Misión

Advanced Software Tools busca proporcionar herramientas de software innovadoras construidas con los más altos estándares de calidad que apoyen el ciclo de vida de desarrollo de sistemas de información en compañías de construcción de software. Las herramientas de AST se enfocan en asegurar la calidad de los distintos artefactos que se construyen en las etapas de un proyecto de desarrollo de software, permitiéndoles a sus clientes cumplir los objetivos de sus proyectos.

3.2.2. Visión

AST desea que sus herramientas sean percibidas por sus clientes como un apoyo necesario e importante a la hora de llevar a cabo cualquier proyecto de ingeniería de software.

A mediano plazo los productos y servicios de AST se deben estar usando en las 10 más importantes casas de software del país.

3.2.3. Principios

1. Mantener un margen de ganancias que le permita a la compañía continuar con el área de investigación con el fin de innovar en sus productos y servicios.

2. Aplicar estándares de calidad en los productos.

3. La operación de la empresa se debe mantener incluso cuando hay interrupción en los sistemas.

4. Las aplicaciones desarrolladas o generadas deben ser usadas por toda la organización. Se debe evitar tener aplicaciones duplicadas o dearrollar aplicaciones similares que solo cubran areas pequeñas de la organización.

5. La Arquitectura de las aplicaciones debe ser basada en diseño de servicios que reflejen actividades del negocio reales que representen procesos del negocio.

6. AST siempre aplicara las mejores prácticas para asegurar la calidad de los productos.

7. AST realizara investigación continua para que los productos desarrollados sean innovadores.

8. Usar Software Libre con el fin de ahorrar costos de operación como licencias, así mismo se debe contribuir a la comunidad de software libre reportando los errores y contribuyendo en los foros de las herramientas utilizadas.

9. Orientar los esfuerzos a las necesidades de nuestros clientes. Crear servicios que aporten valor a la producción de software de forma que los clientes nos perciban como un aliado necesario en sus procesos.

10. La transparencia como fuente de confianza entre los clientes y la organización.

11. La propiedad intelectual debe ser protegida. La proteccion debe reflejarse en el diseño de la arquitectura de IT, su implementación y el gobierno de sus procesos.

12. La información es un activo de gran valor para la organización y en consecuencia debe ser manejado como tal. Los datos son protegidos de acceso no autorizado y revelacion no autorizada.

3.2.4. Estructura Organizacional

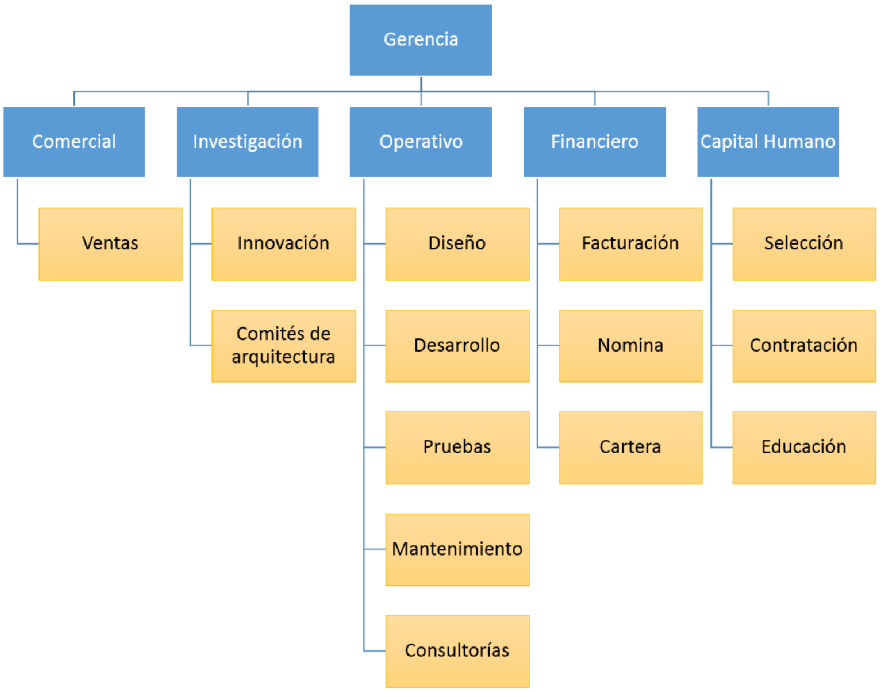


Figura 3.1 Organigrama

3.2.5. Productos

- **MOD-CAS**: MOD-CAS Es un modelo Ontológico de depuración de especificaciones de casos de uso, su objetivo es desambiguar las especificaciones de casos de uso aportando consitencia a dichas especificaciones.

- **Consultoría e Interventoría**: AST presta servicio de consultoría principalmente en levantamiento y validación de requerimientos.

Capítulo 4

Capa de Negocio

4.1. Definición

En la capa de negocio se presenta la estructura, los procesos y los interesados que interactúan en la organización.

4.2. Punto de Vista de Organización

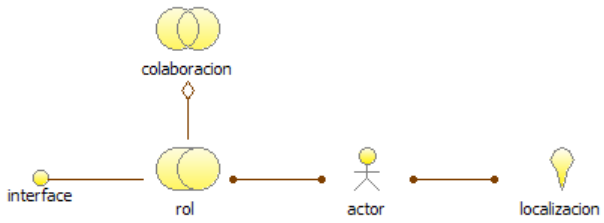


Figura 4.1 Metamodelo Vista de Organización

El punto de vista de organización presenta los principales elementos para comenzar a identificar la interación entre los interesados y la organización. En el metamodelo los actores se encuentran ubicados en una localización y se define cual es su rol y con cuales otros roles colaboran para cumplir sus funciones.

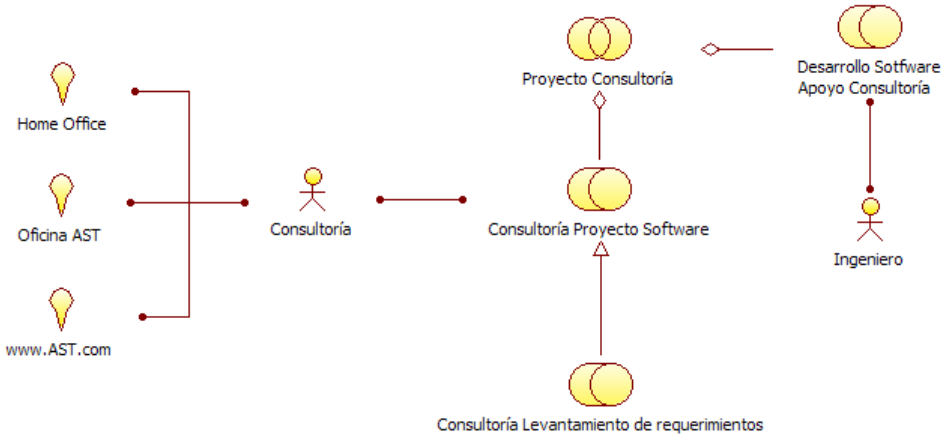


Figura 4.2 Vista de Organización

AST es una empresa que presta servicios de consultoría en levantamiento y evaluación de requerimientos. Los proyectos de consultoría son apoyados por el area de desarrollo de software que aporta herramientas tecnicas para hacer mas eficiente los procesos que se deben llevar a cabo en un proyecto.

AST desea ayudar a sus clientes a alcanzar las metas propuestas en los proyectos de desarrollo de software mejorando la calidad de los requerimientos con los que se pretende construir su sistema, del mismo modo AST opta por mejorar la calidad de vida de sus ingenieros y la calidad de vida de la ciudad en general permitiendo que las labores que no necesiten presencia en la oficina se ejecuten desde su hogar evitando traslados demorados y descongestionando la ciudad.

4.3. Punto de Vista de Cooperación de Actor

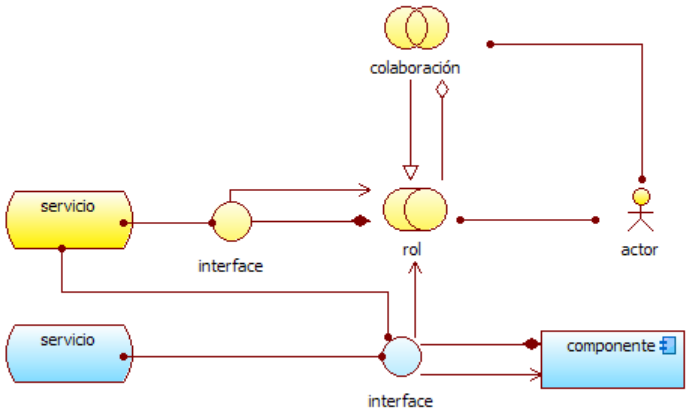


Figura 4.3 Metamodelo Vista de Cooperación de Actor

El punto de vista de cooperación de actor permite identificar con mayor precisión cuales son los servicios que dicho rol aporta a la organización y como este rol se relaciona por medio de una interfaz con estos servicios. El metamodelo también incluye como este servicio que un interesado realiza para la organización esta relacionado con un elemento de software que puede ser un servicio o un componente lógico.

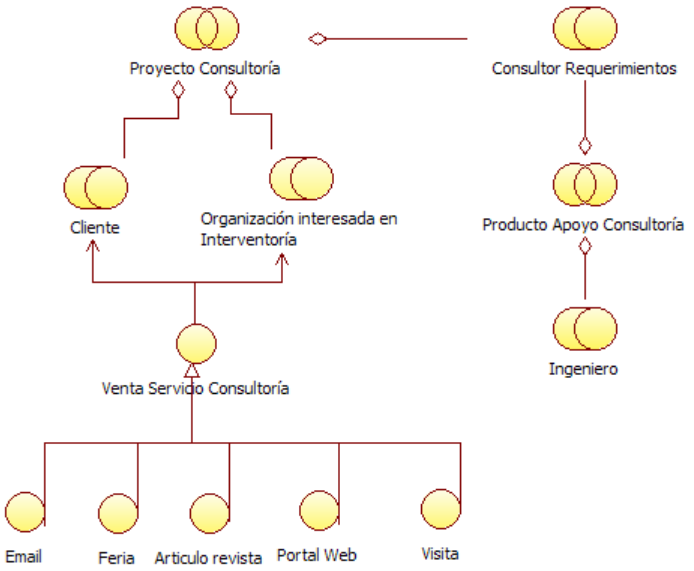


Figura 4.4 Cooperación de Actor

Los clientes de AST se van a relacionar con la organización por medio de la celebración y ejecución de un contrato de consultoría al cual va a estar asignado uno o varios ingenieros consultores que van a colaborar con los ingenieros de desarrollo y las herramientas internas desarrolladas para apoyo a las tareas de consultoría.

Previamente los agentes comerciales de AST han realizado un contacto con los clientes por medios electronicos o presenciales ofreciendo los servicios de consultoría especializada en requerimientos.

Los clientes pueden iniciar el contacto por medio electronico o por haber visitado los puestos de AST en ferias ó haber leido sobre los proyectos de AST en revistas especializadas de tecnología.

4.4. Punto de Vista de Función de Negocio

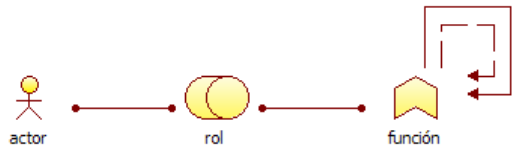


Figura 4.5 Metamodelo Punto de Vista de Función de Negocio

El punto de vista de función de negocio permite identificar cuales son las funciones de las que se encarga que un actor que esta jugando un determinado rol en la organización. Las funciones en este caso no son funciones de nivel de software sino funciones de su cargo que aportan a los objetivos de la organización.

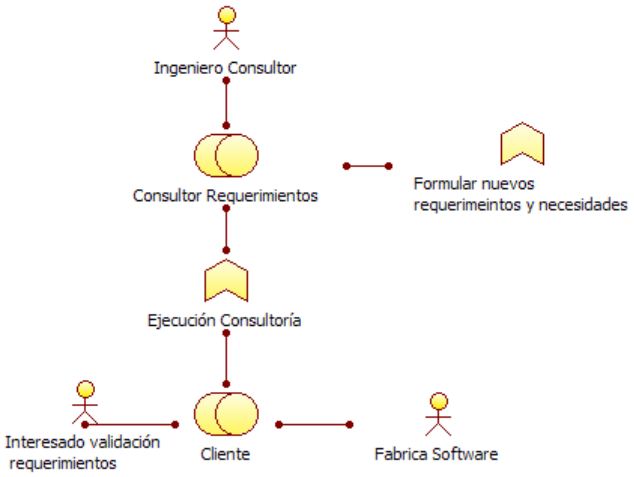


Figura 4.6 Función de Negocio 1

Un consultor de requerimientos se encarga de llevar a cabo la consultoría contratada por un cliente, en lo posible usando las herramientas internas para optimizar su trabajo. El consultor de requerimientos cumple otra función importante que es ayudar a plantear a la organización nuevas formas de mejorar la calidad y la eficacia de su trabajo en términos de ideas de software que serán transmitidas al arquitecto de software de la organización.

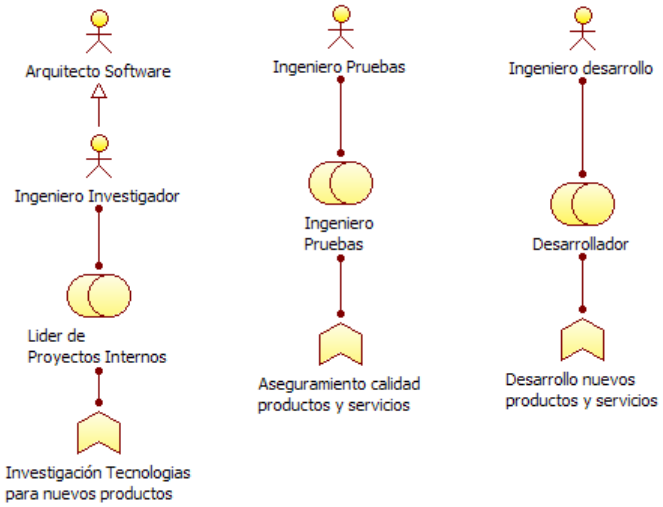


Figura 4.7 Función de Negocio 2

El arquitecto de software se encarga de recibir las propuestas ó ideas planteadas por el ingeniero de requerimientos y llevar a cabo el desarrollo de dichas propuestas. EL arquitecto debe estar en constante formación y entre sus funciones debe estar investigar nuevas tecnologías que permitan apoyar las tareas de consultoría.

Los ingenieros de desarrollo y pruebas se encargan de implementar las nuevas herramientas propuestas y diseñadas por el arquitecto y los consultores de requerimientos.

4.5. Punto de Vista de Proceso de Negocio

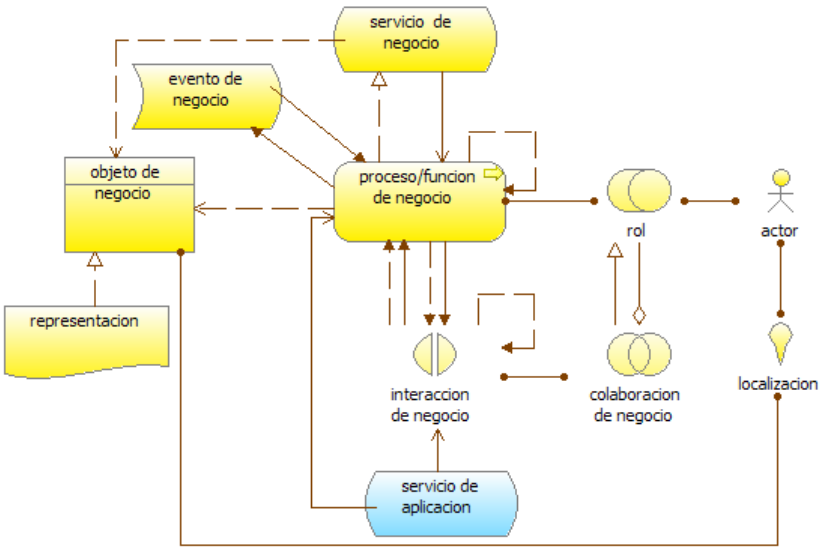


Figura 4.8 Metamodelo Vista Proceso de Negocio

El punto de vista de proceso de negocio es el punto de vista en el que se van a modelar los procesos de negocio de la organización que se esta modelando. El punto de vista introduce elementos importantes como el proceso que nos permite representar un conjunto de tareas de una organización, estos procesos de negocio pueden ser iniciados por un evento y estar asociados a un objeto de negocio que representa información a nivel de proceso de una organización.

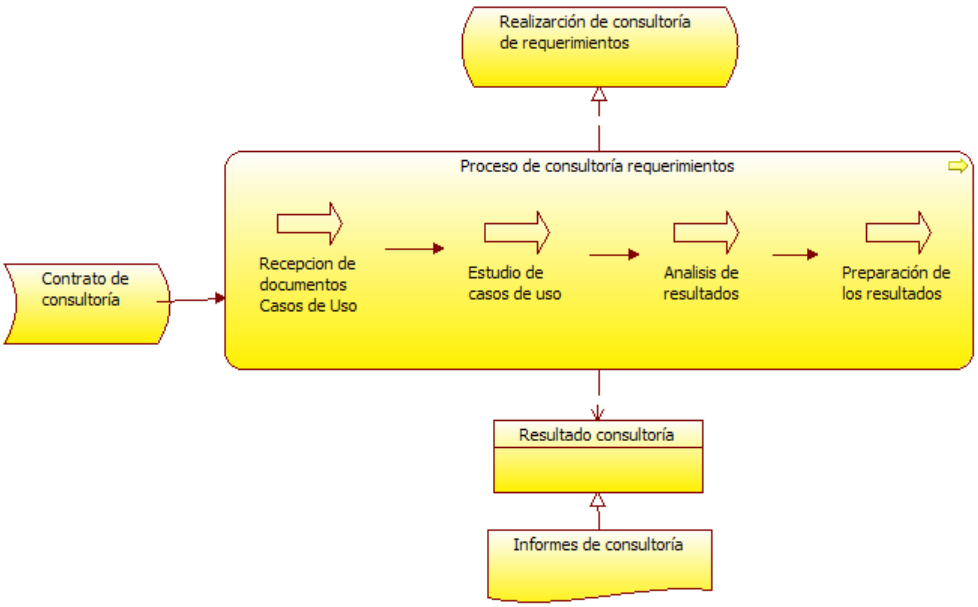


Figura 4.9 Proceso de Negocio

El proceso se inicia con el contrato de una consultoría que va a perseguir dar una opinión experta con respecto a unos requerimientos especificados en casos de uso. Al final de la consultoría se deben presentar las especificaciónes mejoradas al cliente.

Dentro del proceso de consultoría se reciben los documentos en los que el cliente ha realizado la especificación, Se normalizan los documentos en una plantilla propia de AST para realizar una mejora por medio del modelo MOD-CAS y se preparan los resultados con las nuevas especificaciones resultantes del proceso.

4.6. Punto de Vista de Cooperación de Proceso de Negocio

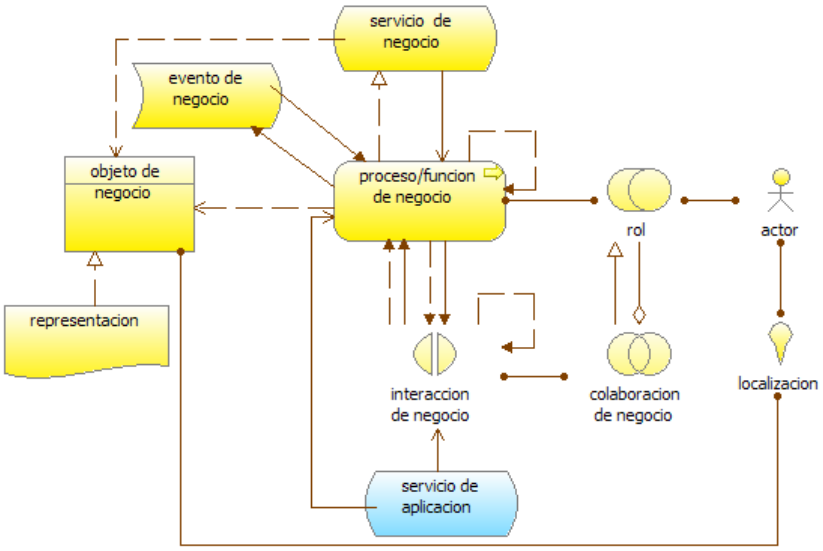


Figura 4.10 Metamodelo Punto de Vista Proceso de Negocio

Aunque el metamodelo del punto de vista de cooperación de proceso de negocio es similar al anterior punto de vista de proceso de negocio, en este punto de vista se puede mostrar como los diferentes roles interactúan y colaboran para lograr realizar los procesos de la organización que se esta modelando.

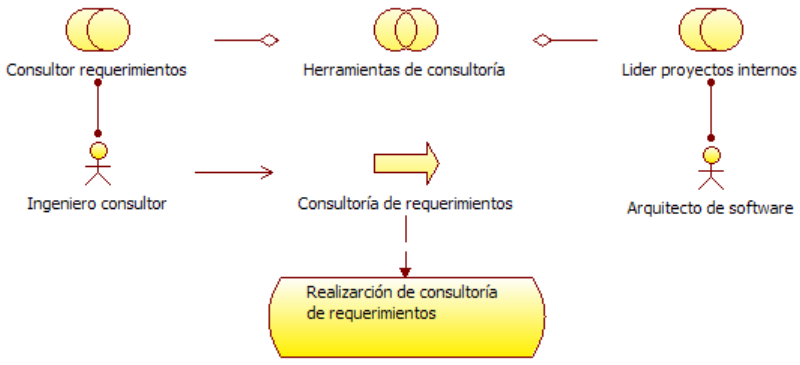


Figura 4.11 Proceso de Negocio

El rol de ingeniero consultor se apoya en las herramientas internas desarrolladas por el equipo de desarrollo interno en cabeza del rol de líder de proyectos para llevar a cabo sus tareas de consultoría.

4.7. Punto de Vista de Producto

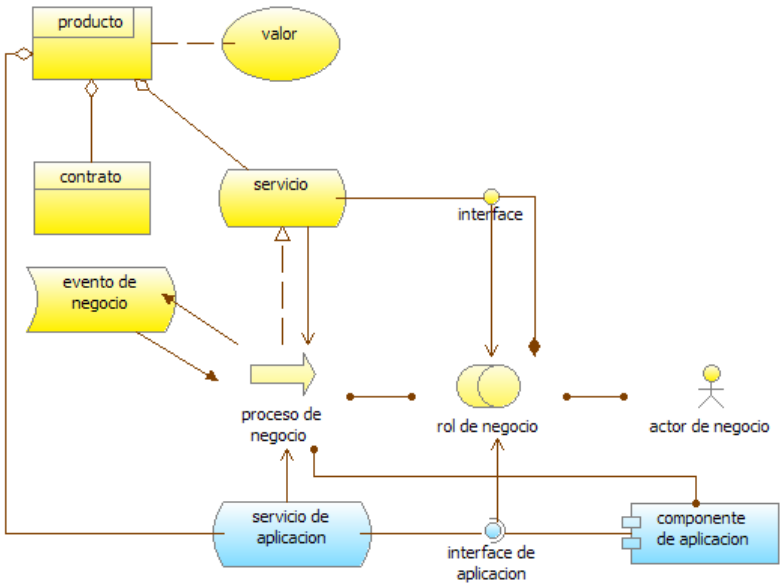


Figura 4.12 Metamodelo Vista de Producto

El punto de vista de producto permite descubrir cuales son los productos que la organización posee o que se van a desarrollar con la propuesta de la arquitectura y que son relevantes para el negocio. Estos productos deben estar relacionados con un servicio de la organización. En este punto de vista se comienzan a identificar los objetos de negocio que van a representar información de la organización. La característica mas importante de este punto de vista es que permite mostrar los valores como aportes que diferencian dicho producto en el mercado.

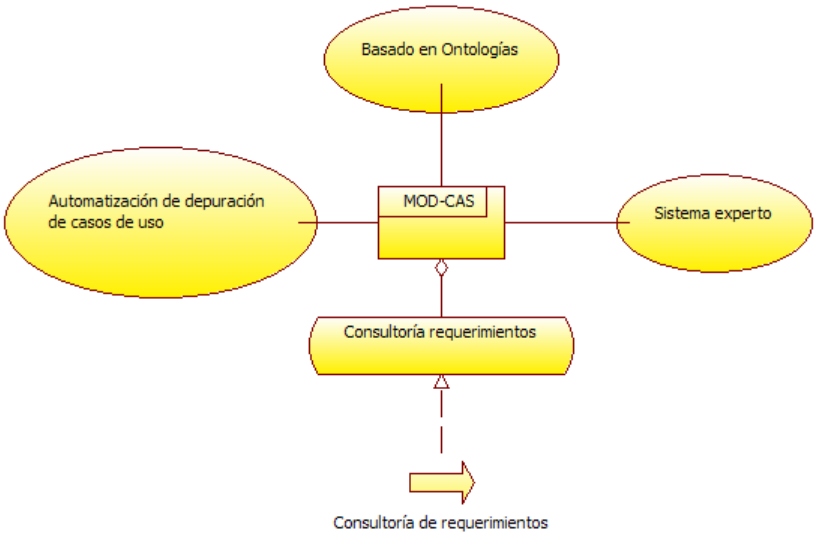


Figura 4.13 Vista de Producto

MOD-CAS Es una aplicación que valiéndose de un modelo ontológico OWL, permite realizar diferentes validaciones a una especificación de caso de uso aportando consistencia al requerimiento que se encuentra expresado en dicha especificación.

Por medio de esta aplicación los consultores pueden realizar mayor cantidad de trabajo en menor tiempo gracias a que la aplicación aporta información sobre el estado del requerimiento y la posible forma de ser mejorado.

Capítulo 5

Capa de Aplicaciones

5.1. Definición

La capa de aplicaciones muestra las relaciones de componentes de software y los servicios que estos deben proveer a la arquitectura.

5.2. Punto de Vista de Comportamiento de Aplicación

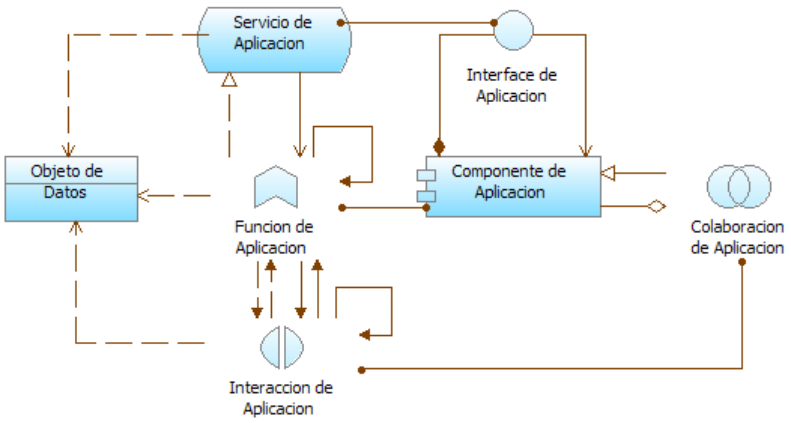


Figura 5.1 Metamodelo Comportamiento de Aplicación

El metamodelo del punto de vista de comportamiento de aplicación muestra los componentes de nivel de software que se van a encontrar a nivel de

capa de aplicación. El objetivo de este punto de vista es relacionar las aplicaciones con los servicios que prestan en la organización y evidenciar las funciones de cada uno de las partes de la aplicación.

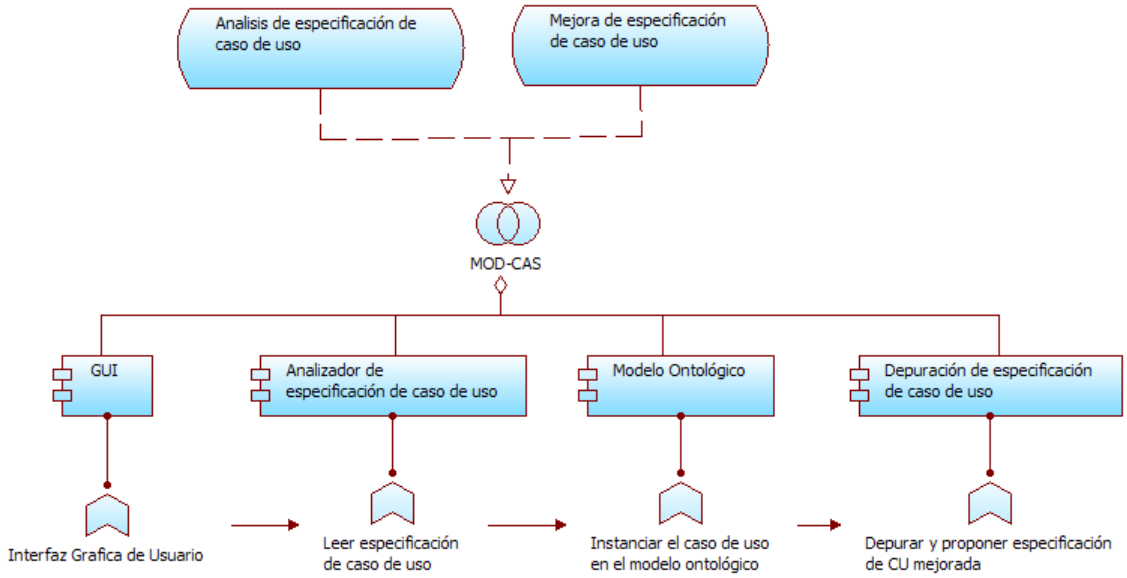


Figura 5.2 Comportamiento de Aplicación

MOD-CAs tiene como objetivo prestar dos servicios: analizar las especificaciones de casos de uso, y presentar mejoras a las especificaciones analizadas.

La aplicación MOD-CAS esta diseñada para cumplir con cuatro funciones que son presentar la información al usuario, analizar el contenido del caso de uso que se esta procesando, completar el modelo ontológico base con la información del caso de uso y finalmente depurar el caso de uso por medio de los mecanismos que provean las APIs para inferir información de una ontología.

5.3. Punto de Vista de Cooperación de Aplicación

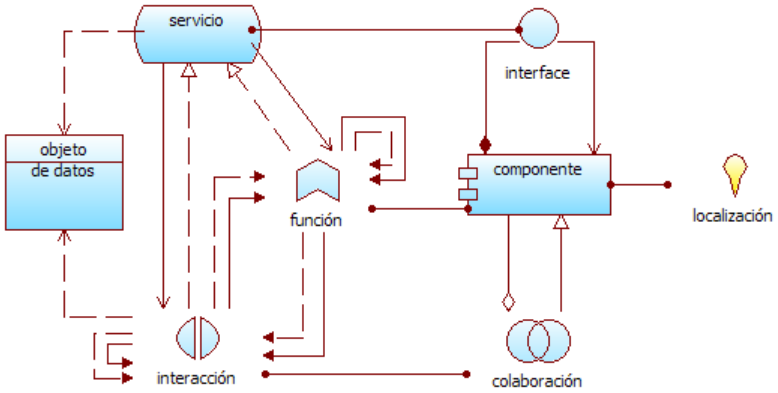


Figura 5.3 Cooperación de Aplicación

El punto de vista de cooperación de aplicación permite mostrar adicional a los elementos de componentes, funciones y datos de las aplicaciones una ubicación lógica de dichos componentes mediante el mismo elemento de ubicación del lenguaje Archimate. Esto permite ver de forma lógica la distribución de los componentes de un sistema que pueden convertirse en módulos lógicos o distribuciones en capas dependiendo de la arquitectura propuesta.

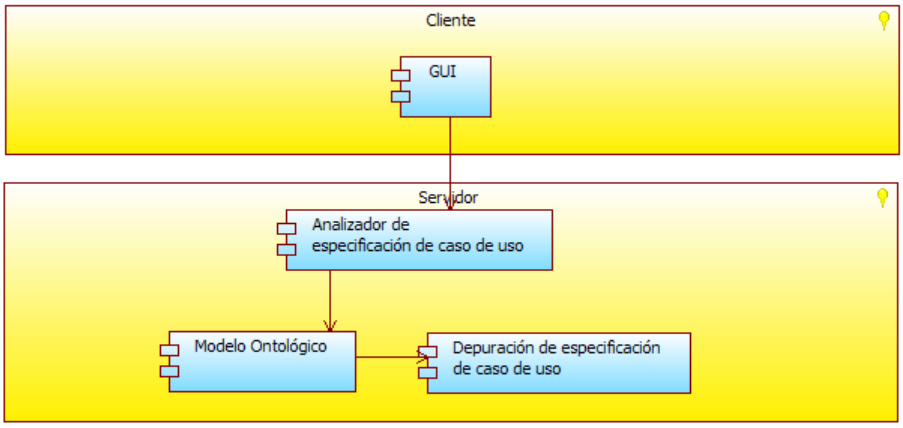


Figura 5.4 Cooperación de Aplicación

La aplicación esta dividida en dos capas, en el cliente se ejecuta la interfaz de usuario y el procesamiento y las mejoras propuestas en para el caso de uso analizado se ejecutan en el servidor

5.4. Punto de Vista de Estructura de Aplicación

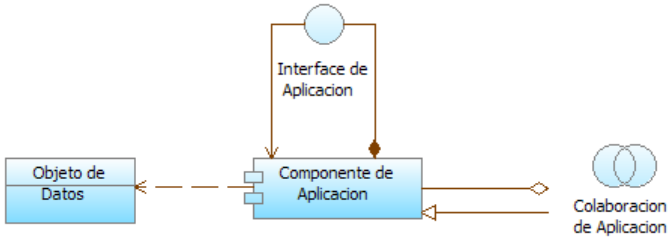


Figura 5.5 Metamodelo Estructura de Aplicación

En el punto de vista de la estructura de aplicación se definen las interfaces que los componentes de la aplicación van a exponer con el fin de comunicarse entre si. Los objetos de datos se relacionan con cada uno de los componentes permitiendo identificar que información se va a manejar en dicho componente.

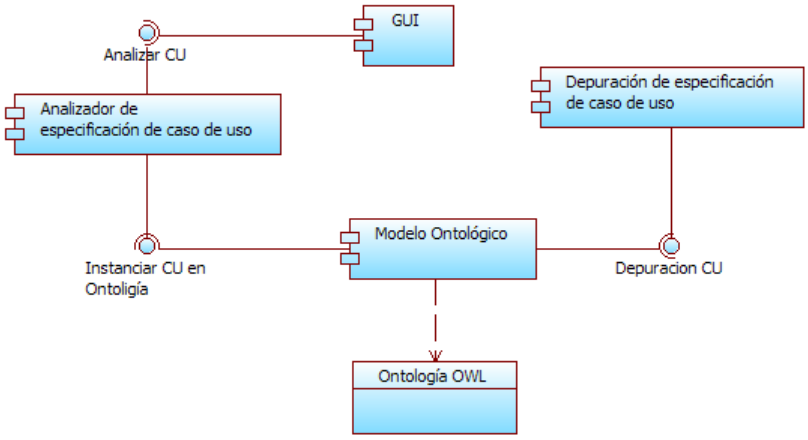


Figura 5.6 Estructura de Aplicación

La aplicación esta compuesta de cuatro componentes:

1. **GUI:** Presenta las opciones de analizar especificación y los resultados del análisis con sus posibles mejoras

2. **Analizador de especificación:** Carga el documento de especificación, y estructura en memoria cada una de las partes y las acciones descritas en el caso de uso

3. **Modelo Ontológico:** Permite que se complete la información del modelo ontológico propuesto con la información del caso de uso que se esta analizando.

4. **Depuración de especificación de caso de uso:** Realiza los procesos de validación del modelo ontológico para el caso de uso y los procesos de razonamiento sobre la ontología para generar información del caso de uso modelado.

5.5. Punto de Vista de Uso de Aplicación

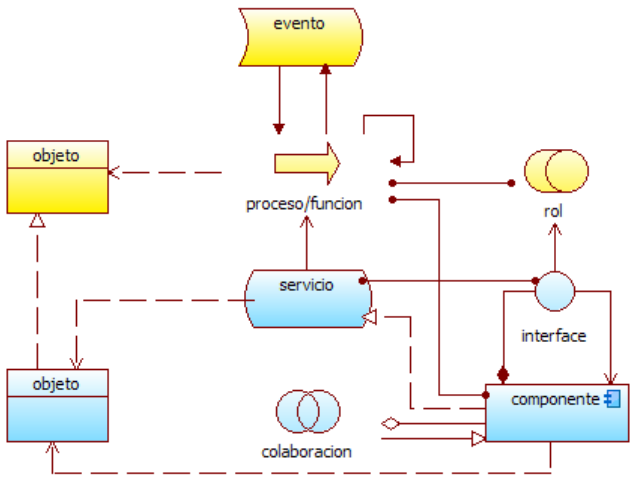


Figura 5.7 Metamodelo Uso de Aplicación

Las aplicaciones presentes en el diseño de la arquitectura tanto en el As Is como en el To Be deberían apuntar a suplir necesidades o servicios de negocio

identificados en la capa de negocio. El punto de vista de uso de aplicación permite relacionar los servicios que prestan las aplicaciones con los procesos de la organización el rol que la ejecuta y los eventos que disparan el uso de la aplicación modelada.

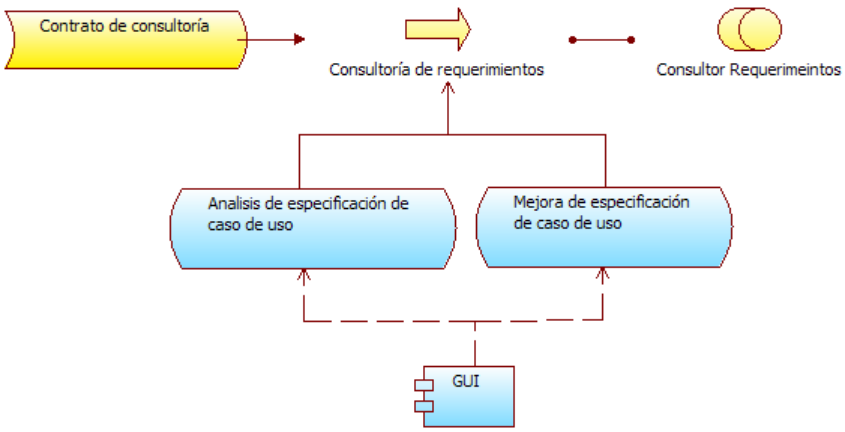


Figura 5.8 Uso de Aplicación

La aplicación MOD-CAS se comienza a usar cuando se genera un contrato para hacer un estudio de consultoría de requerimientos con el fin de apoyar el proceso de análisis de los casos de uso y proponer mejoras para las ambigüedades que se encuentren. La aplicación debe ser un apoyo para el consultor de requerimientos que podrá analizar mas casos de uso y evitar posibles omisiones en el resultado de la consultoría.

Capítulo 6

Nivel de Infraestructura

6.1. Definición

El nivel de infraestructura permite modelar los elementos de tecnología que se usaran para soportar la aplicación.

6.2. Punto de Vista de Infraestructura

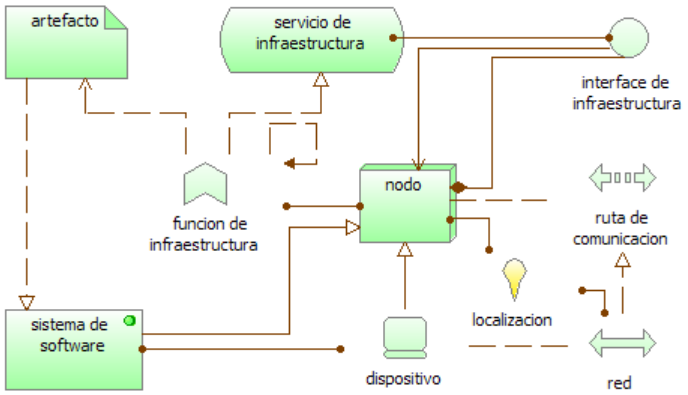


Figura 6.1 Metamodelo de Infraestructura

El metamodelo de punto de vista de Infraestructura muestra los servicios y funciones de cada uno de los componentes de la infraestructura de la arquitectura que se esta diseñando. El principal componente del punto de vista

es el nodo que representa un componente de hardware en el que se despliegan diferentes sistemas de software. El punto de vista permite relacionar una ubicación física o lógica a cada nodo y las rutas de comunicación entre los componentes.

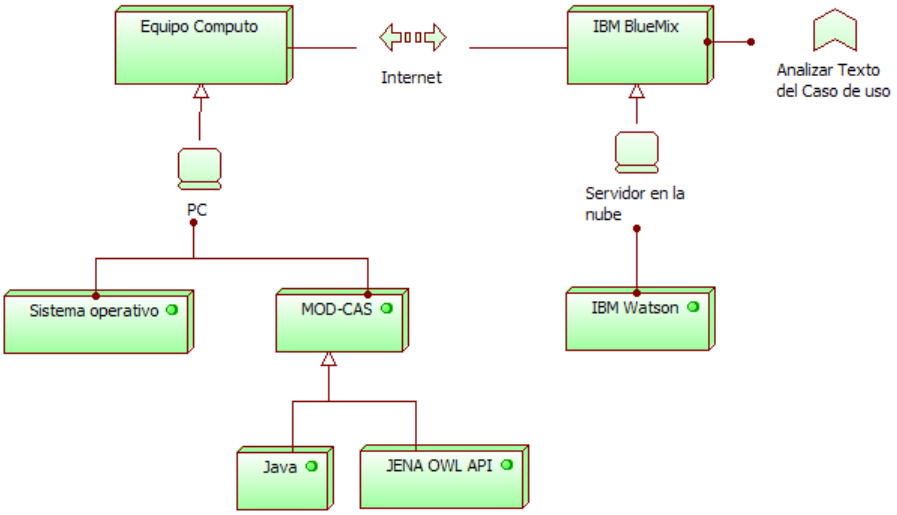


Figura 6.2 Infraestructura

MOD-CAS se desarrollara en lenguaje Java versión 1.8 permitiendo instalarse en diferentes sistemas operativos. Para el uso de la ontología diseñada se trabajara con la librería JENA que provee todas las características necesarias para el trabajo con OWL y RDFS.

MOD-CAS requiere hacer un análisis semántico del contenido de cada uno de los pasos del caso de uso que se este procesando, para lograr esto MOD-CAS hace uso de un servicio REST de IBM que permite analizar y desglosar una oración o párrafo obteniendo sus componentes mas importantes como sujeto, verbo principal y predicado. Este análisis se realiza con el objetivo de verificar si una acción esta bien formada, clasificar el tipo de acción que se esta expresando y obtener información acerca del actor y del recurso de la acción.

6.3. Uso de Infraestructura

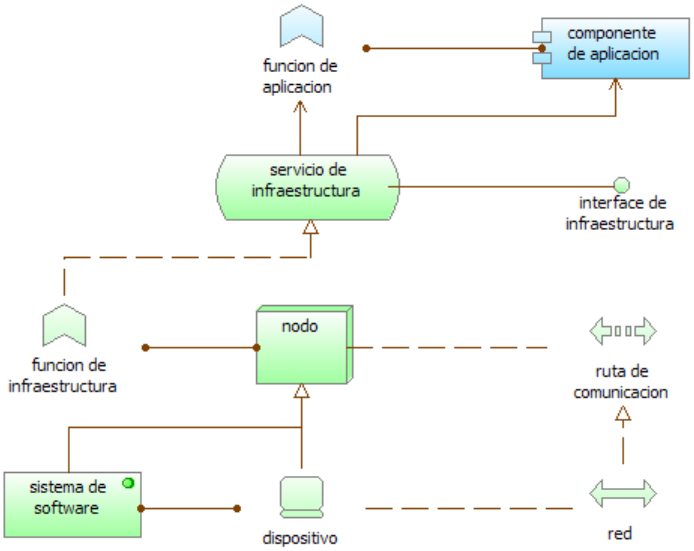


Figura 6.3 Metamodelo de Uso de Infraestructura

El punto de vista de uso de infraestructura permite ver que componentes diseñados en la capa de aplicación serán desplegados en cada uno de los nodos definidos en el punto de vista de infraestructura.

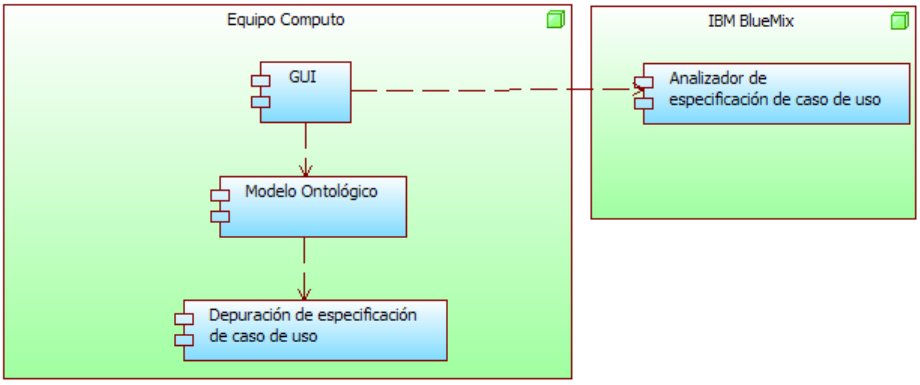


Figura 6.4 Uso de Infraestructura

Desde un punto de vista de uso de la infraestructura la aplicación MOD-CAs se separa en dos componentes, un cliente que se encarga de ejecutar la

interfaz de usuario y realizar los análisis del caso de uso que se este procesando y un componente en una tecnología Cloud de IBM que se usa como PaaS que permite usar un servicio de IBM para realizar análisis semántico de textos.

6.4. Organización e Implementación

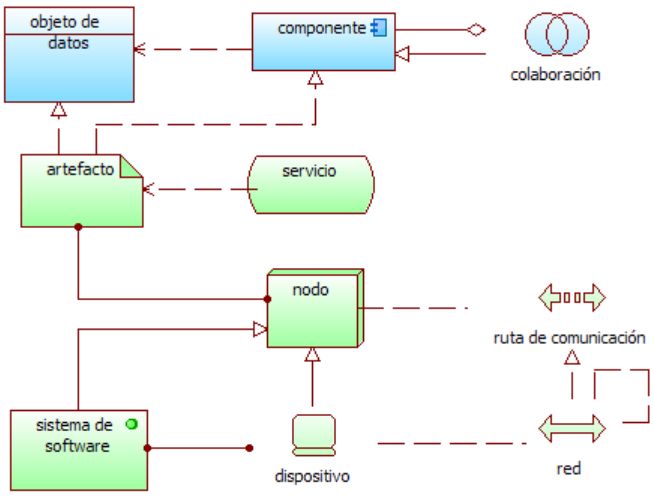


Figura 6.5 Metamodelo de Organización e Implementación

El metamodelo muestra como relacionar los nodos de la infraestructura con los objetos y componentes de aplicación. Los artefactos definidos deben prestar un servicio que le permita a las demás capas conseguir los objetivos estratégicos que se definieron en la visión del proyecto.

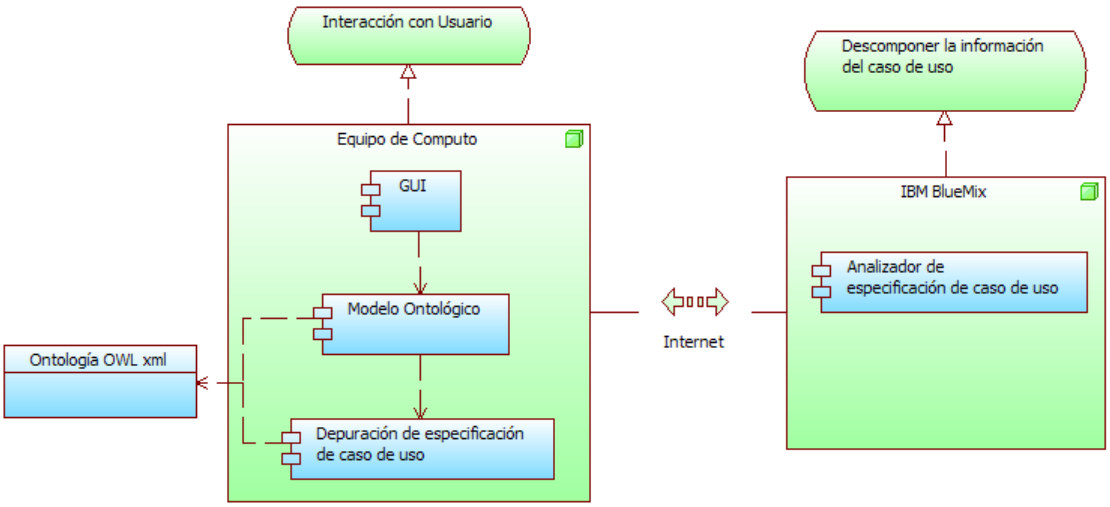


Figura 6.6 Organización e Implementación

Este punto de vista muestra la responsabilidad de cada parte de la infraestructura y de los componentes lógicos que se instalan en cada capa de la aplicación. En el cliente se instalan los componentes de interacción con el usuario y también los componentes encargados de leer la estructura inicial de la ontología y completarla con la información del caso de uso y presentar los resultados del análisis. En el servicio de IBM se construye un componente que permite invocar el servicio de análisis semántico de IBM y la comunicación se establece por medio de protocolo RESTful, los resultados son retornados al cliente en notación JSON.

6.5. Punto de Vista de Estructura de Información

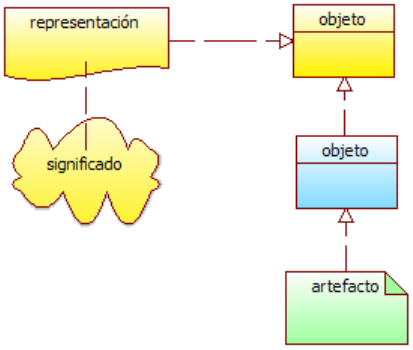


Figura 6.7 Metamodelo de Estructura de Información

El metamodelo de estructura de la información relaciona los objetos identificados en la capa de negocio con su representación de objetos de aplicación. Los objetos de aplicación se pueden relacionar con artefactos de la capa de infraestructura que pueden dar soporte como persistencia a la información que representan.

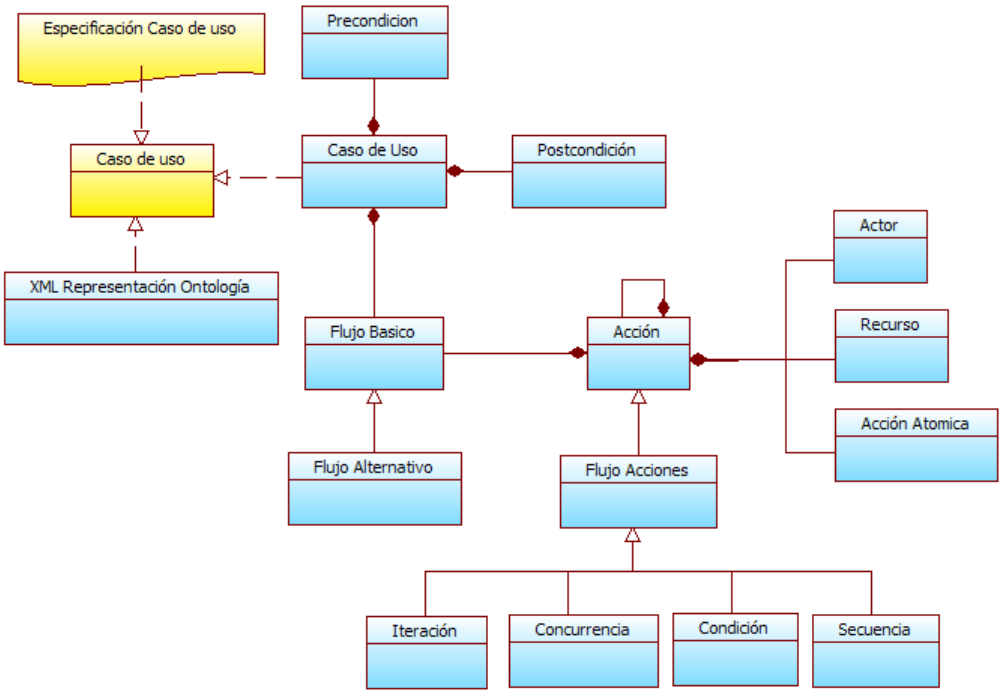


Figura 6.8 Estructura de Información

El punto de vista muestra la relación entre las entidades identificadas del negocio que básicamente es un caso de uso, representado en un documento de especificación de caso de uso. El caso de uso tiene varios componentes representados en objetos de datos y sus relaciones así:

- Caso de uso: Compuesto de flujos básicos o alternativos, precondiciones y postcondiciones.
- Precondición: Representa un estado necesario del sistema para ejecutar satisfactoriamente el caso de suo.
- Postcondición: Representa el estado que el sistema debe alcanzar después de ejecutarse el caso de uso.
- Flujo Básico: La serie de pasos o acciones principales del caso de uso.
- Flujo alternativo: Representa la serie de pasos o acciones que se desprenden de una condición especifica en un paso del flujo básico.
- Acción (Paso): Representa una requerimiento puntual que un actor solicita al sistema sobre un recurso. También puede representar una acción del sistema en la que se responde a una acción solicitada por el actor. Se pueden definir algunos subtipos de acciones:
    1. Iteración: Una acción que se repite un numero determinado de veces.
    2. Concurrencia: Una acción que se ejecuta a al mismo tiempo que otras acciones.
    3. Condición: Una acción que puede sugerir una ejecución de un flujo alternativo dadas unas condiciones especificas.

6.6. Punto de Vista de Realización del Servicio

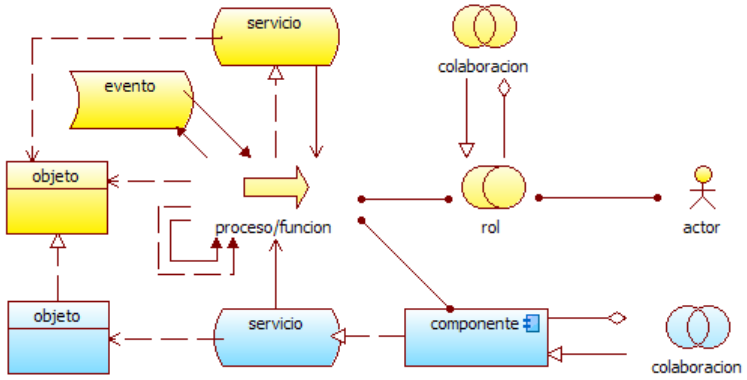


Figura 6.9 Metamodelo de Realización del Servicio

El metamodelo de realización de servicio relaciona los procesos y funciones de negocio con los servicios de las aplicaciones diseñadas. Tiene como objetivo evidenciar que los procesos que la arquitectura identifico o propuso tiene su soporte en un servicio que provee una aplicación diseñada y que los datos que el proceso necesita están contemplados como datos que van a manejar las aplicaciones.

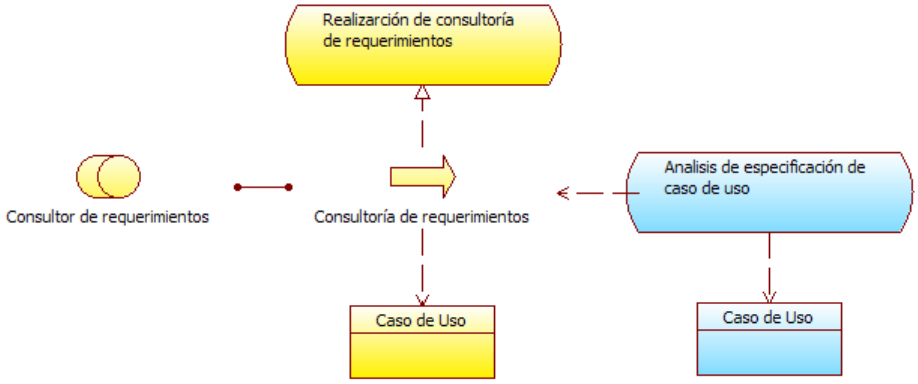


Figura 6.10 Realización del Servicio

El punto de vista muestra como se relaciona el servicio de negocio y el proceso de negocio con el servicio de la aplicación diseñada. MOD-CAS es

una aplicación pensada para apoyar labores de consultoría y de especificación de casos de uso, la labor del ingeniero es verificar la calidad de las especificaciones que se están estudiando, de modo que la aplicación le pude informar no solo que esta mal sino sugerir como mejorar la especificación.

6.7. Punto de Vista de Capas

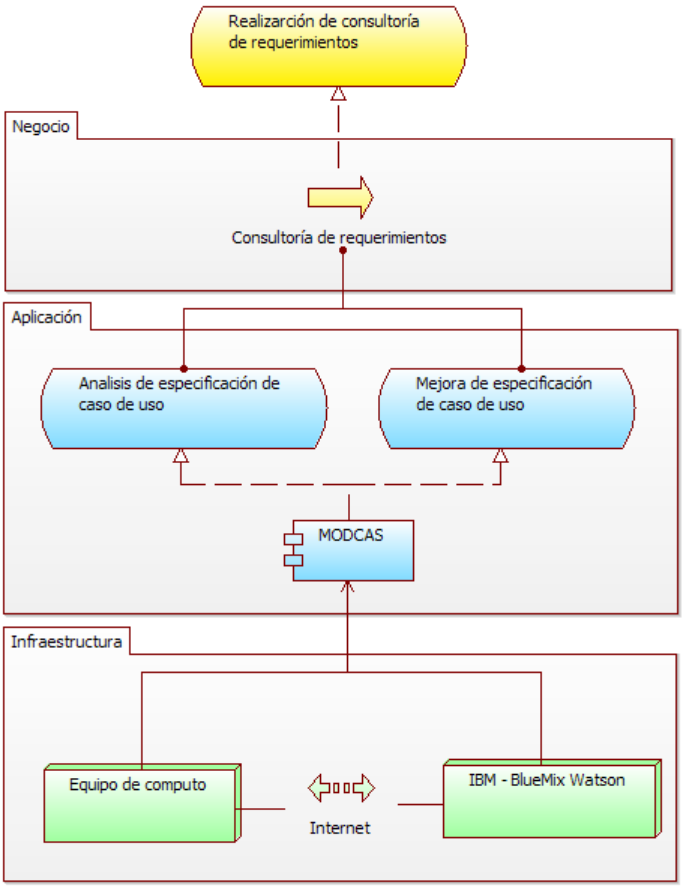


Figura 6.11 Vista de Capas

El punto de vista presenta las capas de la aplicación MOD-CAS, evidenciando cuales son sus principales componentes y funciones a nivel de negocio, aplicación e infraestructura.

MOD-CAS se va a ejecutar en un equipo de computo de escritorio, pero necesita de un servicio que provee un servidor en internet, la aplicación permite analizar y proponer mejoras a especificaciones de casos de uso con el fin de apoyar procesos de consultoría de requerimientos.

Capítulo 7

Motivación

7.1. Definición

La vista de Motivación presenta información acerca de los interesados, las motivaciones y expectativas de implementar el proyecto.

7.2. Punto de Vista de Stakeholder

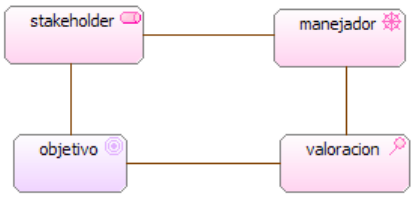


Figura 7.1 Metamodelo Vista de Stakeholder

El metamodelo de la vista de Stakeholder relaciona los interesados con sus objetivos, drivers de negocio y los valores que estos objetivos tienen para el interesado.

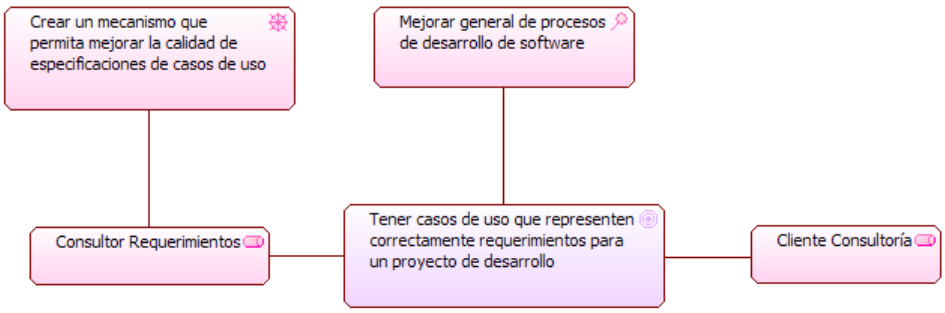


Figura 7.2 Vista de Stakeholder

Se pueden identificar dos stakeholder principales, el cliente de la consultoría que desea asegurar la calidad de los requerimientos que pretende implementar y el consultor que usara MOD-CAS como apoyo para estudiar los casos de uso que esta encargado de validar.

7.3. Punto de Vista de Realización de Objetivos

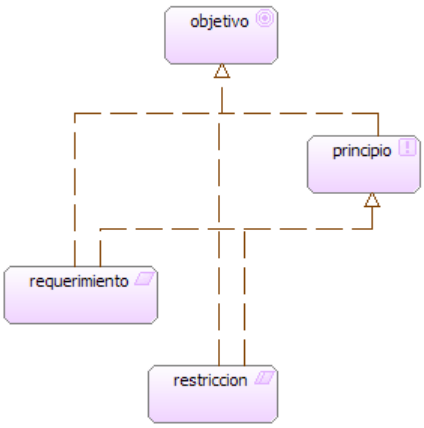


Figura 7.3 Metamodelo Vista de Realización de Objetivos. Fuente [Archimate 2.0][Diseñado con Coloso]

El metamodelo de realización de objetivos muestra como los objetivos son delimitados por requerimientos, restricciones y principios de una organización.

Las restricciones pueden ser de diferentes tipos, como por ejemplo legales, de tiempo o tecnológicas.

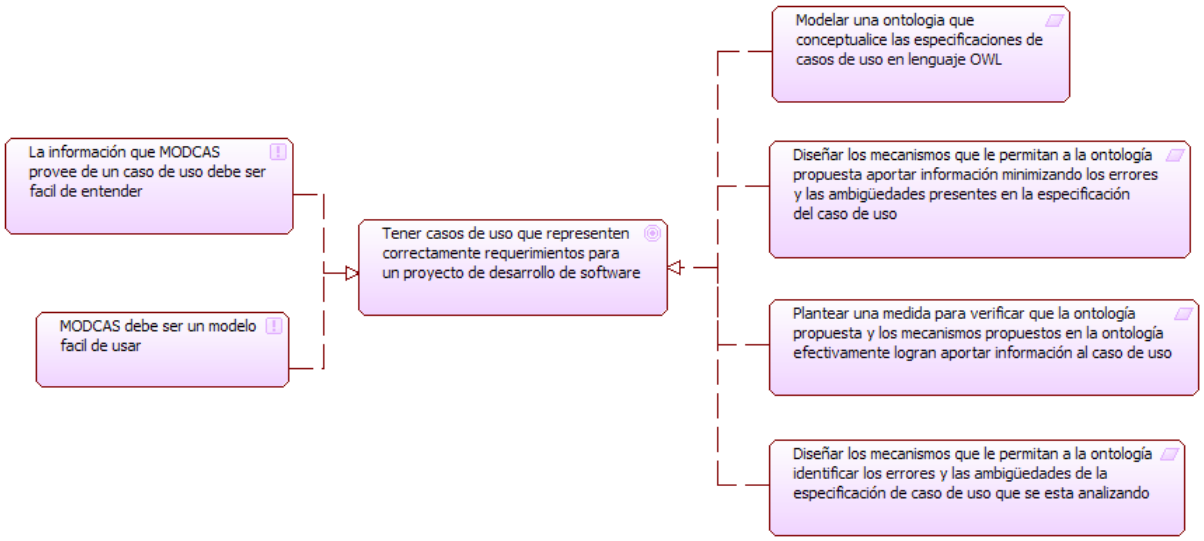


Figura 7.4 Vista de Realización de Objetivos. Fuente: [Autor]

El objetivo principal que motiva la construcción de la aplicación es poder contar con casos de uso que se encuentren correctamente especificados, para que redunde en la calidad de las implementaciones y las pruebas del desarrollo que se desee llevar a cabo.

7.4. Punto de Vista de Contribución

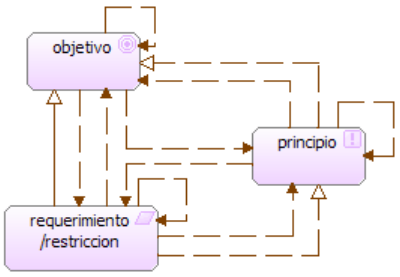


Figura 7.5 Metamodelo de Vista de Contribución

El metamodelo de vista de contribución añade al objetivo la calificación de los requerimientos y las restricciones que permiten identificar si estos afectan positiva o negativamente la capacidad de lograr el objetivo propuesto.

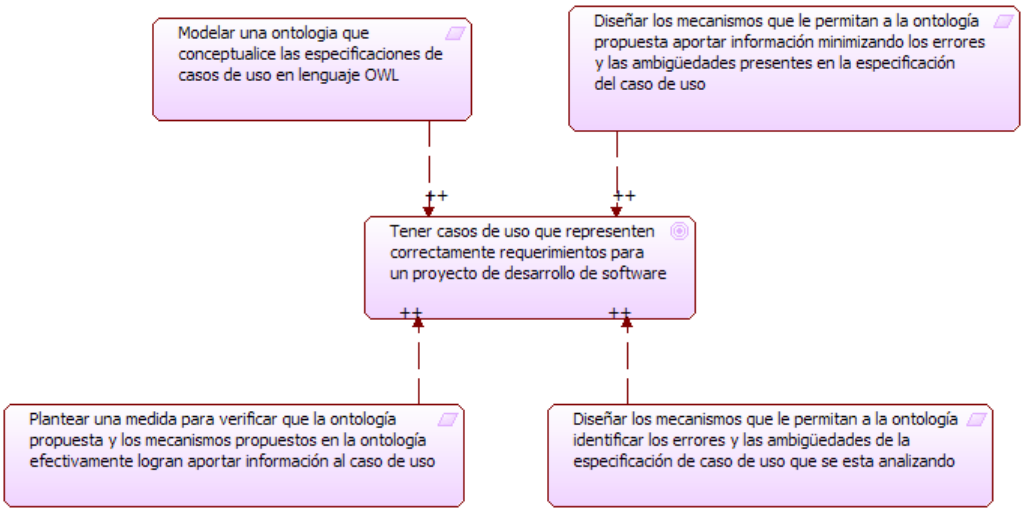


Figura 7.6 Vista de Contribución

Las contribuciones de los requerimientos al objetivo de tener casos de uso mas claros y consistentes son positivas, generan un aporte al proceso de desarrollo de software y aumentan las posibilidades de que todo el proceso de un proyecto de software culmine de manera exitosa.

7.5. Punto de Vista de Principios

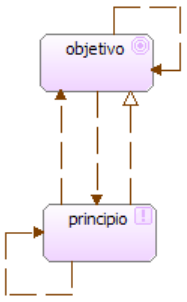


Figura 7.7 Metamodelo de Vista de Principios

Los principios son directivas que la organización espera cumplir siempre en sus proyectos, los objetivos deben enmarcarse en estos principios de la organización o del proyecto.

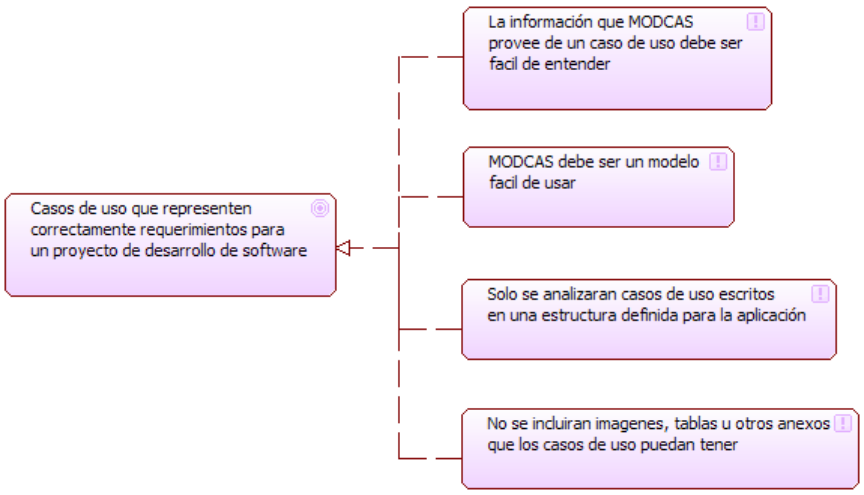


Figura 7.8 Vista de Principios

En principios MOD-CAS debe ser una aplicación fácil de usar, debe permitirle al consultor de requerimientos observar fácilmente si un caso de uso presenta ambigüedades y de la misma forma debe presentarle como podría mejorarse ese caso de uso.

Dado que existen múltiples formatos para escribir una especificación de

casos de uso MOD-CAS analiza únicamente la información de determinadas secciones de las especificaciones, y no toma en cuenta anexos como imágenes o tablas que las especificaciones tengan.

7.6. Punto de Vista de Realización de Requerimientos

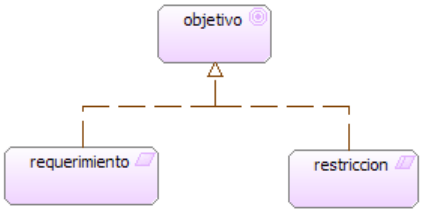


Figura 7.9 Metamodelo de Realización de Requerimientos

El metamodelo muestra como los objetivos se realizan en términos de requerimientos y restricciones, los requerimientos o restricciones del punto de vista se especifican en requerimientos mas puntuales que lleven a conseguir el objetivo.

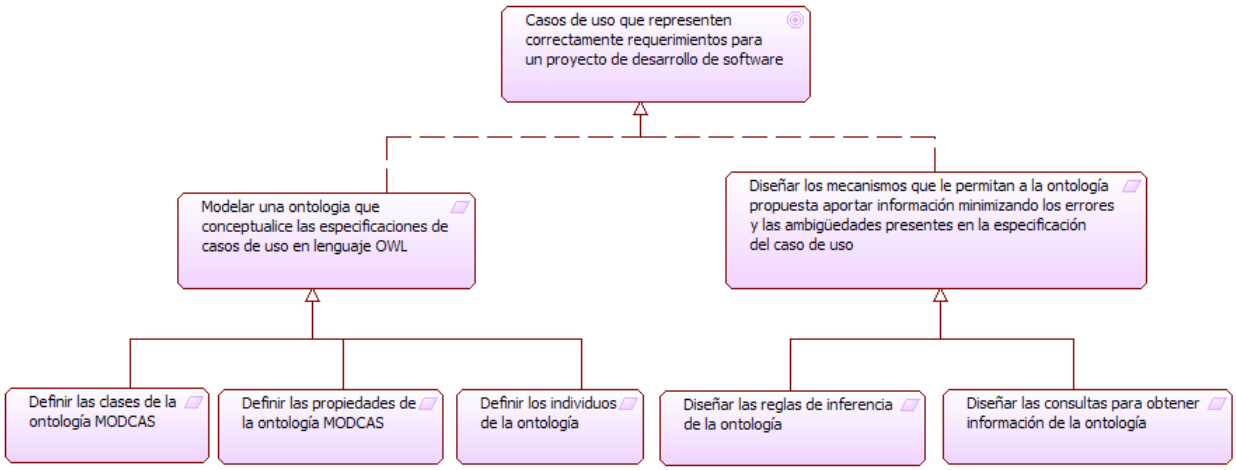


Figura 7.10 Realización de Requerimientos

Para modelar la ontología se seguirá una metodología propuesta en el marco teórico de este trabajo. Las reglas de inferencia y las posibles consultas

que se puedan realizar a la ontología se diseñaran basados en patrones de diseño de ontologías definidos en el marco teórico.

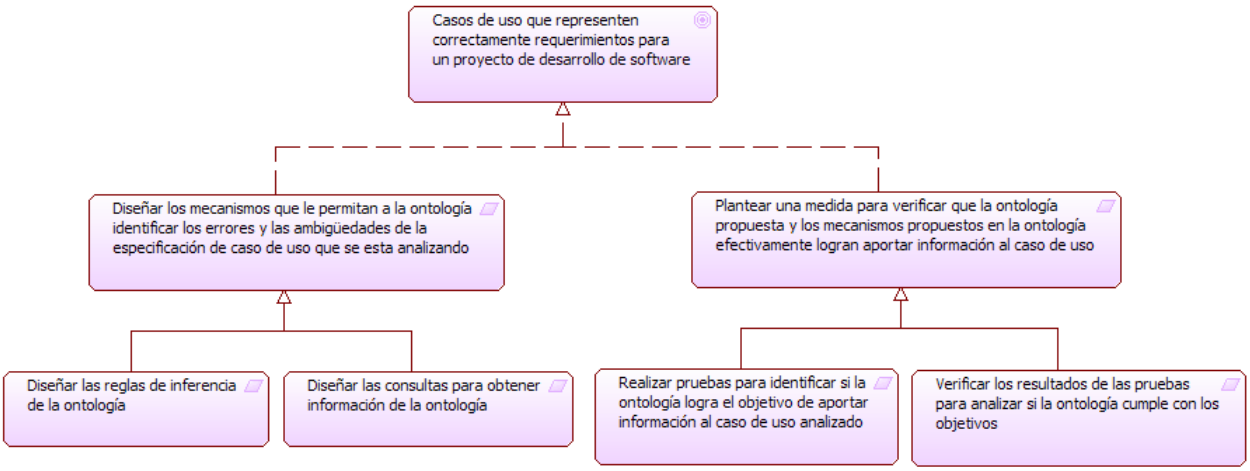


Figura 7.11 Vista de Contribución

Para validar el diseño de la ontología es necesario realizar pruebas al modelo verificando si es capaz de realizar las mejoras esperadas a casos de uso que contengan errores o ambigüedades. Estos resultados se deben medir para poder verificar la eficacia del modelo propuesto.

7.7. Punto de Vista de Motivación

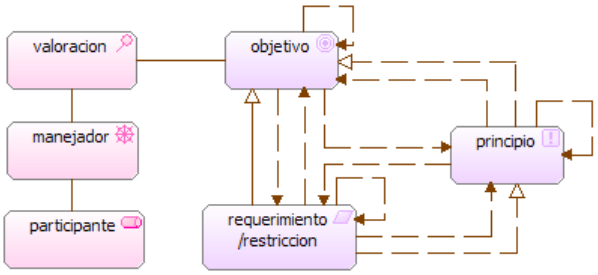


Figura 7.12 Metamodelo de Vista de Motivación

El metamodelo permite relacionar los objetivos propuestos para el proyecto con los interesados los drivers de negocio y las motivaciones que generaron el desarrollo del proyecto.

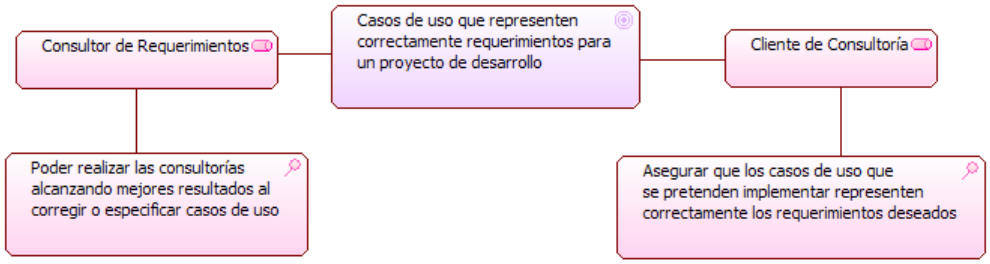


Figura 7.13 Vista de Motivación

Para el ingeniero consultor que debe examinar las especificaciones de casos de uso y dar una opinión de la calidad del caso de uso descrito es muy importante contar con una herramienta que automatice en parte su trabajo y le aporte detalles que pueda pasar por alto.

Para la empresa que contrate la consultoría para verificar la calidad de los casos de uso con los que pretende desarrollar un sistema es importante que pueda confiar en que el trabajo de los consultores logre los mejores resultados y que no se pasen detalles inconsistentes después de la revisión.

7.8. Punto de vista de Proyecto

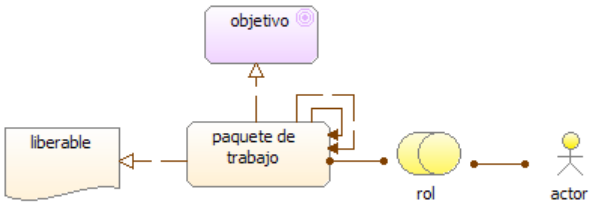


Figura 7.14 Metamodelo de Vista de Proyecto

El metamodelo del punto de vista de proyecto relaciona un objetivo del proyecto con los paquetes de trabajo que se deben realizar para alcanzar dicho objetivo. Cada paquete de trabajo puede tener relacionados liberables que son resultados concretos de terminar un paquete de trabajo.

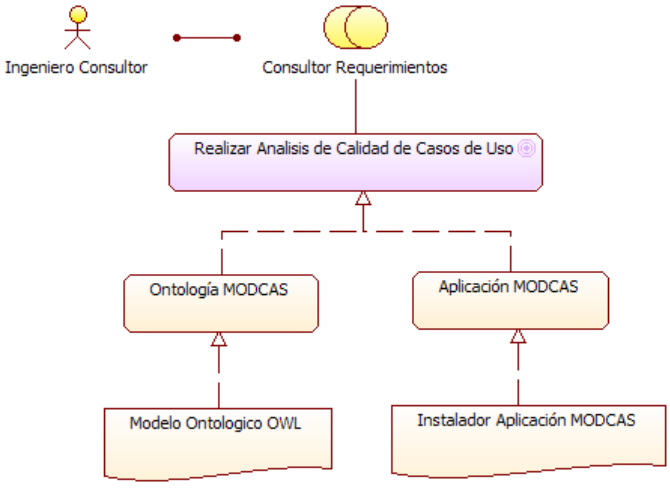


Figura 7.15 Vista de Proyecto

Para cumplir con el objetivo que buscar el consultor de requerimientos el proyecto debe contar con dos paquetes de trabajo: el diseño del modelo ontológico y su validación y la aplicación que permita hacer uso de esta ontología de forma simple y entendible.

7.9. Punto de Vista de Migración

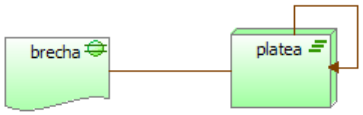


Figura 7.16 Metamodelo de Vista de Migración

El punto de vista de migración relaciona los escalones que se deben lograr para llegar de requerimiento al siguiente y las acciones que se necesitan para alcanzar cada paso.

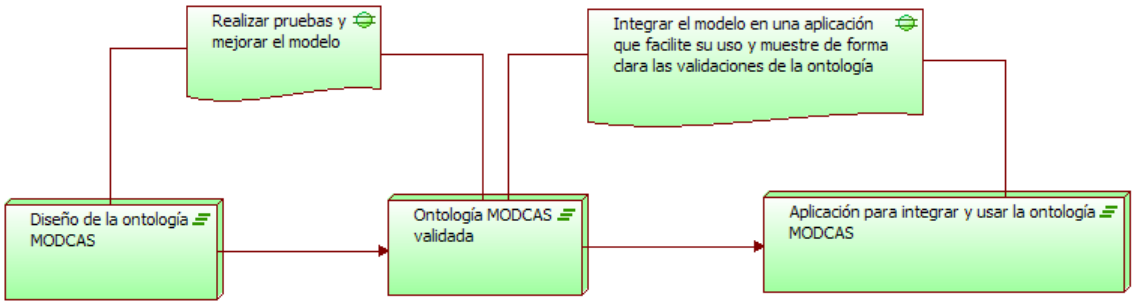


Figura 7.17 Vista de Migración

El desarrollo del proyecto debe contar con tres pasos importantes:

1. Desarrollo del modelo ontológico MOD-CAS en lenguaje OWL que incluya los mecanismos de inferencia necesarios para cumplir con los objetivos del modelo.

2. Validación y refinamiento del modelo ontológico mediante la ejecución de pruebas y el análisis de los resultados obtenidos.

3. La construcción de la aplicación que permita usar de forma fácil las características de la ontología y muestre los resultados que se obtengan al analizar un caso de uso.

7.10. Punto de Vista de Migración e Implementación

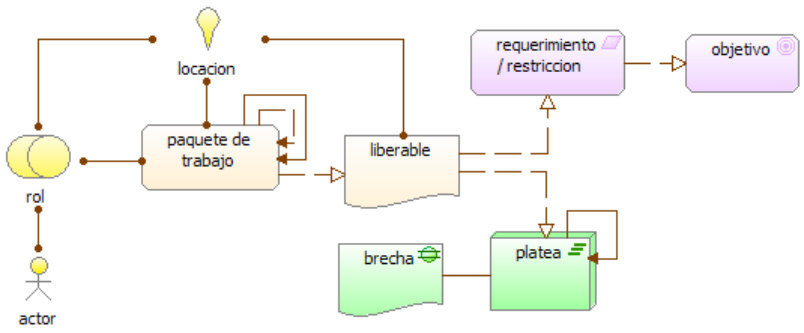


Figura 7.18 Metamodelo de Vista de Migración e Implementación

El metamodelo de las migración e implementación muestra como se relacionan los liberables con las plateas que se deben alcanzar. Permite mostrar en que plateas se lograran los objetivos que se están diseñando.

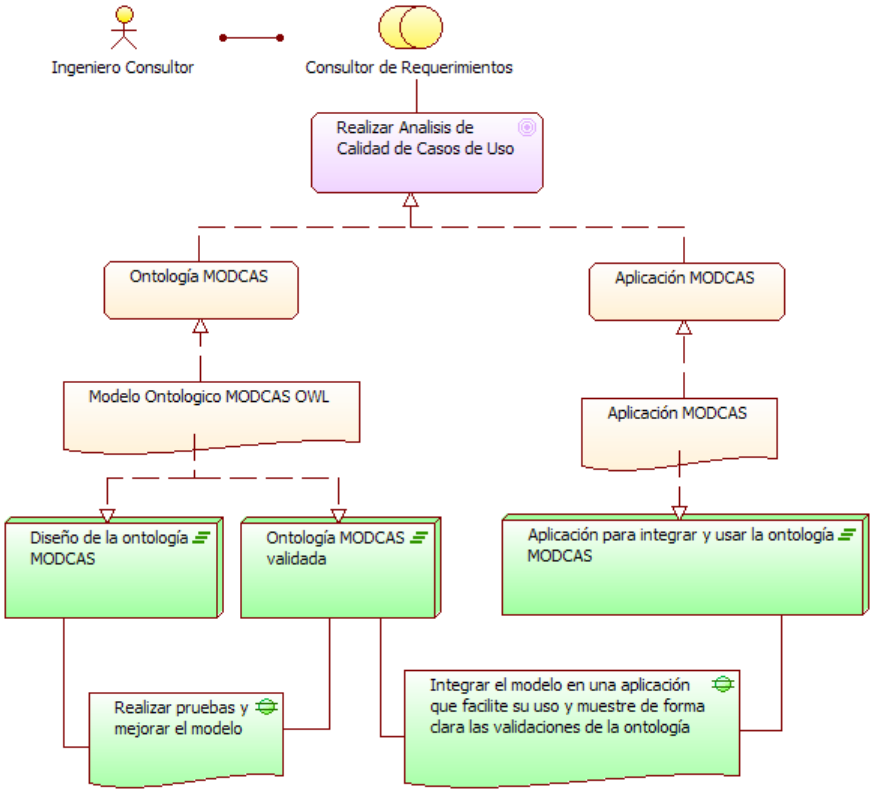


Figura 7.19 Vista de Migración e Implementación

Para lograr realizar el entregable de la ontología se deben alcanzar las dos primeras partes de la implementación del proyecto que consisten en el desarrollo y la validación de la ontología. La siguiente parte consiste en el desarrollo de la aplicación que permita usar las propiedades de la ontología y representa liberable final del proyecto.

Capítulo 8

Arquitectura de bajo nivel

Esta sección de la definición de la arquitectura se centra en el modelado de los patrones de diseño necesarios para implementar los componentes propuestos en la capa de aplicación de Archimate.

8.1. Contexto

Habiendo definido la arquitectura de la aplicación MOD-CAS es necesario definir a un nivel técnico las estructuras necesarias para implementar los componentes propuestos en la capa de aplicación.

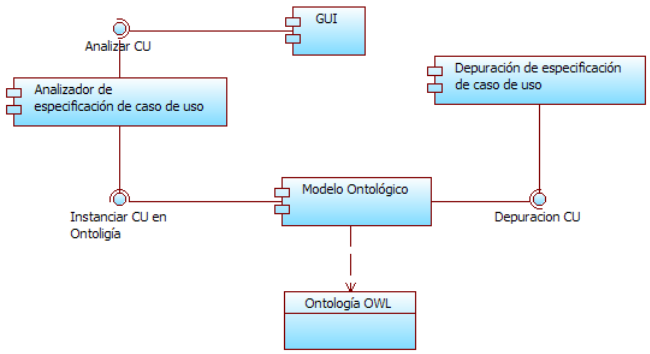


Figura 8.1 Estructura de la aplicación MOD-CAS

Los patrones descritos a continuación tienen como objetivo presentar el diseño a nivel estructural, de creación y de comportamiento de la aplicación MOD-CAS enmarcados dentro de los siguientes componentes:

8.2. Patrones Creacionales

Tienen como objetivo ocultar la complejidad de crear objetos.

8.2.1. Método Fabrica

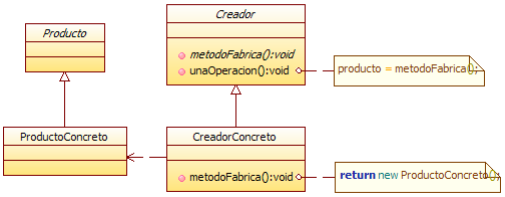


Figura 8.2 Metamodelo de Método Fabrica

El patrón Método Fabrica define una interfaz para crear objetos, que a su vez serán creados por las implementaciones concretas de la interfaz.

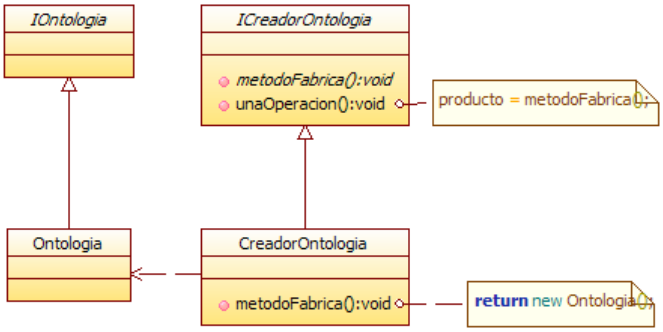


Figura 8.3 Método Fabrica

Este patrón tendrá como objetivo crear la instancia del objeto Ontología que representara la Ontología MOD-CAS con la que trabajara el sistema.

8.2.2. Prototipo

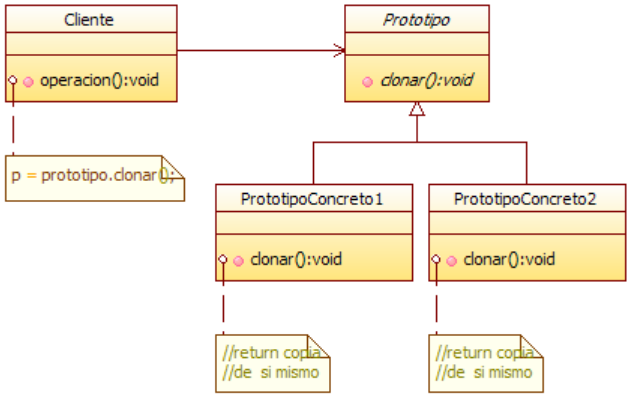


Figura 8.4 Metamodelo de Prototipo

El patrón prototipo permite crear nuevos objetos basándose en un objeto ya existente y funciona como prototipo.

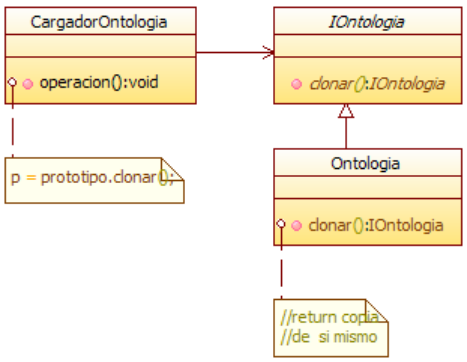


Figura 8.5 Prototipo

Este patrón tendrá como objetivo permitir la clonación del objeto Ontología para realizar las modificaciones y análisis necesarios sin afectar la estructura original del diseño de la Ontología MOD-CAS.

8.3. Patrones Estructurales

Tienen como objetivo solucionar problemas de composición entre clases.

8.3.1. Puente

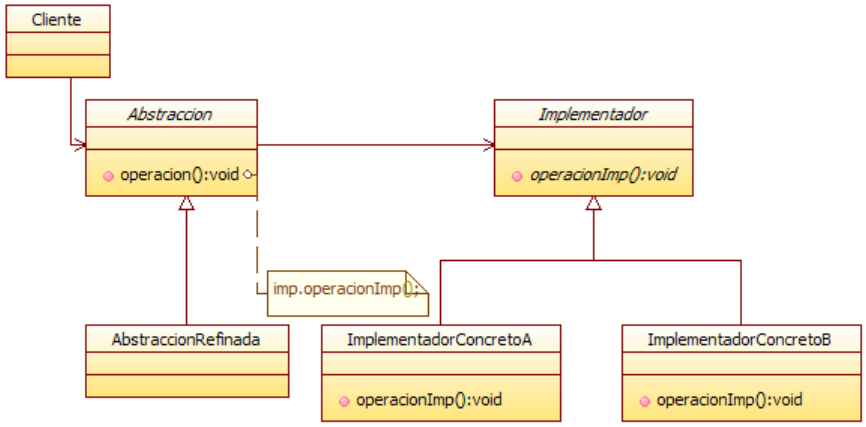


Figura 8.6 Metamodelo de Puente

Permite desacoplar una abstracción de su implementación de forma que cada uno puede variar independientemente.

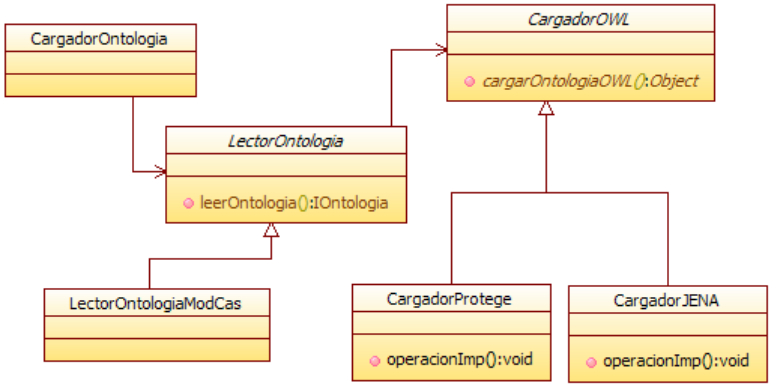


Figura 8.7 Puente Ontología

Esta implementación del patrón Puente permitirá variar de forma fácil el API para trabajos con ontologías que se desee utilizar.

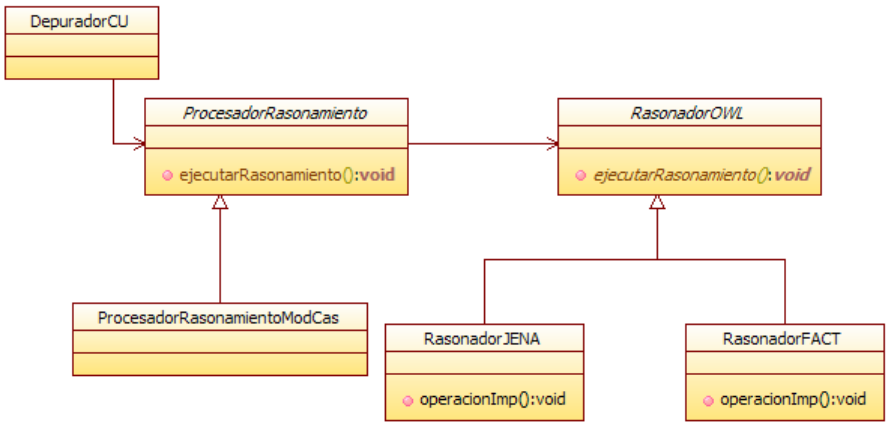


Figura 8.8 Puente Razonador

Esta implementación del patrón Puente permitirá seleccionar el API con la que se desea realizar los trabajos de inferencia de la ontología.

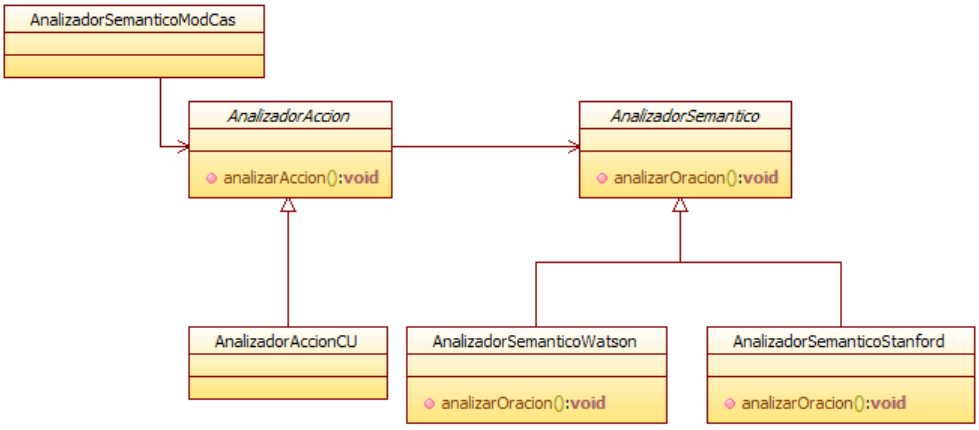


Figura 8.9 Puente Analizador Semántico

Esta implementación del patrón Puente permitirá seleccionar el API con la que se desea realizar el análisis semántico de las acciones del caso de uso.

8.3.2. Componente

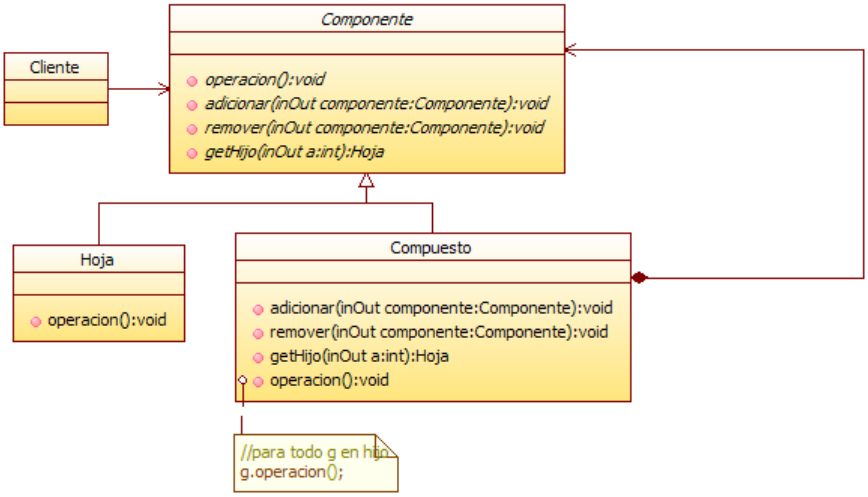


Figura 8.10 Metamodelo de Componente

El patrón componente permite implementar estructuras de objetos de la forma todo-parte. La composición de los objetos forma una jerarquía.

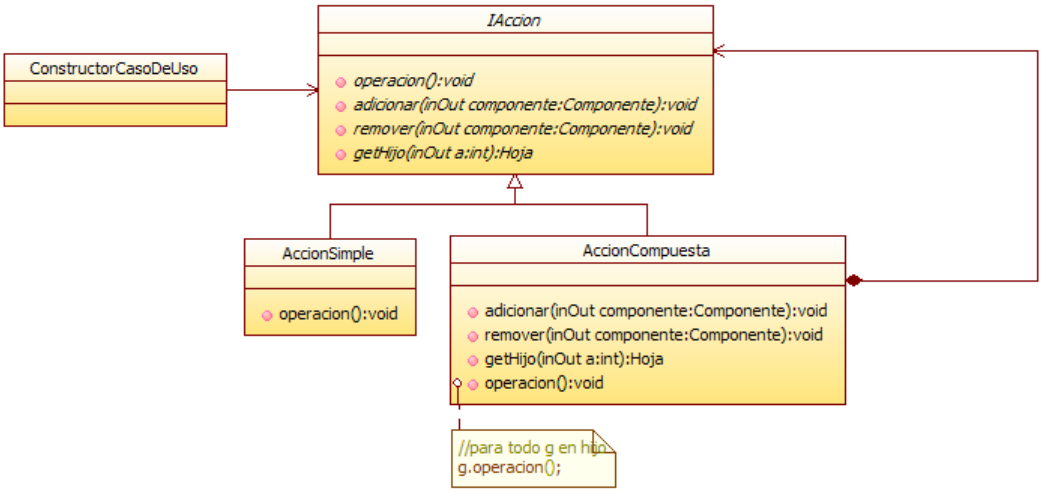


Figura 8.11 Componente

Analizando la composición de un caso de uso, las acciones o pasos de los flujos básico o alternativos pueden componerse de mas de una acción, pero siempre deberían indicar un que se desea ejecutar quien esta ejecutando y

sobre que recurso del sistema se esta ejecutando cada acción, por este motivo se propone un patrón de composición para modelar estas estructuras de los casos de uso.

8.3.3. Fachada



Figura 8.12 Metamodelo de Fachada

El patrón fachada define una interfaz unificada para dar acceso a un componente complejo facilitando su uso.

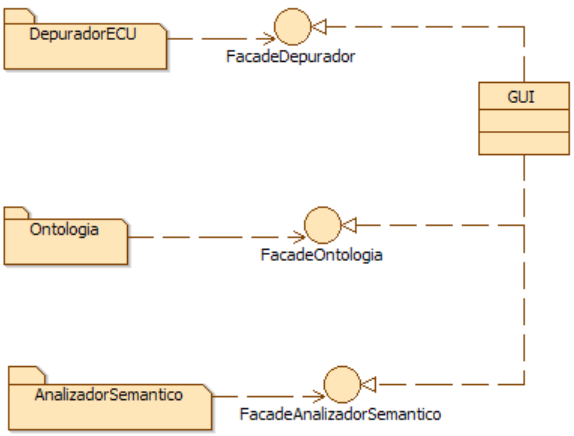


Figura 8.13 Fachada

Los 3 componentes principales de la aplicación MOD-CAS deben exponer una interfaz que permita que su uso sea facil y claro ocultando la complejidad de su implementación y permitiendo que no se acoplen con la interfaz de usuario.

8.3.4. Proxy

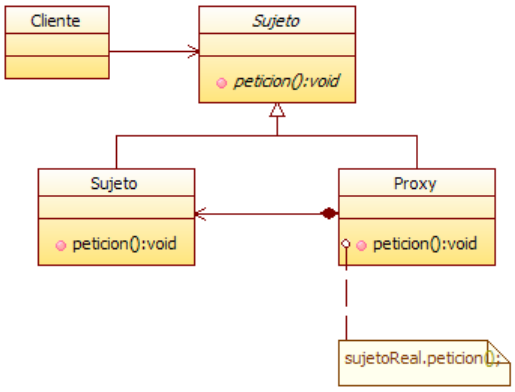


Figura 8.14 Metamodelo de Proxy

El patrón proxy permite a una clase controlar el acceso a un recurso representándolo de forma local.

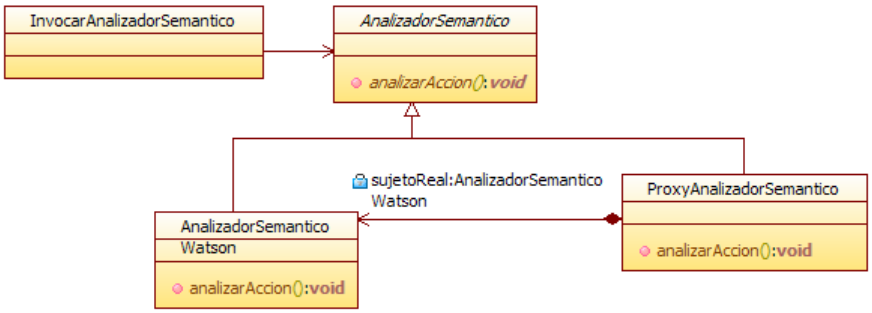


Figura 8.15 Proxy

El analizador semántico tiene como objetivo identificar la composición de las acciones de los pasos en el caso de uso, para conocer quien es el actor, la acción concreta y el recurso del sistema sobre el que se ejecuta la acción. El analizador que se desea usar en el proyecto es un servicio de IBM que se expone en internet de modo que se requiere un proxy para generar las peticiones y tener control de la disponibilidad del servicio.

8.4. Patrones de Comportamiento

Tienen como objetivo solucionar problemas interacción y responsabilidad de las clases.

8.4.1. Comando

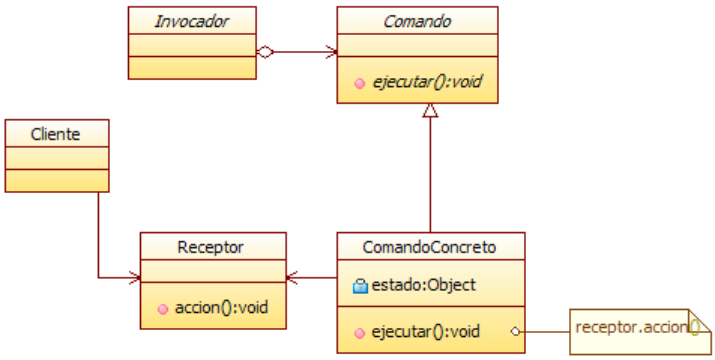


Figura 8.16 Metamodelo de Comando

Permite desacoplar la solicitud de una petición a un objeto, de la implementación que va a ejecutar la operación solicitada.

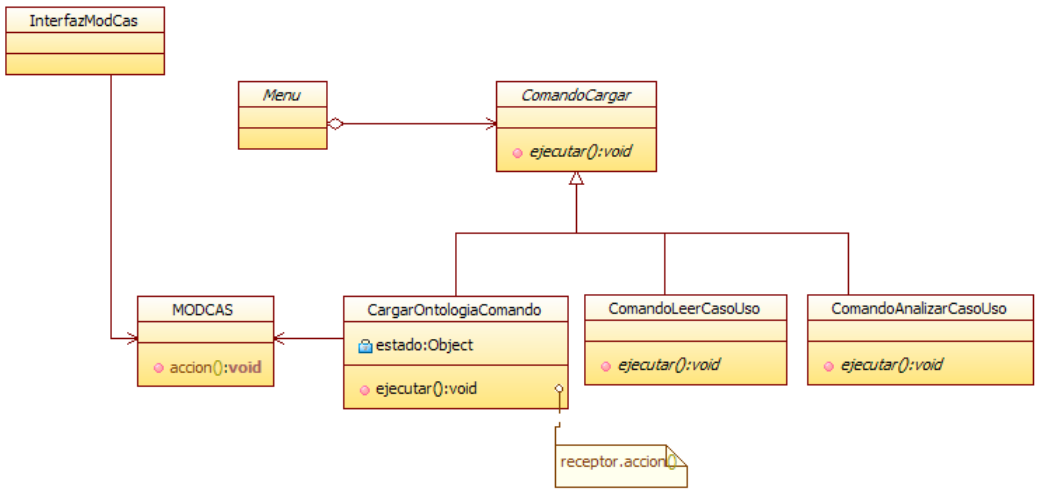


Figura 8.17 Comando

El patrón comando se plantea para desacoplar la interfaz gráfica de las funciones de la aplicación, como leer el caso de uso, cargar la ontología o realizar el análisis del caso de uso.

8.4.2. Mediador

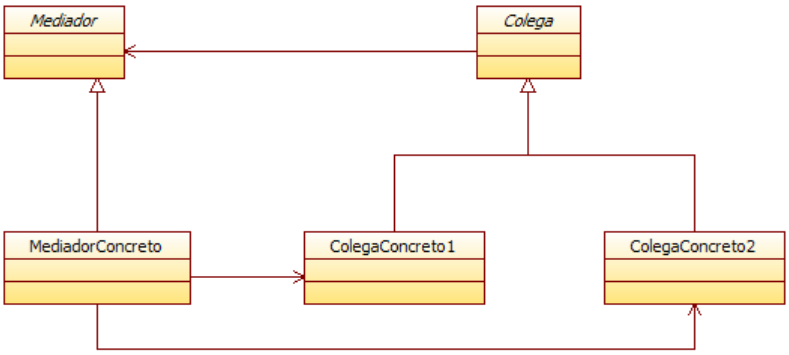


Figura 8.18 Metamodelo de Mediador

El mediador permite encapsular un conjunto de objetos que van a interactuar para conseguir un objetivo sin que se acoplen entre ellos, centrando la orquestación de la operación en el mediador.

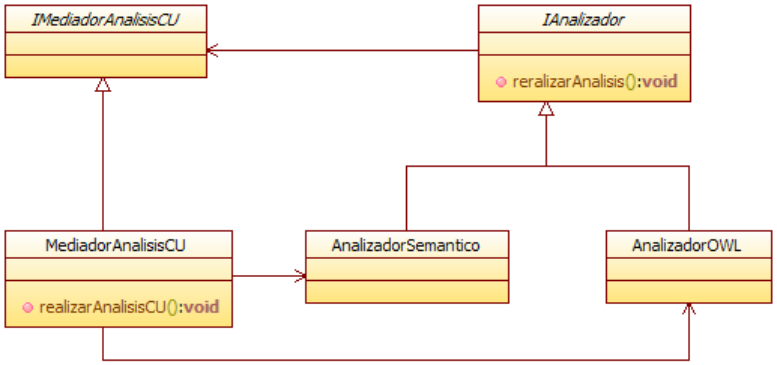


Figura 8.19 Mediador

El patrón mediador permite diseñar el proceso completo que se necesita para realizar el análisis de un caso de uso con la ontología MOD-CAS pero

desacoplando los componentes que se encargan de cada paso del proceso. De esta forma existirá una clase encargada de cada fase del análisis mientras que el mediador se encarga de invocar cada método de análisis.

8.4.3. Visitador

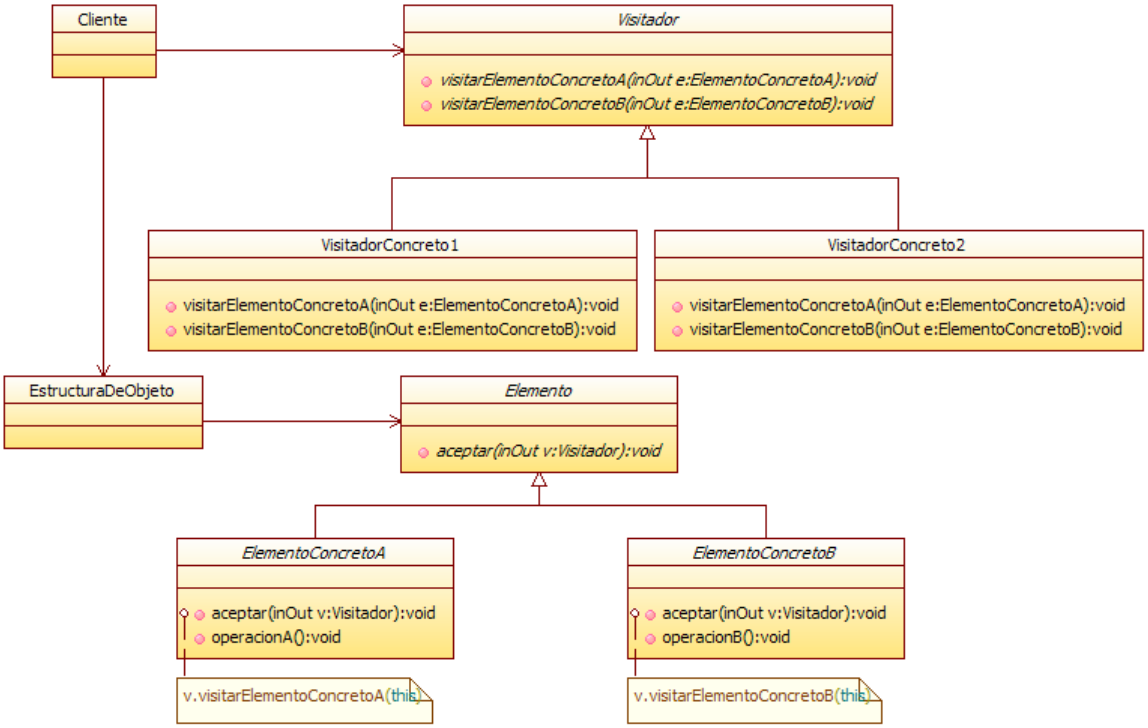


Figura 8.20 Metamodelo de Visitador

Representa una operación que se va a ejecutar en una jerarquía o estructura de objetos. La operación puede modificarse sin necesidad de modificar los objetos de la estructura o jerarquía.

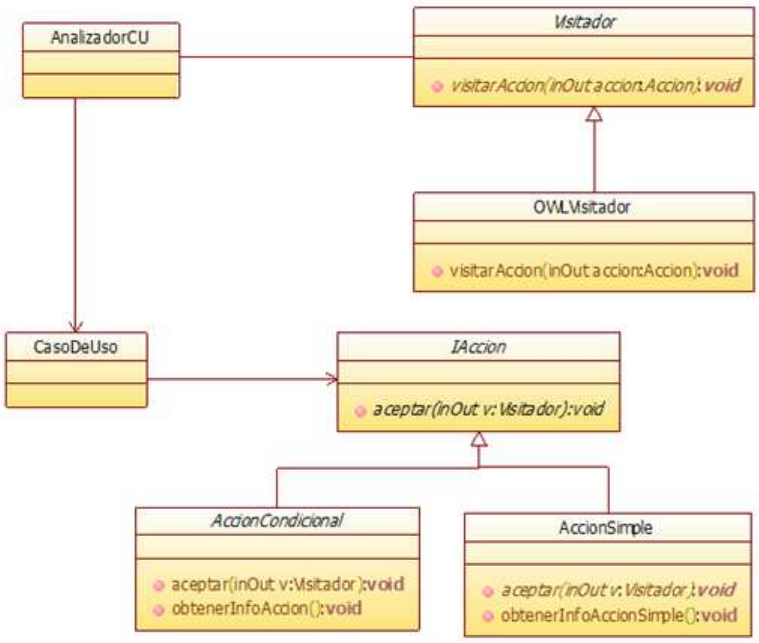


Figura 8.21 Visitador

El patrón Visitador se usa para recorrer la composición anteriormente definida que comprende las acciones o pasos de un caso de uso. Su objetivo es ir analizando acción por acción del caso de uso para extractar la información relevante para componer la Ontología y crear los individuos que representen el caso de uso.

Capítulo 9

Ontología MOD-CAS

A continuación se definen los componentes, relaciones, axiomas, patrones y mecanismos de inferencia que se usaran para definir la ontología MOD-CAS.
La metodología usada corresponde con las recomendaciones propuestas por Natalya F. Noy y Deborah L. McGuinness de la universidad de Stanford.

9.1. Dominio y alcance de la ontología MOD-CAS

La ontología MOD-CAS busca definir las especificaciones de casos de uso y servir de herramienta para desambigüar las especificaciones contribuyendo a obtener mejores definiciones de los requerimientos para los procesos de desarrollo de software.
MOD-CAS debe estar en capacidad de identificar la siguiente lista de errores en una especificación de casos de uso que fueron enunciados como parte del marco de referencia del trabajo de investigación basados en estudios de calidad de especificaciones de casos de uso:

1. **Tamaño del caso de uso:** Se puede medir contando la cantidad de pasos que un caso de uso presenta en el escenario principal o normal

2. **La complejidad de la estructura del caso de uso:** Se puede medir en términos de qué tan complejas son las acciones atómicas, o las acciones que agrupan acciones atómicas.

3. **Inconsistencias de lenguaje:** La categoría de inconsistencias habla del uso de pronombres en la redacción de un caso de uso (el, ella, nosotros...) y de mezclar el nivel de abstracción de la ECU.

4. **Descomposición del caso de uso:** El número de escenarios alternativos a considerar debe ser el necesario. ¿Hay flujos alternativos tan extensos que debieron modelarse como un caso de uso diferente?. Se abusa de la cantidad de flujos alternativos para representar flujos normales muy complejos o extensos.

5. **Elementos faltantes:** Faltan elementos de la especificación como precondiciones, postcondiciones, actores, título etc.

6. **Flujos incorrectos:** Hay errores en los pasos que se referencian en los flujos alternativos ó los flujos alternativos retornan a pasos incorrectos en el flujo principal.

7. **Ausencia de pares adyacentes:** Los pares adyacentes fueron propuestos por Phalp et al. La idea es que cada interacción propuesta en el sistema tenga una respuesta.

8. **Errores de actores no mencionados** Los errores de actores no mencionados corresponden a que en la ECU se presente una gran cantidad de pasos sin que el actor sea mencionado. ¿Existen sinónimos, homónimos, pronombres, y referencias que son usadas innecesariamente?

9. **Inconsistencias en numeración de los pasos:** ¿La numeración de todos los pasos es consistente?

10. **La condición para acceder a un flujo alternativo no es clara:** ¿La condición para acceder al flujo alternativo está claramente especificada y en el lugar correcto?

9.2. Términos clave de la ontología MOD-CAS

Como segunda parte de la metodología propuesta para el desarrollo de MOD-CAS se deben identificar los términos importantes para la ontología en una lista con algunos términos relacionados con cada termino mencionado. En los posteriores pasos se definirán y relacionan mejor estos conceptos y propiedades.

1. Caso De Uso
2. Actor, Usuario
3. Sistema
4. Objetivo
5. Secuencia de pasos
6. Variante
7. Inicio
8. Final
9. Escenario Normal
10. Escenario Alternativo
11. Precondición
12. Postcondición
13. Paso
14. Condicionales
15. Paso atómico
16. Acción
17. Recurso
18. Estado Final
19. Error
20. Tamaño
21. Cantidad de pasos
22. Complejidad

23. Estructura
24. Inconsistencia
25. Redacción
26. Par Adyacente
27. Objetivo del CU
28. Numeración
29. Condición

9.3. Definición de las clases y jerarquía de clases de la ontología MOD-CAS

Similar a el diseño orientado a objetos en OWL también se definen clases y jerarquías de clases sin embargo las clases en las ontologías OWL no están definidas en términos de funcionalidad o comportamiento. En las ontologías OWL las clases de mayor nivel representan conceptos comunes a un gran número de entidades mientras que las clases de bajo nivel en la jerarquía representan conceptos mas concretos que representan un número menor de entidades.

Los modelos de datos semánticos están mas orientados a las relaciones entre entidades que en las estructura de las entidades.

Basados en los conceptos de casos de uso y en la lista de conceptos y preguntas que debe responder la ontología MOD-CAS se plantea la siguiente jerarquía de clases:

9.3.1. Jerarquía de Clases:

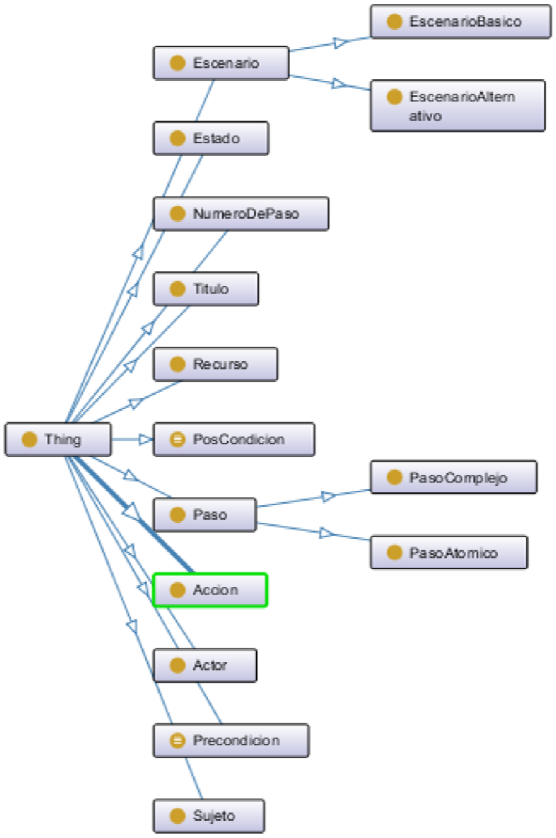


Figura 9.1 Herencia de clases de la ontología MOD-CAS

Es importante declarar un espacio de nombres para la una ontología que tendrá como función distinguir las clases y las propiedades declaradas de en la ontología de clases y propiedades con nombres iguales en ontologías distintas con las que se puede interactuar en el futuro.

http://www.modcas.com.co/modcas

En adelante para abreviar la escritura de los recursos de la ontología se usara el espacio de nombres resumido *modcas*.

Las Clases declaradas para la ontología en lenguaje OWL se presentan a continuación:

```
modcas:Accion rdf:type :Class ;
modcas:Actor rdf:type :Class ;
modcas:Escenario rdf:type :Class ;
modcas:EscenarioAlternativo rdf:type :Class ;
                    rdfs:subClassOf modcas:Escenario ;
modcas:EscenarioBasico rdf:type :Class ;
                  rdfs:subClassOf modcas:Escenario ;
modcas:Estado rdf:type :Class .
modcas:NumeroDePaso rdf:type :Class ;
modcas:Paso rdf:type :Class ;
modcas:PasoAtomico rdf:type :Class ;
              rdfs:subClassOf modcas:Paso ;
modcas:PasoComplejo rdf:type :Class ;
               rdfs:subClassOf modcas:Paso ;
modcas:PosCondicion rdf:type :Class ;
               :equivalentClass [ rdf:type :Restriction ;
                             :onProperty modcas:estadoAlcanzado ;
                             :onClass modcas:Estado ;
                             :minQualifiedCardinality
                                "1"^^xsd:nonNegativeInteger
                           ] ;
modcas:Precondicion rdf:type :Class ;
               :equivalentClass [ rdf:type :Restriction ;
                             :onProperty modcas:estadoRequerido ;
                             :onClass modcas:Estado ;
                             :minQualifiedCardinality
                                "1"^^xsd:nonNegativeInteger
                           ] ;
modcas:Recurso rdf:type :Class ;
modcas:Sujeto rdf:type :Class ;
modcas:Titulo rdf:type :Class ;
```

9.4. Definición de las propiedades y relaciones de la ontología MOD-CAS

El siguiente paso en la metodología propuesta para el desarrollo de ontologías consta en definir las propiedades y las características de las propiedades de la ontología.

En los modelos semánticos las propiedades se definen independientemente de las clases, aunque se puede declarar que recursos o clases usan estas propiedades. Una propiedad puede estar relacionada con multiples clases pero existen dos formas principales para relacionar propiedades con clases que van a tener consecuencias importantes sobre el comportamiento de la propiedad dentro de la ontología:

- Dominio: El dominio de una propiedad hace referencia al sujeto de una tripleta en al que se usa la propiedad.
- Rango: El rango de una propiedad hace referencia al objeto de una tripleta en la que se usa la propiedad.

A continuación se enuncian las propiedades básicas para la ontología MOD-CAS

```
modcas:afectaA rdf:type :ObjectProperty ;
           rdfs:domain modcas:Accion ;
           rdfs:range modcas:Recurso .
modcas:dependeDe rdf:type :ObjectProperty ;
             rdfs:subPropertyOf modcas:tienePrerequisito .
modcas:ejecuta rdf:type :ObjectProperty ;
           rdfs:range modcas:Accion ;
           rdfs:domain modcas:Sujeto .
modcas:esPrerequisitoPara rdf:type :ObjectProperty ,
                          :TransitiveProperty ;
                    rdfs:subPropertyOf modcas:pasoRelacionado .
```

```
modcas:estadoAlcanzado rdf:type :ObjectProperty ;
                  rdfs:range modcas:Estado ;
                  rdfs:domain modcas:Recurso .
modcas:estadoCondicion rdf:type :ObjectProperty ;
                  rdfs:domain modcas:EscenarioAlternativo ;
                  rdfs:range modcas:Estado .
modcas:estadoRequerido rdf:type :ObjectProperty ;
                  rdfs:range modcas:Estado ;
                  rdfs:domain modcas:Recurso .
modcas:numeradoCon rdf:type :FunctionalProperty ,
                      :InverseFunctionalProperty ,
                      :ObjectProperty ;
               rdfs:range modcas:NumeroDePaso ;
               rdfs:domain modcas:Paso .
modcas:pasoRelacionado rdf:type :ObjectProperty ;
modcas:permite rdf:type :ObjectProperty ;
           rdfs:subPropertyOf modcas:esPrerequisitoPara .
modcas:pertenece rdf:type :ObjectProperty ;
             rdfs:range modcas:Escenario ;
             rdfs:domain modcas:Paso .
modcas:recursoCondicion rdf:type :ObjectProperty ;
                  rdfs:domain modcas:EscenarioAlternativo ;
                  rdfs:range modcas:Recurso ;
                  rdfs:subPropertyOf :topObjectProperty .
modcas:seRealizaAccion rdf:type :ObjectProperty ;
                  rdfs:range modcas:Accion ;
                  rdfs:domain modcas:Paso .
modcas:tienePrerequisito rdf:type :ObjectProperty ,
                         :TransitiveProperty ;
                   rdfs:subPropertyOf modcas:pasoRelacionado .
modcas:tienenTitulo rdf:type :ObjectProperty ;
```

9.5. Modelos de la Ontología MOD-CAS para identificar los errores propuestos

Como se ha expuesto anteriormente la Ontología debe ser capaz de identificar la lista de errores que conforman el alcance del modelo MOD-CAS.

Para conseguir identificar los errores y aportar información al requerimiento especificado hace falta usar herramientas adicionales de las ontologías OWL. Existen otros recursos en OWL ademas de las propiedades con los que podemos establecer relaciones entre las clases de una ontología con el fin de aportar mecanismos para inferir información.

RDFS y OWL definen una serie de propiedades con comportamiento especial que se pueden trabajar en conjunto para definir relaciones mas complejas entre recursos.

A continuación se exponen las estructuras adicionales para identificar cada uno de los errores propuestos en el alcance de la ontología.

9.5.1. Tamaño del caso de uso

En la bibliografía es complejo encontrar mediciones y recomendaciones del tamaño de un caso de uso. En el libro Writing Effective Use Cases se encuentra un test de validacion para especificaciones de casos de uso, en este test se habla que el tamaño de un escenario principal debe tener entre 3 y 9 pasos. Con base a esta recomendación se establecera el tamaño del escenario principal que la Ontología evaluara.
La ontología MOD-CAS debe estar en capacidad de medir si un caso de uso tiene menos de 3 pasos o mas de 9 pasos es decir si esta por fuera del tamaño recomendado para el escenario principal.

Para contar el numero de pasos se va a crear una nueva tripleta construida a partir de una consulta en lenguaje SPARQL versión 1.1 que incluye operaciones de agrupación similares al SQL.

```
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT (count (?paso) as ?numeroPasos)
WHERE {
?paso rdf:type modcas:Paso .
?paso modcas:pertenece modcas:escenario_Basico . }
```

El resultado de la ejecución de la anterior consulta SPARQL contara los pasos que pertenecen al escenario principal o basico del caso de uso.

9.5.2. La complejidad de la estructura del caso de uso

Se puede medir en términos de qué tan complejas son las acciones atómicas, o las acciones que agrupan acciones atómicas.

Las acciones complejas se identifican como pasos que comprenden mas de una acción que debe ejecutar el actor o el sistema.

Anteriormente se había declarado las relaciones y las clases para modelar los pasos y subtipos de pasos de un caso de uso que la ontología va manejar, ahora se debe modelar como se va a componer cada uno de estos pasos y en caso de ser un paso de un caso de uso que implique mas de una acción como se va a modelar este comportamiento.

```
modcas:Paso rdf:type owl:Class .
   modcas:PasoComplejo rdfs:subClassOf modcas:Paso .

modcas:realizaAccion rdf:type rdf:Property .
   modcas:realizaAccion rdfs:domain modcas:Paso .
   modcas:realizaAccion rdfs:range modcas:Accion .
```

Un paso complejo relaciona varios individuos de tipo *modcas:Accion* con un individuo de tipo *modcas:Paso*. Mientras que un paso atomico solo relaciona un individuo de tipo *modcas:Accion* con un individuo de tipo *modcas:Paso*.

Podemos entonces identificar si un caso de uso tiene una complejidad alta cuantos mas individuos de tipo *modcas:PasoComplejo* se presenten respecto a *modcas:PasoAtomico*, de nuevo se recurre a realizar una cuenta de cada uno de los tipos de pasos para poder establecer una proporción de los dos tipos de pasos del caso de uso.

```
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT (count (?paso) as ?numeroPasosComplejos)
WHERE { ?paso rdf:type modcas:PasoComplejo . }


PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT (count (?paso) as ?numeroPasosAtomicos)
WHERE { ?paso rdf:type modcas:PasoAtomico .}
```

La relación que se establece entre estos dos valores (C) indica que tan complejo es el caso de uso:

$$C = \frac{pasosAtomicos}{pasosComplejos}$$

- C menor que uno: Existen mas pasos complejos que pasos atómicos en el caso de uso, el caso de uso puede aparentar tener menos pasos pero la complejidad de cada uno de estos puede ser grande ya que se requiere ejecutar mas acciones para ir de un paso a otro.
- C igual a uno: Existen igual cantidad de pasos atómicos y pasos complejos, si bien la cantidad de pasos complejos no sobrerpasa a los atómicos se puede indicar que la mitad de los pasos del caso de uso son complejos y puede ser difícil de implementar.

- C mayor a uno: El caso de uso fue diseñado para tener acciones sencillas de ejecutar en cada paso, esta seria la proposición ideal a encontrar.

9.5.3. Inconsistencias de lenguaje

En cada paso del caso de uso se debe identificar claramente un sujeto que ejecute la acción del paso, la acción que se desea ejecutar y el recurso sobre el que se va a ejecutar dicha acción.

Se propone que la ontología pueda identificar si alguno de los pasos del caso de uso esta incompleto porque le falte actor, acción o recurso.

```
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?sujeto ?accion ?recurso
   WHERE { ?sujeto modcas:ejecuta ?accion .
   ?accion modcas:afectaA ?recurso .}
```

9.5.4. Descomposición del caso de uso

Igual que se midió la proporción entre las acciones complejas y las acciones atómicas la cantidad de escenarios alternativos y la cantidad de pasos que estos flujos tienen respecto a la cantidad de pasos del escenario básico del caso de uso.

```
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?escenarioAlternativo (count (?paso) as ?pasosXEscenario)
   WHERE { ?paso modcas:pertenece ?escenarioAlternativo .
      ?escenarioAlternativo rdf:type modcas:EscenarioAlternativo . }
GROUP BY ?escenarioAlternativo
```

La anterior consulta SPARQL retorna cuantos pasos tiene cada uno de los escenarios alternativos de esta manera se puede realizar la comparación con el tamaño del caso de uso en su escenario básico.

La comparación puede dar idea si existen escenarios alternativos tan grandes que posiblemente debieron ser modelados como un caso de uso independiente o si hay tantos escenarios alternativos que se puede estar abusando de la figura de escenario alternativo queriendo controlar mas de lo que se debe las acciones del caso de uso.

9.5.5. Elementos faltantes

Una vez que el caso de uso es modelado en la ontología modcas se puede establecer que cuales de los elementos del caso de uso faltan en el modelo, sea por consultas o usando validaciones de cardinalidad que OWL ofrece.

Para modelar las precondiciones y las poscondiciones, se establece que para la ontología MOD-CAS debe existir al menos una precondicion y una poscondicion para ser valida. Las clases de Precondición y Poscondición se representan como el conjunto de individuos que contienen las condiciones para iniciar la ejecución o para establecer como exitosa la ejecución del caso de uso así:

```
modcas:PosCondicion rdf:type :Class ;
                :equivalentClass [ rdf:type :Restriction ;
                                   :onProperty modcas:estadoAlcanzado ;
                                   :onClass modcas:Estado ;
                                   :minQualifiedCardinality
                                      "1"^^xsd:nonNegativeInteger
                                 ] ;
modcas:Precondicion rdf:type :Class ;
                :equivalentClass [ rdf:type :Restriction ;
                                   :onProperty modcas:estadoRequerido ;
                                   :onClass modcas:Estado ;
                                   :minQualifiedCardinality
                                      "1"^^xsd:nonNegativeInteger
                                 ] ;
```

9.5.6. Flujos incorrectos

El objetivo del siguiente diseño es verificar que los flujos básicos y alternativos sean consistentes, es decir que los pasos del caso de uso estén bien relacionados.

Para lograr este objetivo se debe modelar una estructura que permita simular las relaciones que se establecen entre los pasos de un caso de uso y verificar si estos pasos cumplen con estas relaciones.

Dos pasos de un caso de uso pueden establecer varias relaciones entre sí, un paso puede depender de que se haya ejecutado paso anterior o si se mira la relación inversa se puede decir que un paso es prerequisito para la ejecución de otro paso.

Por medio de este modelo se puede obtener todas las secuencias de pasos de un caso de uso esto también permite conocer si existen pasos del caso de uso que no se pueden alcanzar, es decir, pasos de escenarios alternativos que no son invocados desde ningun paso del escenario básico.

```
modcas:dependeDe rdfs:subPropertyOf modcas:TienePrerequisito .
modcas:TienePrerequisito rdf:type owl:TransitiveProperty .
modcas:permite rdfs:subPropertyOf modcas:EsPrerequisitoPara .
modcas:EsPrerequisitoPara rdf:type owl:TransitiveProperty .
modcas:TienePrerequisito rdfs:subPropertyOf modcas:pasoRelacionado .
modcas:EsPrerequisitoPara rdfs:subPropertyOf modcas:pasoRelacionado .
```

9.5.7. Ausencia de pares adyacentes

Para identificar los pares adyacentes se necesita conocer quien esta ejecutando cada paso del caso de uso de esta forma se puede ver si existen series de pasos en las que esta recomendación de diseño no se cumple.

Se lista la secuencia de pasos del caso de uso y se relaciona con el sujeto identificado en cada paso, lo que permite identificar secuencias largas donde no se cumple la condicion de pares adyacentes.

```
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?numPaso ?sujeto
   WHERE {
            ?sujeto modcas:ejecuta ?accion .
            ?paso modcas:seRealizaAccion ?accion .
            ?paso modcas:pertenece ?escenarioBasico .
            ?escenarioBasico rdf:type modcas:EscenarioBasico .
            ?paso modcas:numeradoCon ?numPaso .}
ORDER BY ASC(?numPaso)
```

9.5.8. Errores de actores no mencionados

De la misma manera que se identificaron los pares adyacentes los actores no mencionados se pueden identificar. No es necesario nombrar en cada paso el actor que lo va a ejecutar, pero no se recomienda tener una serie larga de pasos en los que el actor no aparezca.

```
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?numPaso ?sujeto ?accion ?recurso
   WHERE { ?sujeto modcas:ejecuta ?accion .
               ?accion modcas:afectaA ?recurso .
               ?paso modcas:numeradoCon ?numPaso .
               ?paso modcas:seRealizaAccion ?accion .}
ORDER BY ASC(?numPaso)
```

9.5.9. Inconsistencias en numeración de los pasos

Uno de los errores que se pretenden identificar con la ontología se produce cuando hay ambigüedad en la numeración de los pasos del caso de uso. En general cada paso de un caso de uso debe tener un numero unico asignado.

Para lograr validar esta característica se plantean las siguientes propiedades y tripletas que le permitirán a la ontología identificar si existen pasos con el mismo numero en una especificación de caso de uso.
Las propiedades que permiten a la ontología relacionar un unico recurso con otro son:

- **owl:FunctionalProperty:** Permite un unico valor como objeto de una tripleta
- **owl:InverseFunctionalProperty:** Permite un unico valor como sujeto de una tripleta

Anteriormente se había definido la propiedad: modcas:numeradoCon con las siguientes características:

```
modcas:numeradoCon rdfs:domain modcas:Paso .
modcas:numeradoCon rdfs:range modcas:NumeroDePaso .
```

Se usa una combinación de las propiedades de OWL owl:FunctionalProperty y
owl:InverseFunctionalProperty así:

```
modcas:numeradoCon rdf:type owl:FunctionalProperty .
modcas:numeradoCon rdf:type owl:InverseFunctionalProperty .
```

La anterior combinación de las propiedades indica en el modelo ontológico que solo puede existir un individuo de tipo modcas:Paso relacionado por medio de modcas:numeradoCon con un individuo de tipo modcas:NumeroDePaso en todo el modelo.

La ontología no impide que se declare mas de un individuo de tipo modcas:Paso relacionado con modcas:NumeroDePaso por medio de la propiedad modcas:numeradoCon, pero esta relación obliga a la ontología a inferir que dos pasos son iguales generando la siguiente tripleta como resultado de la inferencia:

```
   modcas:Paso owl:sameAs modcas:Paso .
```

Posteriormente como validación del caso de uso se puede consultar en un caso de uso existen pasos que cumplan ser el mismo owl:sameAs lo que no debe ocurrir en ningun caso.

9.5.10. La condición para acceder a un flujo alternativo no es clara

Un paso de un caso de uso que tenga asociado un escenario alternativo debe especificar claramente en que condición se debe ejecutar el escenario alternativo. Se puede relacionar la condición de entrada al escenario alternativo con un estado especifico en el que se encontró un recurso al momento de ejecutar un paso. Para obtener la información de la ontología sobre la claridad de las condiciones para acceder a un caso de uso se plantea la siguente consulta SPARQL:

```
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?escenarioAlternativo ?recurso ?estado
   WHERE {?escenarioAlternativo rdf:type modcas:EscenarioAlternativo .
      ?escenarioAlternativo modcas:recursoCondicion ?recurso .
      ?escenarioAlternativo modcas:estadoCondicion ?estado .
   }
```

9.6. Análisis de Casos de Uso usando la Ontología MOD-CAS

A continuación se analiza una especificación de un caso de uso por medio del modelo ontológico planteado en la sección anterior, la especificación completa del caso de uso y el modelado completo del caso de uso en la ontología se encuentra en los anexos.

El caso de uso que se analiza a continuación tiene como objetivo permitir a un usuario comprador adquirir un título valor en un sitio web registrando los datos del título comprado en un portafolio que debe tener inscrito dicho comprador.

9.6.1. Tamaño del caso de uso

Para comprobar el tamaño del caso de uso segun la validación propuesta se ejecuta en el modelo la consulta para el caso de uso de estudio. La imagen muestra los pasos que se encuentran modelados en la ontología para el caso de uso de estudio CU001.

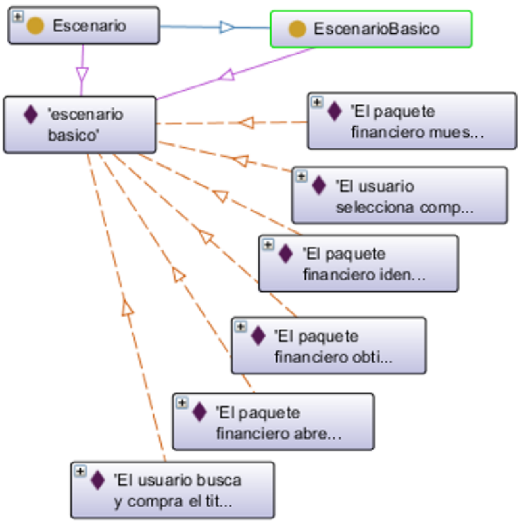


Figura 9.2 Pasos del escenario básico del CU001 Fuente: [Autor]

Al ejecutar la consulta se obtienen como resultado que el caso de uso tiene 6 pasos, por lo tanto no supera el numero de pasos sugerido en la bibliografía como lo muestra la figura.

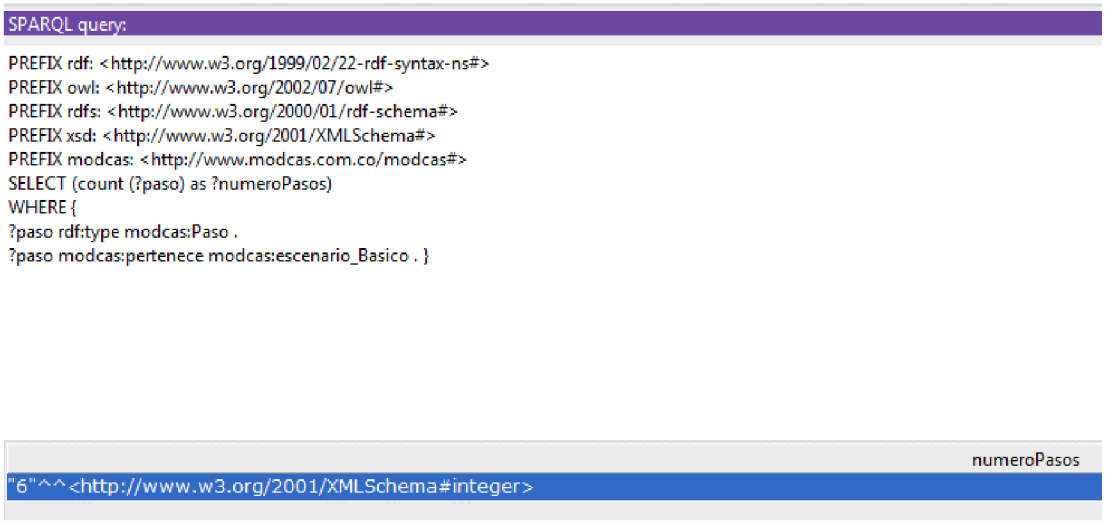


Figura 9.3 Total de Pasos del escenario básico del CU001. Fuente: [Autor]

9.6.2. La complejidad de la estructura del caso de uso

Para analizar la complejidad del caso de uso como lo propone el modelo MOD-CAS se debe identificar cuantos pasos atómicos, pasos que solo ejecutan una acción, y cuantos pasos complejos, pasos que ejecutan mas de una acción, para hallar la proporción existente entre estos dos tipos de pasos.

Las siguientes figuras muestran los pasos atómicos del caso de uso y los pasos complejos del caso de uso:

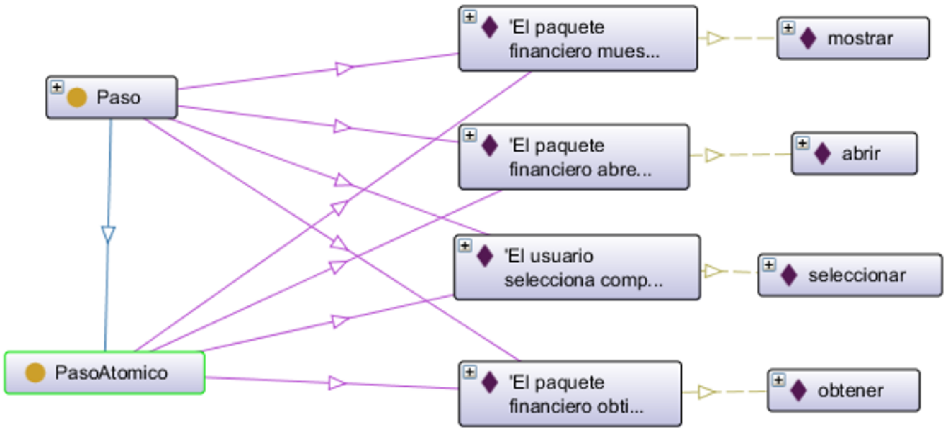


Figura 9.4 Pasos con una sola acción en el CU001. Fuente: [Autor]

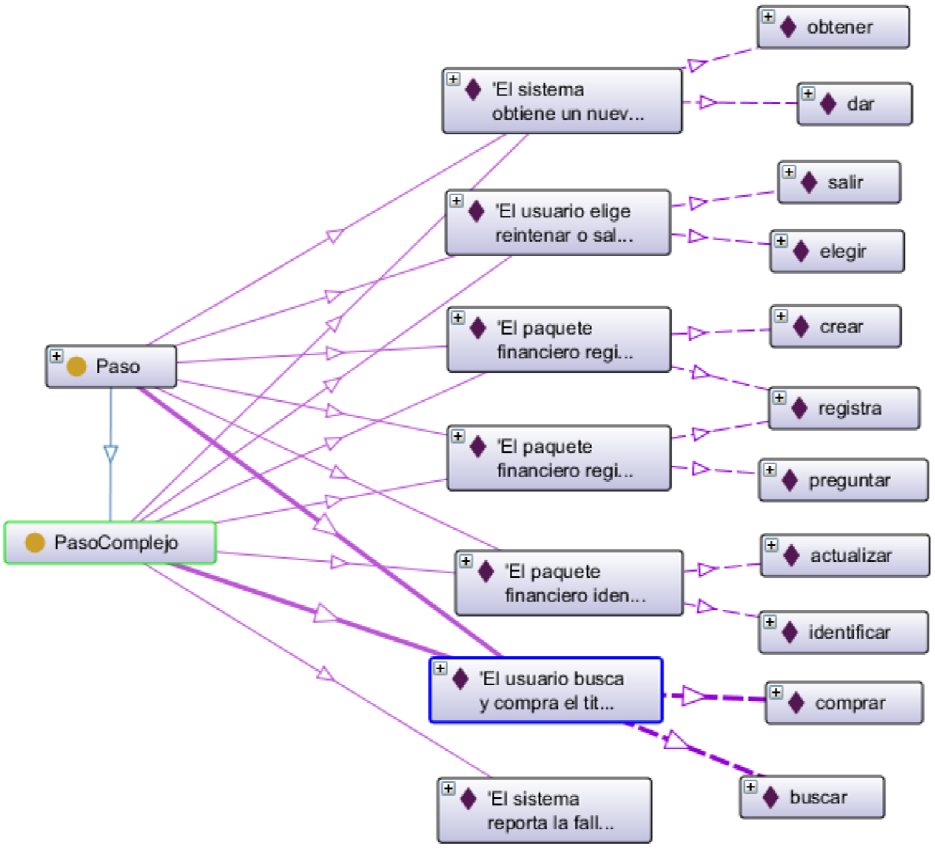


Figura 9.5 Pasos con mas de una acción en el CU001. Fuente: [Autor]

Al ejecutar las consultas propuestas el modelo arroja la cantidad de pasos complejos y atómicos que permite establecer la proporción:

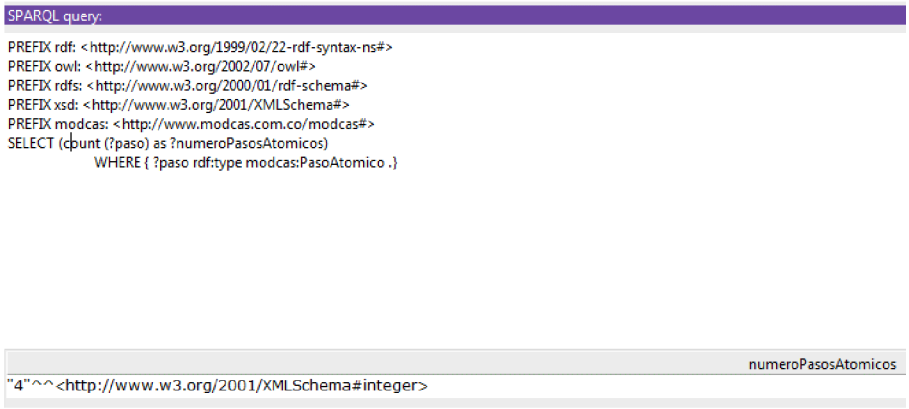


Figura 9.6 Cantidad de pasos con una sola acción en el CU001 Fuente: [Autor]

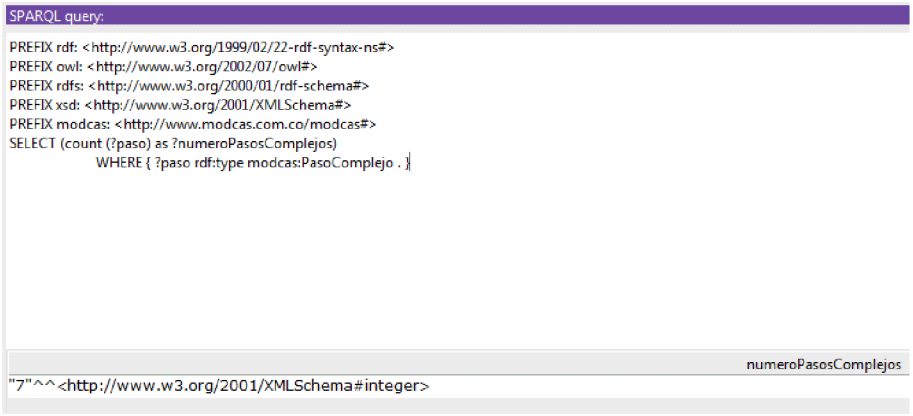


Figura 9.7 Cantidad de pasos con mas de una acción en el CU001. Fuente: [Autor]

El caso de uso tiene 4 pasos atómicos y 7 pasos complejos, la proporción entonces:

$$C = \frac{pasosAtomicos}{pasosComplejos} = \frac{4}{7} = 0,57$$

En conclusión el caso de uso tiene mayor cantidad de pasos complejos, lo que hace que el caso de uso pueda aparentar tener pocos pasos, pero se deben ejecutar varias acciones en los pasos para alcanzar el objetivo del caso de uso.

9.6.3. Inconsistencias de lenguaje

La validación consiste en poder identificar claramente para cada paso del caso de uso quien ejecuta la acción, que acción se ejecuta y sobre que se ejecuta la acción.

Tomando como ejemplo el paso 6 del flujo básico del CU001 que se esta analizando dice: *"6. El paquete financiero muestra el portafolio del usuario actualizado"* se modela en la ontología asi:

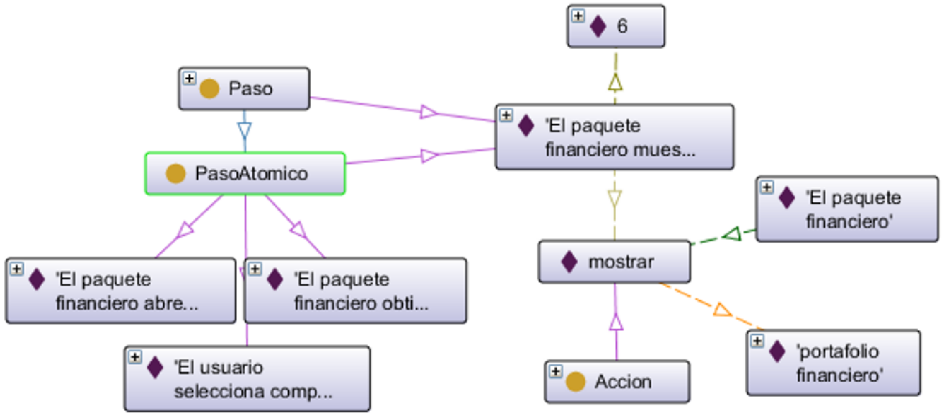


Figura 9.8 Paso 6 del CU001 modelado en la ontolgía MOD-CAS Fuente: [Autor]

En la figura se muestra el paso mencionado que se descompone en su acción que es *mostrar*, quien ejecuta la acción que es *El paquete financiero* y que se debe mostrar: *el portafolio del usuario actualizado*

Para comprobar que todos los pasos del caso de uso tienen los elementos de lenguaje requeridos ejecutamos la consulta propuesta y se espera que en el resultado no falte para ningun paso alguno de los elementos: acción, sujeto, predicado.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?sujeto ?accion ?recurso
WHERE { ?sujeto modcas:ejecuta ?accion .
?accion modcas:afectaA ?recurso .}
```

sujeto	accion	recurso
'El paquete financiero'	obtener	'un nuevo nombre de sitio web del usuario'
'El paquete financiero'	obtener	'nombre del sitio web de compra de titulos seleccionado por el usuario'
'El Usuario'	elegir	'salir del caso de uso'
'El Usuario'	elegir	reintenar
'El Usuario'	buscar	'el titulo en el sitio web'
'El paquete financiero'	mostrar	'portafolio financiero'
'El paquete financiero'	abrir	'la conexión al sitio web seleccionado'
'El Usuario'	comprar	'el titulo en el sitio web'
'El Usuario'	salir	'del caso de uso'
'El Usuario'	registra	'la falta de información'
'El Usuario'	registra	'la transacción en espera'
'El sistema'	retornar	'al paso anterior'
'El paquete financiero'	identificar	'el titulo comprado'

Figura 9.9 Resultado de la composición de los pasos del caso de uso CU001. Fuente: [Autor]

Como resultado se puede apreciar que no hay pasos en los que no se puedan identificar los elementos de los que se compone cada paso para el caso de uso de estudio.

9.6.4. Descomposición del caso de uso

La siguiente validación consta de identificar el tamaño de los escenarios alternativos respecto al tamaño del escenario básico, la idea es identificar si hay flujos alternativos mas complejos o mas extensos que el flujo básico. Para esto se ejecuta la consulta propuesta:

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?escenarioAlternativo (count (?paso) as ?pasosXEscenario)
WHERE { ?paso modcas:pertenece ?escenarioAlternativo .
?escenarioAlternativo rdf:type modcas:EscenarioAlternativo . }
GROUP BY ?escenarioAlternativo
```

escenarioAlternativo	pasosXEscenario
'El usuario desea ingresar a un sitio que el paquete financiero no soporta'	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
'Se produce un fallo en el sitio web de compra de titulos.'	"2"^^<http://www.w3.org/2001/XMLSchema#integer>
'El sitio web no retorno la información necesaria para registrar la compra'	"1"^^<http://www.w3.org/2001/XMLSchema#integer>
'El sitio web no acepta inmediatamente la compra, pero la pone en espera'	"1"^^<http://www.w3.org/2001/XMLSchema#integer>

Figura 9.10 Número de pasos de cada escenario alternativo del caso de uso CU001. Fuente: [Autor]

La consulta permite conocer que no hay escenarios alternativos con un gran número de pasos, y que tampoco superan el número de pasos del escenario básico en este sentido el caso de uso de estudio esta bien proporcionado.

9.6.5. Elementos faltantes

Para el análisis de los elementos faltantes adicional a las consultas ya expresadas que muestran los actores, escenarios y demas elementos, se realiza la validación de que existan las precondicones y las poscondiciones.

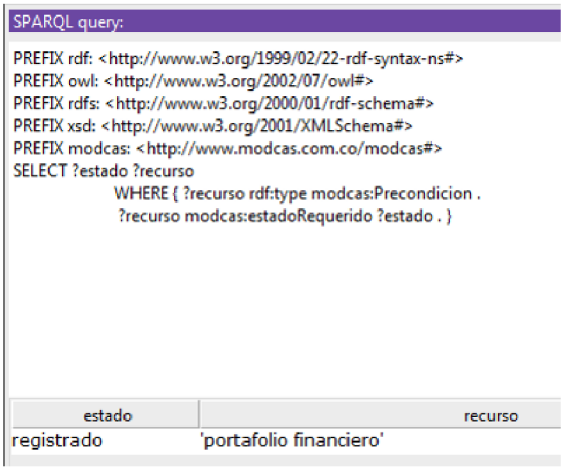


Figura 9.11 Precondiciones del caso de uso CU001. Fuente: [Autor]

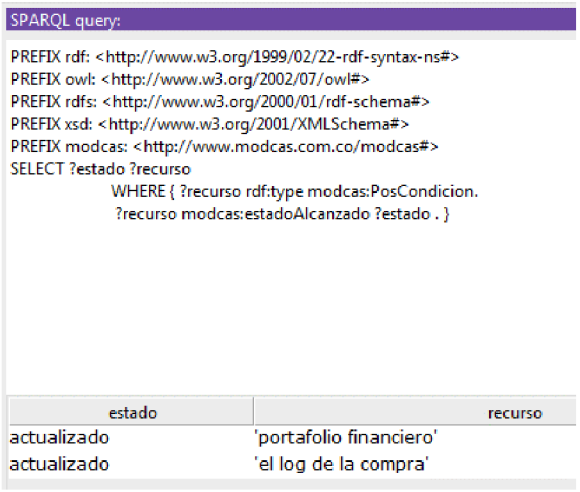


Figura 9.12 Poscondiciones del caso de uso CU001. Fuente: [Autor]

En ambos casos los componentes de una precondición o una poscondición son un recurso y un estado que se puedan identificar.

Con el animo de aportar información a la especificación del caso de uso se plantea una consulta que permite conocer cuales son todos los recursos que se van a utilizar en el caso de uso, esto permite identificar posibles precondiciones que no hayan sido enunciadas asi:

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?recurso
WHERE {?recurso rdf:type modcas:Recurso .
}
```

recurso
'la conexión al sitio web seleccionado'
'un temporizador para registrar la compra posteriormente'
'portafolio financiero'
'salir del caso de uso'
'la falla al usuario'
'el titulo comprado'
'la opcion de cancelar la ejecución del caso de uso'
'la compra'
'el sitio web de compra de titulos'
'la información necesaria para registrar la compra'
'al paso anterior'
reintenar

Figura 9.13 Recursos utilizados en el caso de uso CU001. Fuente: [Autor]

9.6.6. Flujos incorrectos

La estructura definida para relacionar los pasos de un caso de uso en la ontología MOD-CAS nos permite realizar multiples consultas para conocer la consistencia de las relaciones entre los pasos.

Como primera medida se puede averiguar cual es el paso anterior de cualquier paso en el caso de uso, es decir de quien depende para ejecutarse,

tomando como ejemplo el último paso del flujo básico del caso de uso *"6. El paquete financiero muestra el portafolio del usuario actualizado"*:

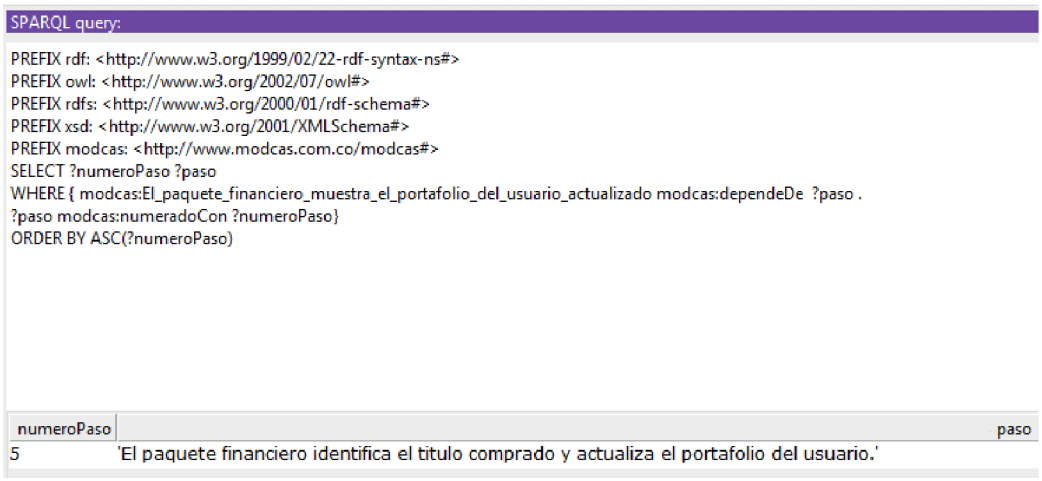


Figura 9.14 Dependencia entre pasos. Fuente: [Autor]

Pero el modelo también permite conocer cuales son todos los pasos que se deben ejecutar antes de llegar a un paso determinado, tomando como ejemplo el mismo paso se tiene:

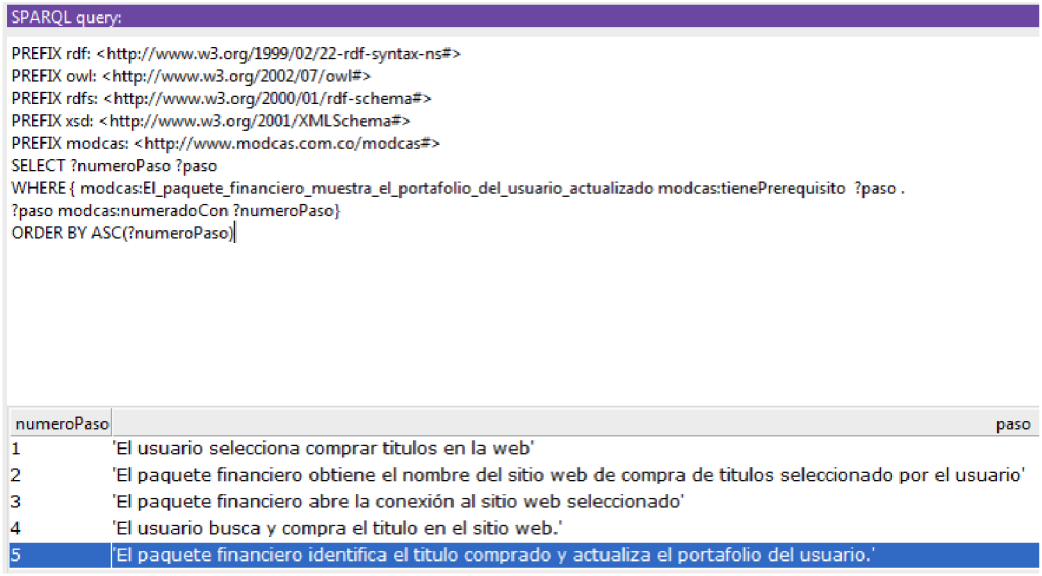


Figura 9.15 Prerequisitos para alcanzar un paso. Fuente: [Autor]

Esta característica de la ontología es importante porque permite conocer si algun paso no esta relacionado con otros, especialmente en los escenarios alternativos, por ejemplo el escenario alternativo *4.a. El sitio web no acepta inmediatamente la compra, pero la pone en espera* ejecuta el paso *4.a.1 El paquete financiero registra la transacción en espera, crea un temporizador para registrar la compra posteriormente.* los prerrequisitos de este paso para el escenario alternativo son:

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT DISTINCT ?numPaso ?paso
WHERE { modcas:El_paquete_financiero_registra_la_transacción_en_espera_crea_un_temporizador_para_registrar_la_compra_posteriormente
modcas:tienePrerequisito ?paso .
?paso modcas:numeradoCon ?numPaso .
?paso rdf:type modcas:Paso .}
ORDER BY ASC(?numPaso)
```

numPaso	paso
1	'El usuario selecciona comprar titulos en la web'
2	'El paquete financiero obtiene el nombre del sitio web de compra de titulos seleccionado por el usuario'
3	'El paquete financiero abre la conexión al sitio web seleccionado'
4	'El usuario busca y compra el titulo en el sitio web.'

Figura 9.16 Prerequisitos para alcanzar un paso del escenario alternativo 4a. Fuente: [Autor]

La figura muestra que para ejecutar el paso 4.a.1 del escenario alternativo 4 debieron ejecutarse primero los pasos 1 al 4 del flujo básico.

9.6.7. Ausencia de pares adyacentes

Los pares adyacentes son una característica importante para los casos de uso, permiten reconocer que para cada interacción de un usuario existe una respuesta del sistema. Para validar este aspecto del caso de uso se plantea ejecuta la consulta propuesta:

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?numPaso ?sujeto
WHERE {
        ?sujeto modcas:ejecuta ?accion .
        ?paso modcas:seRealizaAccion ?accion .
        ?paso modcas:pertenece ?escenarioBasico .
        ?escenarioBasico rdf:type modcas:EscenarioBasico .
        ?paso modcas:numeradoCon ?numPaso .}
ORDER BY ASC(?numPaso)
```

numPaso	sujeto
1	'El Usuario'
2	'El paquete financiero'
3	'El paquete financiero'
4	'El Usuario'
4	'El Usuario'
5	'El paquete financiero'
6	'El paquete financiero'

Figura 9.17 Actores que ejecutan cada paso del flujo básico. Fuente: [Autor]

La consulta muestra que a travez de los pasos del caso de uso que se establece una interacción para cada acción ejecutada por el usuario, en algunos casos es mas de una acción pero no hay un número elevado de acciones por parte del usuario o el sistema en las que no halla interaccón.

9.6.8. Errores de actores no mencionados

En el caso de uso de estudio podemos identificar claramente quien ejecuta cada uno los pasos y las acciones dentro de los pasos, ya sean pasos del escenario básico o de los escenarios alternativos.

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT DISTINCT ?numPaso ?sujeto ?accion ?recurso
WHERE { ?sujeto modcas:ejecuta ?accion .
?accion modcas:afectaA ?recurso .
?paso modcas:numeradoCon ?numPaso .
?paso modcas:seRealizaAccion ?accion .}
ORDER BY ASC(?numPaso)
```

numPaso	sujeto	accion	recurso
1	'El Usuario'	seleccionar	'Comprar titulos en la Web'
2	'El paquete financiero'	obtener1	'nombre del sitio web de compra de titulos seleccionado por el usuario'
2a1	'El sistema'	dar	'la opcion de cancelar la ejecución del caso de uso'
2a1	'El paquete financiero'	obtener	'un nuevo nombre de sitio web del usuario'
3	'El paquete financiero'	abrir	'la conexión al sitio web seleccionado'
3a1	'El sistema'	reportar	'la falla al usuario'
3a1	'El sistema'	retornar	'al paso anterior'
3a2	'El Usuario'	elegir1	reintenar
3a2	'El Usuario'	salir	'del caso de uso'
4	'El Usuario'	buscar	'el titulo en el sitio web'
4	'El Usuario'	comprar	'el titulo en el sitio web'
4a1	'El paquete financiero'	registra	'la transacción en espera'
4a1	'El paquete financiero'	crear	'un temporizador para registrar la compra posteriormente'
5	'El paquete financiero'	actualizar	'portafolio financiero'
5	'El paquete financiero'	identificar	'el titulo comprado'
5a1	'El Usuario'	registrar	'la falta de información'
6	'El paquete financiero'	mostrar	'portafolio financiero'

Figura 9.18 Secuencia de pasos indicando actores. Fuente: [Autor]

Adicionalmente el caso de uso enuncia que el actor es *El Comprador* sin embargo no se hace mención en los pasos a el comprador, en cambio en los pasos se nombra a quien realiza las acciones como *El usuario*, la ontología permite establecer que El Comprador y El Usuario son la misma persona por medio de la propiedad *owl:sameAs* de forma que se forma la tripleta:

El Comprador owl:SameAs El usuario

Al consultar de nuevo los pasos del caso de uso, se puede observar que al decir que El Comprador y El Usuario son el mismo individuo, todas

las acciones ejecutadas dentro del CU se pueden consultar como acciones ejecutadas ya no por El Usuario sino por El Comprador que es originalmente el Actor enunciado por el caso de uso como se muestra en la figura:

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?numPaso ?actor  ?accion ?recurso
WHERE { ?actor modcas:ejecuta ?accion .
?actor rdf:type modcas:Actor .
?accion modcas:afectaA ?recurso .
?paso modcas:numeradoCon ?numPaso .
?paso modcas:seRealizaAccion ?accion .}
ORDER BY ASC(?numPaso)
```

numPaso	actor	accion	recurso
1	'El Usuario'	seleccionar	'Comprar titulos en la Web'
1	Comprador	seleccionar	'Comprar titulos en la Web'
3a2	'El Usuario'	salir	'del caso de uso'
3a2	'El Usuario'	elegir1	reintenar
3a2	Comprador	salir	'del caso de uso'
3a2	Comprador	elegir1	reintenar
4	'El Usuario'	buscar	'el titulo en el sitio web'
4	'El Usuario'	comprar	'el titulo en el sitio web'
4	Comprador	comprar	'el titulo en el sitio web'
4	Comprador	buscar	'el titulo en el sitio web'
5a1	'El Usuario'	registrar	'la falta de información'
5a1	Comprador	registrar	'la falta de información'

Figura 9.19 Pasos ejecutados por el Actor: El Comprador. Fuente: [Autor]

Otra de las características que aporta la ontología MOD-CAS es la capacidad de encontrar nuevos actores que quizas no han sido mencionados en la sección de actores de la especificación del caso de uso. Al realizar simplemente una consulta de quienes son los sujetos dentro del caso de uso que realizan las acciones se pueden hallar nuevos posibles actores.

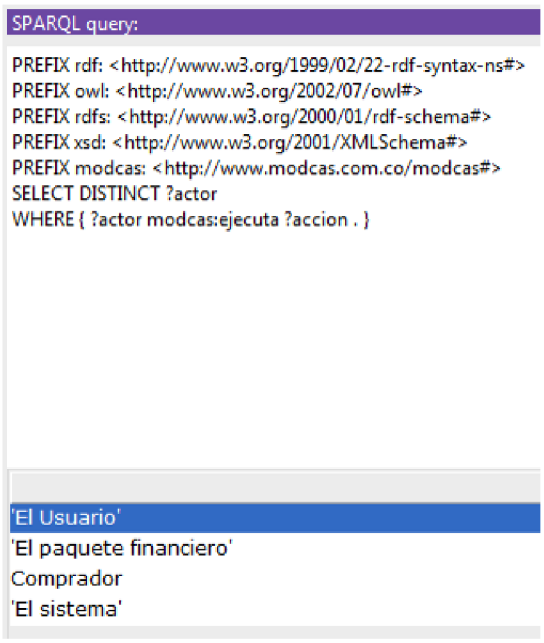


Figura 9.20 Consulta de sujetos que ejecutan acciones en el caso de uso. Fuente: [Autor]

Para el caso de estudio especifico no hay actores que no se mencionen en la sección de Actores de la especificación del caso de uso y se encuentren como sujetos que ejecutan acciónes.

9.6.9. Inconsistencias en numeración de los pasos

Dentro de la ontología cada paso debe tener un número unico sea un paso del escenario básico o de los escenarios alternativos, si existiese un paso con el mismo número la consulta propuesta mostraria los pasos que tiene el mismo número. Para el caso de estudio no hay pasos con el mismo número.

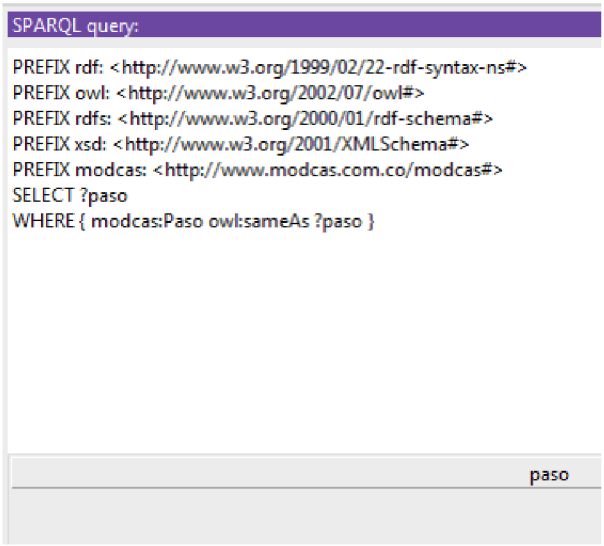


Figura 9.21 Pasos con el mismo número. Fuente: [Autor]

9.6.10. La condición para acceder a un flujo alternativo no es clara

Cada escenario alternativo debe especificar bajo que condición se debe ejecutar, los escenarios alternativos en la ontología MOD-CAS estan relacionados con las condiciones necesarias para que se ejecute cada escenario. La consulta planteada permite conocer si se pueden identificar estas condiciones en el caso de uso modelado.

En la imagen se muestra un escenario alternativo del caso de uso estudiado, que se relaciona con un recurso y el estado en el que ese recurso se debe encontrar para que se active la ejecución del escenario alternativo.

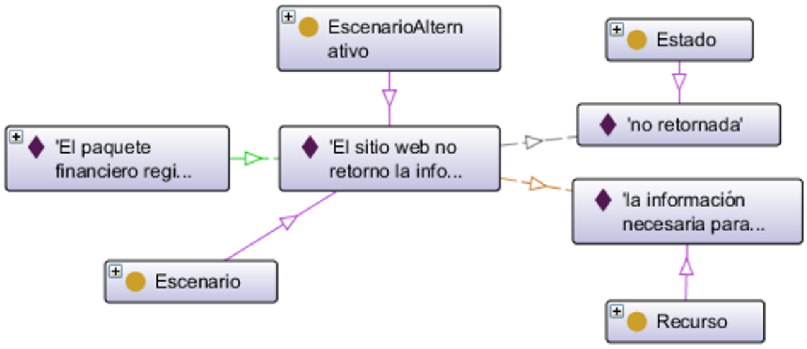


Figura 9.22 Modelado de una condición para un escenario alternativo. Fuente: [Autor]

Al ejecutar la consulta propuesta se pueden obtener para cada escenario alternativo cual es el recurso que relaciona y cual es el estado de dicho recurso que causa la ejecución del escenario alternativo

SPARQL query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX modcas: <http://www.modcas.com.co/modcas#>
SELECT ?escenarioAlternativo ?recurso ?estado
WHERE {?escenarioAlternativo rdf:type modcas:EscenarioAlternativo .
?escenarioAlternativo modcas:recursoCondicion ?recurso .
?escenarioAlternativo modcas:estadoCondicion ?estado .
}
```

recurso	estado	escenarioAlternativo
'la información necesaria para registrar la compra'	'no retornada'	'El sitio web no retorno la información necesaria para registrar la compra'
'ingresar a un sitio'	'no soporta'	'El usuario desea ingresar a un sitio que el paquete financiero no soporta'
'el sitio web de compra de titulos'	'produce un fallo'	'Se produce un fallo en el sitio web de compra de titulos.'
'la compra'	'no aceptada'	'El sitio web no acepta inmediatamente la compra, pero la pone en espera'

Figura 9.23 Recursos y Estados relacionados con los escenarios alternativos. Fuente: [Autor]

Capítulo 10

Conclusiones

10.1. Conclusiones

1. OWL es un lenguaje que nació para definir relaciones semánticas en la web pero es lo suficientemente flexible y robusto para conceptualizar cualquier objeto de estudio.

2. MOD-CAS Permite aportar información a los casos de uso analizados por medio de las inferencias que el diseño de la ontología logra crear.

3. MOD-CAS Esta en capacidad de identificar los errores propuestos en el desarrollo del trabajo.

4. Archimate es un lenguaje para definir arquitectura empresarial que permite tener una trazabilidad completa de la relación entre el negocio de la organización, sus aplicaciones, su infraestructura y el plan de desarrollo de los proyectos que se deseen llevar a cabo.

10.2. Trabajos futuros

1. Complementar el modelo ontológico MOD-CAS para validar varios casos de uso a la vez.

2. Implementación de la arquitectura propuesta para el desarrollo de la aplicación MOD-CAS.

3. Integración de la aplicación con un sistema que permita especificar casos de uso y que apoyado en la ontología y la herramienta MOD-CAS verifique el caso de uso a medida que se esta escribiendo y no cuando y se encuentre especificado.

Apéndice A

Anexos

A.1. Caso de Uso de Prueba

El caso de uso enunciado a continuación se tomo como ejemplo del libro [Coc01] *Writing Effective Use Cases*

A.1.1. Caso de Uso Número CU001:

Titulo:*Comprar titulos en la Web*
Actor:*Comprador*
Precondiciónes:

- El usuario tiene registrado al menos un portafolio financiero

Escenario Principal

1. El usuario selecciona comprar titulos en la web
2. El paquete financiero obtiene el nombre del sitio web de compra de titulos seleccionado por el usuario
3. El paquete financiero abre la conexión al sitio web seleccionado.

4. El usuario busca y compra el titulo en el sitio web.

5. El paquete financiero identifica el titulo comprado y actualiza el portafolio del usuario.

6. El paquete financiero muestra el portafolio del usuario actualizado.

Escenarios Alternativos

2.a El usuario desea ingresar a un sitio que el paquete financiero no soporta

2.a.1 El sistema obtiene un nuevo nombre de sitio web del usuario ó da la opcion de cancelar la ejecución del caso de uso.

3.a Se produce un fallo en el sitio web de compra de titulos.

3.a.1 El sistema reporta la falla al usuario y retorna al paso anterior.

3.a.2 El usuario elige reintenar o salir del caso de uso.

4.a El sitio web no acepta inmediatamente la compra, pero la pone en espera

4.a.1 El paquete financiero registra la transacción en espera, crea un temporizador para registrar la compra posteriormente.

5.a El sitio web no retorno la información necesaria para registrar la compra

5.a.1 El paquete financiero registra la falta de información y pregunta la información incompleta al usuario.

Poscondiciones

- El sitio web seleccionado retorna completamente la información de la compra, el log de la compra y el portafolio del usuario se actualizan correctamente.

A.2. Ontología MOD-CAS

A continuación se enuncia la ontología en formato XML/OWL, este formato puede ser leido por el software Protégé 5.0

El ejemplo de la ontología se puede encontrar publicado para lectura en la versión Web de Protégé con el nombre de proyecto: *modcas*, en la dirección: http://webprotege.stanford.edu/

```
<?xml version="1.0"?>
<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.modcas.com.co/modcas"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    ontologyIRI="http://www.modcas.com.co/modcas"
    versionIRI="http://www.modcas.com.co/modcas/1.0.8">
   <Prefix name="" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Prefix name="rdf"
      IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="modcas" IRI="http://www.modcas.com.co/modcas#"/>
   <Declaration>
      <Class IRI="#Accion"/>
   </Declaration>
   <Declaration> <Class IRI="#Actor"/> </Declaration>
   <Declaration> <Class IRI="#Escenario"/> </Declaration>
   <Declaration>
      <Class IRI="#EscenarioAlternativo"/>
   </Declaration>
   <Declaration>
      <Class IRI="#EscenarioBasico"/>
```

```
</Declaration>
<Declaration> <Class IRI="#Estado"/> </Declaration>
<Declaration>
   <Class IRI="#NumeroDePaso"/>
</Declaration>
<Declaration> <Class IRI="#Paso"/> </Declaration>
<Declaration>
   <Class IRI="#PasoAtomico"/>
</Declaration>
<Declaration>
   <Class IRI="#PasoComplejo"/>
</Declaration>
<Declaration>
   <Class IRI="#PosCondicion"/>
</Declaration>
<Declaration>
   <Class IRI="#Precondicion"/>
</Declaration>
<Declaration> <Class IRI="#Recurso"/> </Declaration>
<Declaration> <Class IRI="#Sujeto"/> </Declaration>
<Declaration>
   <Class IRI="#Titulo"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#afectaA"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#dependeDe"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#ejecuta"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#esPrerequisitoPara"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#estadoAlcanzado"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#estadoCondicion"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#estadoRequerido"/>
</Declaration>
```

```
<Declaration>
   <ObjectProperty IRI="#numeradoCon"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#pasoRelacionado"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#permite"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#pertenece"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#recursoCondicion"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#seRealizaAccion"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#tienePrerequisito"/>
</Declaration>
<Declaration>
   <ObjectProperty IRI="#tienenTitulo"/>
</Declaration>
<Declaration>
   <AnnotationProperty
      IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
</Declaration>
<Declaration>
   <AnnotationProperty
      IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
</Declaration>
<Declaration>
   <AnnotationProperty
      IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
</Declaration>
<Declaration>
   <AnnotationProperty
      IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
</Declaration>
<Declaration>
   <AnnotationProperty
      IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
</Declaration>
```

```
<EquivalentClasses>
   <Class IRI="#PosCondicion"/>
   <ObjectMinCardinality cardinality="1">
      <ObjectProperty IRI="#estadoAlcanzado"/>
      <Class IRI="#Estado"/>
   </ObjectMinCardinality>
</EquivalentClasses>
<EquivalentClasses>
   <Class IRI="#Precondicion"/>
   <ObjectMinCardinality cardinality="1">
      <ObjectProperty IRI="#estadoRequerido"/>
      <Class IRI="#Estado"/>
   </ObjectMinCardinality>
</EquivalentClasses>
<SubClassOf>
   <Class IRI="#EscenarioAlternativo"/>
   <Class IRI="#Escenario"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#EscenarioBasico"/>
   <Class IRI="#Escenario"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#PasoAtomico"/>
   <Class IRI="#Paso"/>
</SubClassOf>
<SubClassOf>
   <Class IRI="#PasoComplejo"/>
   <Class IRI="#Paso"/>
</SubClassOf>
<SubObjectPropertyOf>
   <ObjectProperty IRI="#dependeDe"/>
   <ObjectProperty IRI="#tienePrerequisito"/>
</SubObjectPropertyOf>
<SubObjectPropertyOf>
   <ObjectProperty IRI="#esPrerequisitoPara"/>
   <ObjectProperty IRI="#pasoRelacionado"/>
</SubObjectPropertyOf>
<SubObjectPropertyOf>
   <ObjectProperty IRI="#permite"/>
   <ObjectProperty IRI="#esPrerequisitoPara"/>
</SubObjectPropertyOf>
<SubObjectPropertyOf>
   <ObjectProperty IRI="#recursoCondicion"/>
```

```
      <ObjectProperty abbreviatedIRI="owl:topObjectProperty"/>
   </SubObjectPropertyOf>
   <SubObjectPropertyOf>
      <ObjectProperty IRI="#tienePrerequisito"/>
      <ObjectProperty IRI="#pasoRelacionado"/>
   </SubObjectPropertyOf>
   <FunctionalObjectProperty>
      <ObjectProperty IRI="#numeradoCon"/>
   </FunctionalObjectProperty>
   <InverseFunctionalObjectProperty>
      <ObjectProperty IRI="#numeradoCon"/>
   </InverseFunctionalObjectProperty>
   <TransitiveObjectProperty>
      <ObjectProperty IRI="#esPrerequisitoPara"/>
   </TransitiveObjectProperty>
   <TransitiveObjectProperty>
      <ObjectProperty IRI="#tienePrerequisito"/>
   </TransitiveObjectProperty>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#afectaA"/>
      <Class IRI="#Accion"/>
   </ObjectPropertyDomain>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#ejecuta"/>
      <Class IRI="#Sujeto"/>
   </ObjectPropertyDomain>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#estadoAlcanzado"/>
      <Class IRI="#Recurso"/>
   </ObjectPropertyDomain>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#estadoCondicion"/>
      <Class IRI="#EscenarioAlternativo"/>
   </ObjectPropertyDomain>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#estadoRequerido"/>
      <Class IRI="#Recurso"/>
   </ObjectPropertyDomain>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#numeradoCon"/>
      <Class IRI="#Paso"/>
   </ObjectPropertyDomain>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#pertenece"/>
```

```
      <Class IRI="#Paso"/>
   </ObjectPropertyDomain>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#recursoCondicion"/>
      <Class IRI="#EscenarioAlternativo"/>
   </ObjectPropertyDomain>
   <ObjectPropertyDomain>
      <ObjectProperty IRI="#seRealizaAccion"/>
      <Class IRI="#Paso"/>
   </ObjectPropertyDomain>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#afectaA"/>
      <Class IRI="#Recurso"/>
   </ObjectPropertyRange>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#ejecuta"/>
      <Class IRI="#Accion"/>
   </ObjectPropertyRange>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#estadoAlcanzado"/>
      <Class IRI="#Estado"/>
   </ObjectPropertyRange>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#estadoCondicion"/>
      <Class IRI="#Estado"/>
   </ObjectPropertyRange>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#estadoRequerido"/>
      <Class IRI="#Estado"/>
   </ObjectPropertyRange>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#numeradoCon"/>
      <Class IRI="#NumeroDePaso"/>
   </ObjectPropertyRange>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#pertenece"/>
      <Class IRI="#Escenario"/>
   </ObjectPropertyRange>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#recursoCondicion"/>
      <Class IRI="#Recurso"/>
   </ObjectPropertyRange>
   <ObjectPropertyRange>
      <ObjectProperty IRI="#seRealizaAccion"/>
```

```
      <Class IRI="#Accion"/>
   </ObjectPropertyRange>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
      <IRI>#Accion</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Accions</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
      <IRI>#Accion</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Accion</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
      <IRI>#Actor</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Actors</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
      <IRI>#Actor</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Actor</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
      <IRI>#Escenario</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Escenarios</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
      <IRI>#Escenario</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Escenario</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
      <IRI>#EscenarioAlternativo</IRI>
      <Literal
         datatypeIRI="&rdf;PlainLiteral">EscenarioAlternativoes</Literal>
```

```
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
        <IRI>#EscenarioAlternativo</IRI>
        <Literal
            datatypeIRI="&rdf;PlainLiteral">EscenarioAlternativo</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
        <IRI>#EscenarioBasico</IRI>
        <Literal
            datatypeIRI="&rdf;PlainLiteral">EscenarioBasicoes</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
        <IRI>#EscenarioBasico</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">EscenarioBasico</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
        <IRI>#NumeroDePaso</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">NumeroDePasoes</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
        <IRI>#NumeroDePaso</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">NumeroDePaso</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
        <IRI>#Paso</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">Pasoes</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
        <IRI>#Paso</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">Paso</Literal>
```

```
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
        <IRI>#PasoAtomico</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">PasoAtomicoes</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
        <IRI>#PasoAtomico</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">PasoAtomico</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
        <IRI>#PasoComplejo</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">PasoComplejoes</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
        <IRI>#PasoComplejo</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">PasoComplejo</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
        <IRI>#PosCondicion</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">PosCondicions</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
        <IRI>#PosCondicion</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">PosCondicion</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
        <IRI>#Precondicion</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">Precondicions</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
```

```
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
      <IRI>#Precondicion</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Precondicion</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
      <IRI>#Recurso</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Recursoes</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
      <IRI>#Recurso</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Recurso</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
      <IRI>#Sujeto</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Sujetoes</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
      <IRI>#Sujeto</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Sujeto</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_pl"/>
      <IRI>#Titulo</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Tituloes</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#CN_sg"/>
      <IRI>#Titulo</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">Titulo</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
```

```
      <IRI>#afectaA</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">afectaA</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
      <IRI>#afectaA</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">afectaAs</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
      <IRI>#afectaA</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">afectaAed</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
      <IRI>#dependeDe</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">dependeDe</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
      <IRI>#dependeDe</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">dependeDes</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
      <IRI>#dependeDe</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">dependeDed</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
      <IRI>#ejecuta</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">ejecuta</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
      <IRI>#ejecuta</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">ejecutas</Literal>
```

```
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
        <IRI>#ejecuta</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">ejecutaed</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
        <IRI>#esPrerequisitoPara</IRI>
        <Literal
            datatypeIRI="&rdf;PlainLiteral">esPrerequisitoPara</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
        <IRI>#esPrerequisitoPara</IRI>
        <Literal
            datatypeIRI="&rdf;PlainLiteral">esPrerequisitoParas</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
        <IRI>#esPrerequisitoPara</IRI>
        <Literal
            datatypeIRI="&rdf;PlainLiteral">esPrerequisitoParaed</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
        <IRI>#estadoRequerido</IRI>
        <Literal datatypeIRI="&rdf;PlainLiteral">estadoRequerido</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
        <IRI>#estadoRequerido</IRI>
        <Literal
            datatypeIRI="&rdf;PlainLiteral">estadoRequeridoes</Literal>
    </AnnotationAssertion>
    <AnnotationAssertion>
        <AnnotationProperty
            IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
```

```
      <IRI>#estadoRequerido</IRI>
      <Literal
         datatypeIRI="&rdf;PlainLiteral">estadoRequeridoed</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
      <IRI>#numeradoCon</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">numeradoCon</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
      <IRI>#numeradoCon</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">numeradoCons</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
      <IRI>#numeradoCon</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">numeradoConed</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
      <IRI>#pasoRelacionado</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">pasoRelacionado</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
      <IRI>#pasoRelacionado</IRI>
      <Literal
         datatypeIRI="&rdf;PlainLiteral">pasoRelacionadoes</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
      <IRI>#pasoRelacionado</IRI>
      <Literal
         datatypeIRI="&rdf;PlainLiteral">pasoRelacionadoed</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
```

```
    <AnnotationProperty
        IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
    <IRI>#permite</IRI>
    <Literal datatypeIRI="&rdf;PlainLiteral">permite</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
    <AnnotationProperty
        IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
    <IRI>#permite</IRI>
    <Literal datatypeIRI="&rdf;PlainLiteral">permites</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
    <AnnotationProperty
        IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
    <IRI>#permite</IRI>
    <Literal datatypeIRI="&rdf;PlainLiteral">permited</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
    <AnnotationProperty
        IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
    <IRI>#seRealizaAccion</IRI>
    <Literal datatypeIRI="&rdf;PlainLiteral">seRealizaAccion</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
    <AnnotationProperty
        IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
    <IRI>#seRealizaAccion</IRI>
    <Literal
        datatypeIRI="&rdf;PlainLiteral">seRealizaAccions</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
    <AnnotationProperty
        IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
    <IRI>#seRealizaAccion</IRI>
    <Literal
        datatypeIRI="&rdf;PlainLiteral">seRealizaAccioned</Literal>
</AnnotationAssertion>
<AnnotationAssertion>
    <AnnotationProperty
        IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
    <IRI>#tienePrerequisito</IRI>
    <Literal
        datatypeIRI="&rdf;PlainLiteral">tienePrerequisito</Literal>
</AnnotationAssertion>
```

```
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
      <IRI>#tienePrerequisito</IRI>
      <Literal
         datatypeIRI="&rdf;PlainLiteral">tienePrerequisitoes</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
      <IRI>#tienePrerequisito</IRI>
      <Literal
         datatypeIRI="&rdf;PlainLiteral">tienePrerequisitoed</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_pl"/>
      <IRI>#tienenTitulo</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">tienenTitulo</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_sg"/>
      <IRI>#tienenTitulo</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">tienenTituloes</Literal>
   </AnnotationAssertion>
   <AnnotationAssertion>
      <AnnotationProperty
         IRI="http://attempto.ifi.uzh.ch/ace_lexicon#TV_vbg"/>
      <IRI>#tienenTitulo</IRI>
      <Literal datatypeIRI="&rdf;PlainLiteral">tienenTituloed</Literal>
   </AnnotationAssertion>
</Ontology>
<!-- Generated by the OWL API (version 3.5.1)
   http://owlapi.sourceforge.net -->
```

Bibliografía

[AU15] Agnes Koschmider Tomasz Janasz Axel Uhl, Matthias Born. *Digital Enterprise Transformation A Business-Driven Approach to Leveraging Innovative IT*. Gower, 2015.

[Bec15] Sean Bechhofer. An introduction to owl. citado en Diciembre 2015, online: http://kmi.open.ac.uk/events/iswc08-semantic-web-intro/slides/02%20-%20Sean.pdf, 2015.

[CG11] Jesús R. Campaña Gómez. Representación y tratamiento semántico de información imprecisa en bases de datos. Master's thesis, Universidad de Granada, Granada España, 2011.

[Coc01] Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2001.

[CR98] Camille Ben Achour Colette Rolland. Guiding the construction of textual use case specifications. *Data Knowledge Engineering Journal*, pages 125–160, 1998.

[fBIR15] Stanford Center for Biomedical Informatics Research. Protégé. citado en Diciembre 2015, online: http://protege.stanford.edu, 2015.

[Gro13] The Open Group. Welcome to archimate® 2.1, an open group standard. citado en Enero 2016, online: http://pubs.opengroup.org/architecture/archimate2-doc/, 2013.

[HPH11] Robert Hirschfeld, Michael Perscheid, and Michael Haupt. Explicit use-case representation in object-oriented programming languages. In *Proceedings of the 7th Symposium on Dynamic Languages*, DLS '11, pages 51–60, New York, NY, USA, 2011. ACM.

[JR98] Grady Booch James Rumbaugh, Ivar Jacobson. *The Unified Modeling Language User Guide.* Addison-Wesley, 1998.

[KS05] H. Kaiya and M. Saeki. Ontology based requirements analysis: lightweight semantic processing approach. In *Quality Software, 2005. (QSIC 2005). Fifth International Conference on*, pages 223–230, Sept 2005.

[Llo07] Roberto Romero Llop. Especificación owl de una ontología para teleeducación en la web semántica. Master's thesis, Universidad Politecnica de Valencia, Valencia, España, 2007.

[Mil15] Michael G. Miller. Business architecture: Information necessity. *Business and Dynamic Change*, pages 11–20, 2015.

[MT14] Azucena Montes Mireya Tovar, David Pinto. Validación de conceptos ontológicos usando métodos de agrupamiento. *Research in Computing Science*, pages 9–16, 2014.

[NFN05] Erick Antezana Natalya F. Noy, Deborah L. McGuinness. *Desarrollo de Ontologías-101: Guía Para Crear Tu Primera*. Stanford University, Stanford, CA,, The address of the publisher, 2005.

[PD14] Gilbert Raymond Philippe Desfray. *Modeling Enterprise Architecture with TOGAF.* Morgan Kaufmann, 2014.

[PSS14] Deepti Parachuri, A. S. M. Sajeev, and Rakesh Shukla. An empirical study of structural defects in industrial use-cases. In *Companion Proceedings of the 36th International Conference on Software Engineering*, ICSE Companion 2014, pages 14–23, New York, NY, USA, 2014. ACM.

[SRP+14] Kiran Prakash Sawant, Suman Roy, Deepti Parachuri, François Plesse, and Pushpak Bhattacharya. Enforcing structure on textual use cases via annotation models. In *Proceedings of the 7th India Software Engineering Conference*, ISEC '14, pages 18:1–18:6, New York, NY, USA, 2014. ACM.

[Su5] Pablo Díaz Suárez. Lenguaje de ontologías web (owl) vista general. citado en Diciembre 2015, online: http://www.w3.org/2007/09/OWL-Overview-es.html, 2015.

[TIPO06] F. Tørner, M. Ivarsson, F. Pettersson, and P. Öhman. Defects in automotive use cases. In *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ISESE '06, pages 115–123, New York, NY, USA, 2006. ACM.

[Woo16] Viveca Woods. Gartner identifies the top 10 strategic technology trends for 2016. citado en Enero 2016, online: http://www.gartner.com/newsroom/id/3143521, 2016.