PROPUESTA DE ARQUITECTURA DIRIGADA POR MODELOS (MDA) CON BASE EN UNA ONTOLOGIA DE DOMINIO PARA EL PROCESO DE PETICIONES, QUEJAS Y RECLAMOS (PQR) EN LAS ENTIDADES DEL ESTADO COLOMBIANO

HERNÁN ORLANDO MORENO HUÉRFANO

CARLOS ENRIQUE PEDROZA NOGUERA

ANIBAL PARRA BUITRAGO

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

ESPECIALIZACIÓN EN INGENIERIA DE SOFTWARE

BOGOTA

2016

PROPUESTA DE ARQUITECTURA DIRIGADA POR MODELOS (MDA) CON BASE EN UNA ONTOLOGIA DE DOMINIO PARA EL PROCESO DE PETICIONES, QUEJAS Y RECLAMOS (PQR) EN LAS ENTIDADES DEL ESTADO COLOMBIANO

HERNÁN ORLANDO MORENO HUÉRFANO

CARLOS ENRIQUE PEDROZA NOGUERA

ANIBAL PARRA BUITRAGO

Proyecto para optar por el título de Especialista en Ingeniería de Software

Director:

Ph.D. SANDRO JAVIER BOLAÑOS CASTRO

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

ESPECIALIZACIÓN EN INGENIERIA DE SOFTWARE

BOGOTA

2016

Nota de Aceptación

Firma del Director

Firma del Jurado

Bogotá, 18 de Mayo de 2016

CONTENIDO

INTRODUCCIÓN 9
PARTE 1. CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN 11
CAPITULO 1. DESCRIPCIÓN DE LA INVESTIGACIÓN 11
1.1. PLANTEAMIENTO/IDENTIFICACIÓN DEL PROBLEMA 11
1.2. OBJETIVOS 12
1.2.1. OBJETIVO GENERAL 12
1.2.2. OBJETIVOS ESPECÍFICOS 12
1.3. JUSTIFICACIÓN DEL TRABAJO/INVESTIGACIÓN 13
1.4. HIPÓTESIS 13
1.5. MARCO REFERENCIAL 14
1.5.1. MARCO TEÓRICO 14
1.5.2. MARCO CONCEPTUAL 17
1.5.3. MARCO NORMATIVO 27
1.6. METODOLOGÍA DE LA INVESTIGACIÓN 28
1.7. ORGANIZACIÓN DEL TRABAJO DE GRADO 29
1.8. ESTUDIO DE SISTEMAS PREVIOS 31
PARTE 2. DESARROLLO DE LA INVESTIGACIÓN 34
CAPÍTULO 2. INVESTIGACIÓN PRELIMINAR 34
2.1. RECOLECCIÓN DE LA INFORMACIÓN 34
2.2. REQUERIMIENTOS FUNCIONALES 43
2.2.1. LISTA DE REQUERIMIENTOS DE ALTO NIVEL 43
2.2.2. MODELO BPM DE ALTO NIVEL 44

2.2.3. MODELO DE DOMINIO 45
2.3. REQUERIMIENTOS NO FUNCIONALES 46
CAPÍTULO 3. CONSTRUCCIÓN DEL MODELO ONTOLÓGICO 49
3.1. FASE INICIAL 49
3.1.1. ESTUDIO DEL DOMINIO 50
3.1.2. PRINCIPIOS DE DISEÑO DEL MODELO ONTOLÓGICO 50
3.1.3. HERRAMIENTA SELECCIONADA: PROTÉGÉ 51
3.1.4. LENGUAJE SELECCIONADO: OWL DL 51
3.2. DESARROLLO METODOLÓGICO 52
3.2.1. DOMINIO Y ALCANCE DE LA ONTOLOGÍA 52
3.2.2. REUTILIZACIÓN DE ONTOLOGÍAS EXISTENTES 53
3.2.3. TÉRMINOS IMPORTANTES EN LA ONTOLOGÍA 53
3.2.4. CLASES Y JERARQUÍA 54
3.2.5. PROPIEDADES 63
3.2.6. RESTRICCIONES 69
3.2.7. CREACIÓN DE INSTANCIAS 70
CAPÍTULO 4. PROPUESTA DE ARQUITECTURA DIRIGIDA POR MODELOS 71
4.1. INTRODUCCION 71
4.2. PROPUESTA MDA 74
4.2.1. PRESENTACIÓN DE CONCEPTOS 74
4.2.2. PRESENTACION DE MODELOS 75
CAPÍTULO 5. DESARROLLO DEL PROTOTIPO DE ALTO NIVEL 86
PARTE 3. CIERRE DE LA INVESTIGACIÓN 95
CAPÍTULO 7. CONCLUSIONES 95
7.1. VERIFICACIÓN, CONTRASTE Y EVALUACIÓN DE LOS OBJETIVOS 95
7.2. SÍNTESIS DEL MODELO PROPUESTO 96

7.3. APORTES ORIGINALES.....97
CAPÍTULO 8. PROSPECTIVA DEL TRABAJO DE GRADO.....97
8.1. LÍNEAS DE INVESTIGACIÓN FUTURA.....97
8.2. TRABAJOS FUTUROS.....98
BIBLIOGRAFIA.....99

FIGURAS

Ilustración 1- Modelo Estándar de Control Interno 14
Ilustración 2 - Proceso MDA 24
Ilustración 3 – Metamodelo proceso eCommerce 33
Ilustración 4 - MECI y el proceso PQDR 35
Ilustración 5 - Mapa conceptual del dominio 36
Ilustración 6 - Resultado de la revisión 42
Ilustración 7 - Lista de requerimientos de alto nivel 43
Ilustración 8 - Lista de actores 44
Ilustración 9 - Proceso PQRD cuando se conoce la respuesta 44
Ilustración 10 - Proceso PQRD cuando NO se conoce la respuesta 45
Ilustración 11 - Modelo Dominio PQRD 46
Ilustración 12 - Términos importantes 54
Ilustración 13 - Taxonomía de primer nivel de la ontología 55
Ilustración 14 - Taxonomía para Actor 55
Ilustración 15 - Taxonomía para Forma 56
Ilustración 16 - Taxonomía para Interés 56
Ilustración 17 - Taxonomía para Notificación 57
Ilustración 18 - Taxonomía para Presentación 57
Ilustración 19 - Taxonomía para Recurso 58
Ilustración 20 - Taxonomía para Requisitos 58
Ilustración 21 - Taxonomía para Respuesta 59
Ilustración 22 - Taxonomía para Situaciones 59
Ilustración 23 - Taxonomía para Solicitud 60
Ilustración 24 - Taxonomía de la Ontología 61
Ilustración 25 - Taxonomía grafica de la ontología 62
Ilustración 26 - Propiedades de datos 63
Ilustración 27 - Propiedades de objetos 64
Ilustración 28 - Relación utiliza 65
Ilustración 29 - Relación DisponeDe 65
Ilustración 30 - Relación esModificadaPor 66
Ilustración 31 - Relación eRequisitoDe 66
Ilustración 32 - Relación informaA 67
Ilustración 33 - Relación esGenerada 67

Ilustración 34 - Relación produce 68
Ilustración 35 - Relación hasSubclass 69
Ilustración 36 Las relaciones entre los modelos, metamodelos y sistemas 72
Ilustración 37 - BMM ejemplo 76
Ilustración 38 - Organization Model 76
Ilustración 39 - Actor Cooperation Model 77
Ilustración 40 - Business Function Model 77
Ilustración 41 - Business Process Model 78
Ilustración 42 - RM ODP Enterprise Viewpoint de un sistema para una biblioteca 79
Ilustración 43 - RM ODP Viewpoints para un Hospital 79
Ilustración 44 - Diagrama de Clases 81
Ilustración 45 - Diagrama de componentes 82
Ilustración 46 - Diagrama de paquetes 82
Ilustración 47 Diagrama de actividades del proceso PQRD cuando el grupo de atención conoce la respuesta 83
Ilustración 48 Diagrama de actividades del proceso PQRD cuando el grupo de atención no conoce la respuesta 83
Ilustración 49 - Diagrama de estado del proceso PQRD cuando el grupo de atención conoce la respuesta 84
Ilustración 50 - Diagrama de estado del proceso PQRD cuando el grupo de atención no conoce la respuesta 84
Ilustración 51 - Diagrama de despliegue 85
Ilustración 52 - Diagrama de casos de uso 85
Ilustración 53 - Comparación herramientas presentadas por la OMG 87
Ilustración 54 - Presentación de las opciones del plugin 88
Ilustración 55 - Vista ECore con detalles de Spring Roo 89
Ilustración 56 - Generación Script Spring Roo 90
Ilustración 57 - Script Spring Roo 90
Ilustración 58 - Aplicación generada 91
Ilustración 59 - Comando ejecución aplicación 92
Ilustración 60 - Servidor en ejecución con la aplicación 92
Ilustración 61 - Aplicación desde el navegador 93
Ilustración 62 - Consulta de Personas dentro de la aplicación 93
Ilustración 63 - Consulta de Solicitudes dentro de la aplicación 94
Ilustración 64 - Consulta de Respuesta dentro de la aplicación 94

INTRODUCCIÓN

Gobierno en línea es el nombre que recibe la estrategia de gobierno electrónico (e-government) en Colombia, que busca construir un estado más eficiente, más transparente y más participativo gracias a las TIC. Esta estrategia hace parte del MinTIC pero gestionada por la dirección de Gobierno en línea donde manifiesta que el gobierno electrónico no es solo tecnología, debe ayudar a satisfacer las necesidades de los ciudadanos y contribuir al desarrollo de la sociedad.

La Industria de TI, la academia y las organizaciones son actores fundamentales para la estrategia, ya que con su implementación podrán desarrollar alianzas y oportunidades de negocio con las entidades públicas del orden nacional y territorial. Son beneficiados por los acuerdos marco de precios para TI, participan en los programas de fortalecimiento a la industria de TI y en los programas de financiación a proyectos de innovación.

Luego de varios años de implementación de la Estrategia de Gobierno en línea en Colombia y reconociendo los significativos avances que en esa materia se han tenido, así como también los resultados y las tendencias mundiales en gobierno electrónico, se hace necesario dar el paso a una evolución que permita a las entidades públicas adaptarse más fácilmente a las necesidades de la ciudadanía.

El estado Colombiano hoy en día, cuenta con un grado de madurez en temas de gobierno electrónico, es así que cuenta con una serie de instrumentos como son el Marco de referencia de interoperabilidad de regulación, el Manual de Gobierno en línea y el Índice de Gobierno en línea. El estado Colombiano además de tener los instrumentos, cuenta con el MECI y con el Código Contencioso Administrativo.

Teniendo en cuenta lo anterior, este proyecto brinda una herramienta a las nuevas entidades del estado Colombiano que se encuentran en las fases iniciales o busquen cumplir con los

lineamientos de la estrategia de gobierno electrónico, específicamente lo mencionado dentro del Manual de Gobierno en línea en el apartado de TIC para Servicios en el punto de Sistema integrado de PQRD, por medio de la propuesta de una arquitectura dirigida por modelos (MDA) basándose en una ontología de dominio.

PARTE 1. CONTEXTUALIZACIÓN DE LA INVESTIGACIÓN

CAPITULO 1. DESCRIPCIÓN DE LA INVESTIGACIÓN

1.1. PLANTEAMIENTO/IDENTIFICACIÓN DEL PROBLEMA

Siendo el proceso de Peticiones, Quejas, Reclamos y Denuncias (PQRDs) un mecanismo para identificar oportunidades de mejora en los procesos y servicios que brinda cada entidad y con el claro objetivo de garantizar un mejor servicio, se hace evidente en las entidades del Estado Colombiano, la necesidad de implementar un software que apoye el proceso teniendo en cuenta el derecho al que todas las personas tienen de presentar peticiones, quejas, reclamos y denuncias de manera respetuosa a las autoridades por motivos de interés general o particular y a obtener pronta respuesta.

Así mismo, los cambios en los artículos y/o normas que rigen el proceso de las PQRDs requieren de mecanismos que permitan una atención ágil y oportuna de los mismos sin que esto implique interrupciones en el servicio.

De acuerdo con las consultas realizadas, no existen antecedentes de este tipo de proyectos dentro de las entidades del Estado Colombiano.

Habitualmente, las entidades del estado Colombiano se basan en las experiencias de otras entidades con software (comercial o de código abierto) que hayan resuelto sus necesidades, es por esto que se encuentran casos exitosos en la utilización del sistema de gestión documental ORFEO como herramienta para el apoyo de los procesos de PQRDs en las entidades del Estado Colombiano.

Por su parte el Ministerio de Tecnologías de la Información y las Comunicaciones (MinTIC), a través de su fábrica de software, ha venido implementando este tipo de

soluciones puntualmente a la entidad que lo requiera, sin tener en cuenta que todas las entidades del Estado Colombiano deberían tenerlas.

Como alternativas para superar la situación actual se propone realizar un levantamiento de información con el fin de poder determinar las funcionalidades que deberían tenerse en cuenta dentro de un software que apoye el proceso de PQRDs.

Adicionalmente, se propone la construcción de una ontología que se convierta en un apoyo para la definición de una arquitectura dirigida por modelos (MDA) con el objetivo de realizar el diseño global del proceso de peticiones, quejas y reclamos en las entidades del Estado Colombiano.

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Proponer una arquitectura dirigida por modelos (MDA) haciendo uso de una ontología de dominio del proceso de peticiones, quejas y reclamos (PQR) que deben seguir las entidades del Estado Colombiano.

1.2.2. OBJETIVOS ESPECÍFICOS

- Describir el dominio de las PQRs utilizando una ontología de aplicación que sirva como base para la construcción de software relacionado con el proceso de las PQRs.

- Proporcionar a través de modelos las especificaciones resultantes de la descripción del dominio de las PQRs para los futuros proyectos de construcción de software realizado en las entidades del estado Colombiano.

- Complementar el dominio de las PQRs con la estrategia de Gobierno En Línea con el fin de alinearse a los objetivos del MinTIC en relación a la construcción de software dentro de las entidades del estado Colombiano.

1.3. JUSTIFICACIÓN DEL TRABAJO/INVESTIGACIÓN

El proyecto surge como apoyo a las entidades del estado colombiano que requieran implementar un mecanismo de software para el proceso de Atención de Petición, Quejas y Reclamos (PQR) atendiendo a lo establecido en el Artículo 76 de la Ley 1474 de 2011 (Ley 1474, 2011) que establece "…En toda entidad pública, deberá existir por lo menos una dependencia encargada de recibir, tramitar y resolver las quejas, sugerencias y reclamos que los ciudadanos formulen, y que se relacionen con el cumplimiento de la misión de la entidad…", además de lo consignado en la Ley 1437 de 2011 por la cual se expide el Código De Procedimiento Administrativo y de lo Contencioso Administrativo donde se definen los principios, actores, términos de respuesta, entre otros para poder tratar un proceso PQR (Ley 1437, 2011).

Por lo anterior, es necesario representar el proceso de Atención de Petición, Quejas y Reclamos (PQR) de forma estandarizada, flexible y consistente con la normatividad vigente, haciendo uso de elementos que permitan tanto representar el conocimiento concertado sobre este dominio, con el fin de lograr una visión más amplia para la especificación del sistema, como obtener una definición de la arquitectura de aplicaciones de mejor calidad para que sea el soporte de desarrollo de cualquier aplicación para el proceso PQR en cualquier entidad del estado colombiano brindando a estas y a los demás interesados, las bases para la implementación del sistema de información para el proceso.

1.4. HIPÓTESIS

La definición de una Arquitectura Dirigida por Modelos (MDA) basada en el uso de una Ontología de dominio del proceso de Peticiones, Quejas y Reclamos (PQR), permite

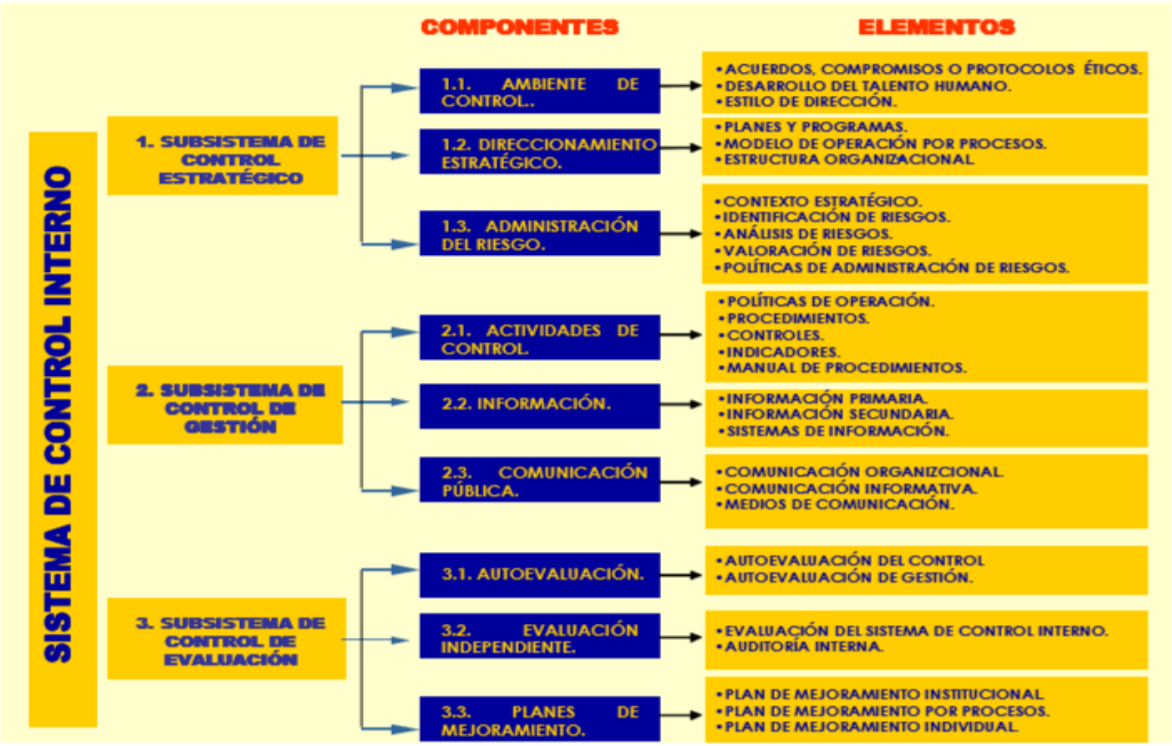
establecer una definición global, estandarizada y consistente con la normatividad vigente para la implementación del sistema de información que respalde este proceso en las entidades del Estado Colombiano.

1.5. MARCO REFERENCIAL

1.5.1. MARCO TEÓRICO

El Modelo Estándar de Control Interno para entidades del Estado (MECI, 2014), se genera tomando como base el artículo 1° de la Ley 87 de 1993, el cual se encuentra compuesto por una serie de Subsistemas, Componentes y Elementos de Control, establecidos en la siguiente ilustración, identificando de esta manera los vínculos existentes entre cada uno de ellos, bajo el enfoque sistémico que establece la citada Ley.

Ilustración 1- Modelo Estándar de Control Interno



Fuente: (MECI, 2014)

Teniendo en cuenta el Modelo Estándar de Control Interno nuestro proyecto se encuentra dentro del Subsistema de Control de Gestión, dentro del componente de Información, y el elemento Información Primaria (Externa); ya que a este grupo pertenece la información que es recopilada primordialmente de la comunidad, demás grupos de interés y su entorno, útil para una adecuada toma de decisiones. Esta información se obtiene a partir de los Sistemas de Quejas y Reclamos y de la Normativa Externa.

También se encuentra dentro del Subsistema de Control de Gestión en el componente de Información y el elemento Sistemas de Información; ya que a este grupo pertenecen el conjunto de recursos humanos y tecnológicos utilizados para la producción de información. Los Sistemas de Información contienen hardware, software, recurso humano, datos e información.

Según la estrategia del Ministerio de Tecnologías de la Información y las Comunicaciones dentro de sus instrumentos orientados al Proyecto de Arquitectura TI Colombia, cuenta con un Marco de Referencia, con la que busca habilitar las estrategias de TIC para Servicios, TIC para la gestión, TIC para el gobierno abierto y para la Seguridad y la privacidad. Este instrumento establece la estructura conceptual, define lineamientos, incorpora mejores prácticas y traza la ruta de implementación de la Arquitectura TI (Arquitectura TI, 2015).

Según el Código De Procedimiento Administrativo y de lo Contencioso Administrativo (Ley 1437, 2011) se definen los principios, actores, términos de respuesta, entre otros, para poder tratar un proceso PQR, los cuales harán parte de los requerimientos que se usarán para la construcción de la Ontología que sirve de apoyo para la propuesta de Arquitectura Dirigida por Modelos de nuestro proyecto.

La Arquitectura Dirigida por Modelos (MDA) como las Ontologías son dos de los recursos cada vez más utilizados en el desarrollo de sistemas de software (Lopez, Servetto, Echeverría, Jeder, Grossi, & Rey, 2011). Una de las tendencias actuales en la ingeniería de software es el uso de ontologías como parte del proceso del Desarrollo Dirigido por Modelos (MDD), es decir, como medio para representar los modelos conceptuales que

definen al sistema a desarrollar en el paradigma MDD (Pascuas, Mendoza, & Córdoba, 2014).

Una ontología representa un modelo abstracto que identifica los componentes más relevantes de cierto fenómeno del mundo real, y los describe en un lenguaje formal, es decir, que puede ser entendido por una computadora, a través de conceptos, propiedades, relaciones, funciones, restricciones y axiomas (Nájera, 2013), además, las ontologías facilitan el desarrollo de aplicaciones mediante la separación de conocimiento acerca del dominio del problema, respecto del resto del código de la aplicación (Zapata, Giraldo, & Urrego, 2010).

La ventaja principal del uso de ontologías como modelo en MDA es que en una sola ontología es posible representar la estructura del sistema, su comportamiento y el modelo de datos, además de describir la lógica de negocio y restricciones de software en el mismo modelo. Entre otras ventajas del uso de las ontologías, además de las que por defecto se heredan de MDD, se encuentra que proporcionan un entendimiento compartido del dominio del problema y una especificación formal de los requisitos de un sistema, lo que permite el reúso de definiciones existentes de conocimiento del dominio; además proporcionan mayor soporte para inferencias lógicas, y la integración e interoperabilidad con otros componentes o aplicaciones (Lopez, Servetto, Echeverría, Jeder, Grossi, & Rey, 2011).

Siendo la Arquitectura Dirigida por Modelos (MDA) y las Ontologías son dos de los recursos actuales frecuentemente utilizados tanto por los desarrolladores cuanto por los teóricos del software, en esta línea podemos ubicar los desarrollos cuyo punto de partida es una ontología, la cual a través de transformaciones es convertida en un modelo. La característica principal de este grupo es la tendencia a la reutilización de ontologías y uno de los principales motivos para el uso de una ontología como base de modelado es que estas representan el conocimiento concertado sobre un dominio determinado, generalmente dado por una comunidad especializada en dicho dominio (Lopez, Servetto, Echeverría, Jeder, Grossi, & Rey, 2011).También la ontología constituye el punto de referencia sobre el cual

es comparado el modelo final para ver si cumple o no con los requerimientos y la conceptualización inicial (Fensel, 2001).

Existen enfoques diferentes para trabajar MDA con las Ontologías, uno de ellos y el más importante y relevante para el proyecto, propone la creación de Modelos Conceptuales partiendo de una Ontología (Wongthongtham, E, Dillon, & Sommerville, 2006). Este método propone obtener una ontología del dominio sobre el cual se desea hacer un desarrollo. En este paso se puede crear la ontología o usar una ya existente que se encuentre disponible. La ontología permitirá dar una visión más amplia para la especificación del sistema y reducirá el número de aspectos no contemplados en el desarrollo (Siddiqui & Alam, 2010). A través de procesos de refinamiento de la ontología y de validación del modelo contra la ontología, se obtendrá un modelo conceptual de mejor calidad para que sea el soporte de desarrollo de cualquier aplicación. Los procesos de refinamiento y validación, realizan el papel de modelos de transformación que usa MDA (Lopez, Servetto, Echeverría, Jeder, Grossi, & Rey, 2011).

1.5.2. MARCO CONCEPTUAL

1.5.2.1. ONTOLOGÍA

El término "Ontología" tiene su origen en la filosofía, en donde es llamada la teoría del ser, es decir, se encarga del estudio de todo lo que es: qué es, cómo es y cómo es posible. La Ontología se ocupa de establecer las categorías fundamentales o modos generales de ser de las cosas.

En las últimas dos décadas, el término Ontología se convirtió en una palabra relevante para la Ingeniera del conocimiento, donde se han tenido muchas definiciones acerca de lo que es una ontología y además se ha visto como estas definiciones han cambiado y han evolucionado a lo largo del tiempo.

Una de las primeras definiciones que se tienen fue propuesta por Neches (R. Neches, 1991): "Una ontología define los términos y relaciones básicas para la comprensión de un área determinada, así como las reglas para combinar los términos para definir las extensiones del vocabulario", según esta definición, puede verse como una ontología define los conceptos de un área de conocimiento determinada aportando las reglas mediante las que puede ampliarse esa área específica de conocimiento con más términos.

Continuando en 1993, Gruber (Gruber, 1993) postula: "Una ontología es una especificación explicita de una conceptualización", esta es la definición más citada en el ámbito ontológico y por ende se convierte en un estándar en el tema, donde:

- Una conceptualización hace referencia a un modelo abstracto y simplificado del mundo real, o mejor, de una parte del mundo real del que se identifican los conceptos que son relevantes.

- Explícita significa que el tipo de los conceptos utilizados, y las limitaciones sobre su uso son explícitamente definidos.

Para Borst (Borst, 1997), quien modifica ligeramente la definición de Gruber, como muchos otros después de la aparición de esta, dice que: "Las ontologías se definen como una especificación formal de una conceptualización compartida". Lo innovador en esta definición es que la complementa con el término "formal" que se refiere al hecho de que la ontología debería ser computarizable y el término "compartida" que refleja la idea de que una ontología captura conocimiento consensual, es decir, que no es aceptado únicamente por un individuo, sino que debe ser aceptada por un grupo de ellos.

Una ontología describe los conceptos y relaciones que son importantes en un sector determinado al cual se denomina dominio, proporcionando un vocabulario para este, así como una especificación informatizada del significado de los términos utilizados en el vocabulario. Las ontologías van desde las taxonomías y clasificaciones, esquemas de bases de datos, a las teorías plenamente axiomatizadas. En los últimos años, las ontologías se han

adoptado en muchos negocios y la comunidad científica como una forma de compartir y reutilizar el conocimiento de un dominio. Las ontologías son fundamentales para muchas aplicaciones como los portales de conocimiento científico, la gestión de la información y la integración de sistemas, comercio electrónico y servicios Web semánticos (Standford).

Es así que, mediante una Ontología se logra obtener la formulación de un esquema conceptual dentro de uno o varios dominios dados, para facilitar la comunicación y el intercambio de información entre diferentes sistemas y entidades.

La idea es modelar en términos más sencillos los requerimientos del dominio o dominios, de tal forma que se consiga una comunicación más fácil con las partes involucradas en el diseño y construcción de la solución informática.

1.5.2.2. METODOLOGÍAS PARA LA CONSTRUCCIÓN DE ONTOLOGÍAS

Para diseñar una ontología es necesario seguir una metodología que defina los pasos precisos a seguir en la representación de la información. Cada ontología tiene sus propias características, sin embargo, la gran mayoría comparten una serie de propiedades (Fernandez lopez & Gómez Pérez, 1997):

- *Ciclo de vida*: Tiempo transcurrido durante la concepción, elaboración y mantenimiento de la ontología.
- *Herramienta de apoyo*: Las metodologías deben ser apoyadas por herramientas ontológicas que faciliten la creación de la ontología.
- *Nivel de modelado:* Da una idea sobre cómo se realiza la extracción de los datos.
- *Nivel de abstracción:* Brinda a una idea el nivel de relación existente entre la información concreta y la forma en que se extrae dicha información. Cuanto mayor sea el nivel de abstracción, más fácil será para la aplicación la extracción de los datos.

- *Uso de ontología base:* Normalmente se reutilizan términos o características de otras ontologías ya creadas.

La siguiente es una clasificación del método de trabajo que siguen las diferentes metodologías según Ding y Foo (Y & S, 2002):

- *Datos-Fuente:* Vocabularios controlados, corpus de sentencias, extracción en texto libre, preguntas de usuarios.
- *Métodos para la extracción de conceptos:* Contempla las diferentes técnicas empleadas en la extracción de información: análisis sintáctico, procesamiento del lenguaje natural.
- *Métodos para la extracción de relaciones:* Se aplica normalmente de forma automática a partir de diversos algoritmos aunque en ocasiones se aplica de forma manual.
- *Reutilización de ontologías:* Suele ser habitual utilizar como base otros instrumentos terminológicos.
- *Representación de la ontología:* Que va desde la estructura jerárquica, pasando por la lógica de la descripción, hasta los grafos conceptuales y el XML.
- *Herramientas o sistemas asociados:* Donde se puede observar que no siempre hay programas informáticos inmiscuidos en los proyectos orientados a ontologías.

En el desarrollo ontológico se pueden mencionar varias metodologías. En 1990, Lenat y Guha (Guha & Lenat, 1990) publicaron el proceso a seguir en el desarrollo de la ontología *Cyc*. Posteriormente, Uschold y King (Uschold, King, Mooralee, & Zorgios, 1996) publicaron el método utilizado para construir la *Enterprise Ontology*, y Gruninger y Fox (Grununger & Fox, 1995) hicieron lo propio para la ontología TOVE. En 1996, Bernaras (Bernaras, Laresgiti, & Corera, 1996) publicó un proceso para construir ontologías a partir de bases de conocimientos.

En el año 1997, se creó la metodología *Methontology* (Fernandez lopez & Gómez Pérez, 1997). En el mismo año surgió un método para la construcción de ontologías basado en la

ontología SENSUS (Swartout, Ramesh, Knight, & Russ, 1997). Unos años más tarde, en 2001, apareció la metodología On-To-Knowledge (Sure & Studer., 2002).

Posteriormente, nuevos métodos ontológicos, fueron desarrollados, UPON (Nicola, Missikoff, & Navigli, 2005) en 2005, HCOME (Kotis & Vouros, 2006) en el 2006.

Destacando de las metodologías anteriores se encuentran Methontology y On-To-Knowledge, ya que incorporan aspectos innovadores. Además, se debe considerar el grado de dependencia entre la ontología desarrollada y su aplicación final. En este sentido, se puede encontrar que (Corcho, Fernandez lopez, & Gomez perez, 2003):

- El método utilizado en el proyecto KACTUS y la metodología On-To-Knowledge son dependientes de la aplicación, la ontología se construye con base a una aplicación determinada.
- La metodología de Gruninger y Fox, y el método utilizado en SENSUS son semidependientes de la aplicación.

El Cyc y la metodología Methontology son independientes de las aplicaciones, ya que el proceso de desarrollo de la ontología es totalmente independiente de los usos de esta.

1.5.2.3. OWL

OWL (W3C), es un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW que tiene como objetivo facilitar un modelo de marcado construido sobre RDF y codificado en XML.

El Lenguaje de Ontologías para la Web, recomendación del W3C (W3C), está diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido únicamente necesita ser procesado por seres humanos. OWL puede usarse para representar explícitamente el significado de términos en vocabularios y las relaciones entre aquellos términos.

Existen tres variantes de OWL:

- **OWL Lite:** La versión más simple para los programadores principiantes, permite la jerarquía de clasificación y restricciones simples.

- **OWL DL:** Esta versión tiene todo el vocabulario OWL completo, es el indicado para los usuarios que requieren el máximo de expresividad, mientras conservan completamente la propiedad de ser computable (se garantiza que todas las conclusiones son computables) y resoluble (todos los cálculos terminaran en tiempo finito).

- **OWL Full:** Esta versión también incluye todo el vocabulario OWL pero en este caso no hay limitaciones para explotar todo su potencial, permite la máxima expresividad y ofrece la libertad sintáctica de RDF pero no asegura la propiedad de ser computable.

1.5.2.4. HERRAMIENTAS PARA LA CONSTRUCCIÓN DE ONTOLOGÍAS

Las herramientas de creación de ontologías o editores de ontologías, son las herramientas que permiten la codificación de una determinada ontología en base a un determinado lenguaje.

Existen herramientas y programas para realizar anotaciones en páginas Web con lenguajes de marcado propios. La mayoría de estos programas permiten describir el contenido de los documentos en forma de metadatos, soportados sobre una ontología representada en RDF Schema (RDFS) o basados en grafos conceptuales. Algunas herramientas de este tipo son:

- **Protégé**: Herramienta open-source que permite la creación y edición de ontologías así como la creación de bases de conocimiento, es uno de los editores más utilizados para construir ontologías sobre RDFS, OWL y XML Schema.

- **OntoEdit:** Es una herramienta de edición de ontologías que apoya el desarrollo y mantenimiento de las mismas utilizando medios gráficos en un entorno web. Esta herramienta permite crear, navegar y modificar ontologías.

- **SWOOP:** Es una herramienta para crear, editar y depurar ontologías OWL. Fue producido por el laboratorio MIND de la Universidad de Maryland, College Park, pero ahora es un proyecto de código abierto con contribuidores de todo el mundo. Se basa en un interfaz web, es decir, SWOOP actúa como un navegador que permite navegar entre las clases (aprovechando las URIs de los recursos para acceder a ellos).

- **WSMO Studio:** Es un editor de ontologías para modelado de servicios de la Web Semántica, desarrollado por el Web Service Modeling Ontology Group, está disponible como un conjunto de plugins para Eclipse.

1.5.2.5. ARQUITECTURA DIRIGIDA POR MODELOS (MDA)

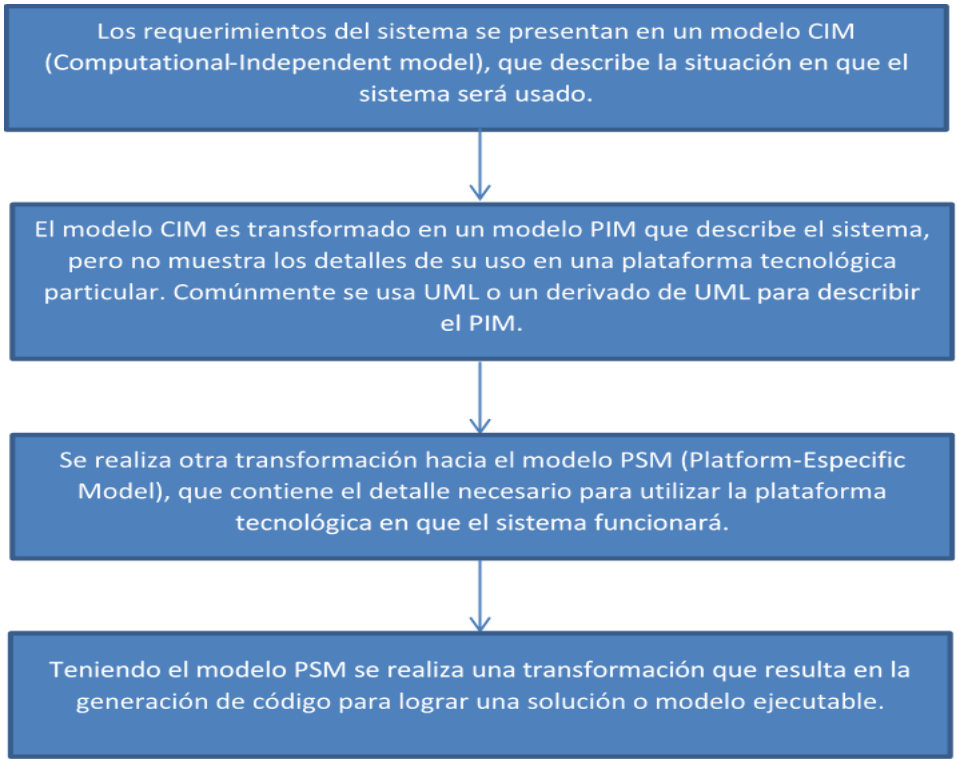
La mayor parte de las metodologías usadas hoy en día en Ingeniería de Software se apoyan en modelos (López & González, 2006), debido a esto hay una tendencia a seguir un enfoque dirigido por modelos que permiten generar sistemas a partir de modelos y transformaciones entre ellos, con los cuales se consiga una menor abstracción.

MDA es un nuevo paradigma de desarrollo de software promovido por Object Management Group (OMG, MDA - The Architecture Of Choice For A Changing World, 2001), donde se usan modelos como herramienta principal para el desarrollo de software, aunque MDA no es un concepto maduro aún.

MDA permite obtener un Modelo Independiente de la Plataforma (PIM), por medio de un lenguaje específico y este modelo obtenido puede convertirse en uno o más Modelos Específicos de la Plataforma (PSM), con el uso de lenguajes concretos del dominio, lenguajes de alto nivel como C#, VB, Java, etc.

MDA tiene como uno de sus principales objetivos separar el diseño de la arquitectura y de las tecnologías de construcción, para que el diseño y la arquitectura puedan ser modificados independientemente. El diseño contiene los requerimientos funcionales y la arquitectura proporciona la infraestructura a través de la cual se concretan requerimientos no funcionales como el rendimiento, la escalabilidad y la fiabilidad. Este tipo de arquitectura proporciona un conjunto de guías que sirven para estructurar especificaciones expresadas como modelos.

Ilustración 2 - Proceso MDA



Fuente: (López & González, 2006)

Las ventajas en el uso de la Arquitectura Dirigida por Modelos (MDA) son:

- MDA permite una estricta separación de responsabilidades. Por un lado se modelan los PIM que representan los modelos de nuestro negocio, y por otro lado, los PSM con las características tecnológicas. Esto permite que ambos modelos puedan

evolucionar por separado. De esta manera si quisiéramos, por ejemplo, modificar un aspecto técnico, bastará con modificar el PSM sin que esto tenga impacto sobre la lógica de negocio. Particularmente, una de esas guías dice que el modelado de la solución debe ser dirigido por el negocio. Esta guía se basa en la afirmación de que un cambio en el negocio seguramente produzca un cambio en el código, pero no lo inverso: los cambios en el código no deberían impactar en el negocio.

- MDA también permite: lidiar con la complejidad del negocio, modelando a éste por separado y permitiendo su análisis y mejora.

- Disminuir costos, si se cuenta con una herramienta MDA adecuada a nuestras necesidades, con la cual se logre la reutilización.

1.5.2.6. TRANSFORMACIONES

Las transformaciones bajo el enfoque (OMG), involucran una especificación del metamodelo llamado Meta Object Facility (MOF), un lenguaje de restricciones llamado Object Constraint Language (OCL) y un lenguaje de transformación llamado Query/View/Transformation (QVT) (OMG, MDA - The Architecture Of Choice For A Changing World, 2001).

La transformación de modelos, por lo general, adopta un enfoque orientado a objetos para la representación y manipulación del tema de sus modelos. La transformación define la generación automática de un modelo a partir de otro modelo. Esta definición es un conjunto de reglas que describen cómo uno o más artefactos en el modelo origen pueden ser transformados en una o más construcciones en el modelo destino (Orozco, Giraldo, & Trefftz, 2013).

Las transformaciones pueden ser:

- **Endógenas**: Si están expresadas en el mismo lenguaje que los modelos.
- **Exógenas**: No están expresadas en el mismo lenguaje que los modelos.
- **Verticales**: Los lenguajes destino son diferentes entre sí.
- **Horizontales**: Los lenguajes destino no son diferentes entre sí.

1.5.2.7. HERRAMIENTAS PARA LA CREACIÓN DE MODELOS

Existen numerosas herramientas para trabajar con MDA y en la mayoría de casos permiten la generación de código para diferentes plataformas, las herramientas más utilizadas son:

- **Enterprise Architect (EA):** Soporta transformaciones de MDA usando plantillas con transformaciones fáciles de editar y desarrollar, es un entorno de modelado basado en la especificación UML 2.0, con soporte completo para el modelo de transformaciones de modelos, así como el modelo impulsado por la generación de común artefactos de desarrollo tales como la documentación, código fuente, scripts de prueba, descriptores de despliegue, esquemas XML, esquemas de bases, etc. (Truyen, 2005).

- **Eclipse:** Plataforma abierta de libre distribución, creada y soportada por una gran comunidad de desarrolladores que han incorporado herramientas para dar apoyo en cada una de las fases del ciclo de vida del enfoque de MDA (The Eclipse Fundation, 2015).

- **AndroMDA:** Es un framework de generación de código fuente abierto que sigue el paradigma de la Arquitectura Dirigida por Modelos (MDA. Toma modelo(s) de herramientas CASE(s) y genera aplicaciones totalmente desplegables y otros componentes (AndroMDA, 2014).

1.5.3. MARCO NORMATIVO

1.5.3.1. MODELO ESTÁNDAR DE CONTROL INTERNO

Las entidades del estado en los últimos años han decidido adoptar herramientas que nacieron en las empresas privadas y que les han servido para mejorar sus procesos y ofrecer un mejor servicio.

Una de esas herramientas es el Sistema de Control Interno, este es un sistema conformado por principios, normas, planes, métodos, procedimientos y mecanismos de verificación y evaluación, cuyo fin es que las actuaciones y actividades de los servidores públicos y el manejo de la información y los recursos, sea eficiente y de calidad.

Por la adopción de este sistema aparece el Modelo Estándar de Control Interno, este modelo se viene utilizando desde la entrada en vigencia del **Decreto 1599 de 2005**, y rige para todas las entidades que hacen parte del ámbito de aplicación de la **ley 87 de 1993**, las organizaciones del estado han involucrado en su cultura organizacional los aspectos básicos para utilizar de manera práctica el Sistema de Control Interno.

En nuestro proyecto debemos tener en cuenta este modelo ya que es la base para la construcción de herramientas desde la perspectiva de la Gestión de Calidad (MECI, 2014).

1.5.3.2. CÓDIGO DE PROCEDIMIENTO ADMINISTRATIVO Y DE LO CONTENCIOSO ADMINISTRATIVO

La publicidad de los actos emanados de la administración es uno de los principios fundamentales de la función pública, motivo por el cual se hace necesario ilustrar al ciudadano con los procedimientos necesarios para la notificación y respuesta frente a las decisiones de las autoridades, las cuales tienen que enmarcarse dentro del “debido proceso”.

Teniendo en cuenta esta necesidad se pretende responder de manera elemental a las inquietudes más frecuentes en esta materia, con el fin que el interesado, pueda hacer valer sus derechos, sin necesidad de tener que acudir a la asesoría especializada.

Se puede decir que el conocimiento de los procedimientos para las notificaciones administrativas, son unas herramientas fundamentales en la defensa de los derechos de los ciudadanos, que es lo que se busca con la promulgación de esta ley.

En nuestro proyecto debemos tener en cuenta esta ley porque ahí es donde se definen los principios, actores, términos de respuesta, entre otros, para poder tratar un proceso PQR en las entidades del estado colombiano (Ley 1437, 2011).

1.5.3.3. ESTRATEGIA DE GOBIERNO EN LÍNEA

La Estrategia de Gobierno en línea, que se plasma en el Decreto Único Reglamentario del Sector de Tecnologías de la Información y las Comunicaciones 1078 de 2015, comprende cuatro (4) grandes propósitos: lograr que los ciudadanos cuenten con servicios en línea de muy alta calidad, impulsar el empoderamiento y la colaboración de los ciudadanos con el Gobierno, encontrar diferentes formas para que la gestión en las entidades públicas sea óptima gracias al uso estratégico de la tecnología y garantizar la seguridad y la privacidad de la información (Marco Jurídico GEL, 2015).

1.6. METODOLOGÍA DE LA INVESTIGACIÓN

El método de investigación que se va a realizar, define la caracterización del problema, las posibles soluciones que se han presentado en el momento o estado del arte de la solución del problema, el conjunto de actividades que se realizarán para construir el prototipo del aplicativo, la aplicación del aplicativo en una población de muestra en el distrito, medición de los resultados obtenidos al hacer uso de la herramienta y finalmente las conclusiones obtenidas al final del estudio.

El método de investigación es análisis, debido a que se inicia por la identificación de cada una de las partes del proceso de la PQRDs. De esta manera se establece la relación causa efecto entre los elementos que componen el objeto de investigación.

Para el desarrollo del proyecto se contemplan varios factores como lo correspondiente al corto tiempo que se tiene para el desarrollo del mismo y lo reducido del equipo de trabajo, por tal motivo se utilizará un modelo basado en el proceso de desarrollo iterativo e incremental lo cual permite ir generando avances.

1.7. ORGANIZACIÓN DEL TRABAJO DE GRADO

CAPITULO 1. DESCRIPCIÓN DE LA INVESTIGACIÓN

En este capítulo lo primero que se hace es identificar el problema, con base en esta información se proponen unos objetivos, la justificación del trabajo y la hipótesis. También en este capítulo se presenta un marco referencial, la metodología que se utilizará y se exponen estudios previos sobre el tema.

CAPÍTULO 2. INVESTIGACIÓN PRELIMINAR

En este capítulo se presenta el estudio profundo del Proceso de Peticiones, Quejas y Reclamos en instituciones públicas del estado colombiano como de la normatividad vigente que lo rige.

CAPÍTULO 3. CONSTRUCCIÓN DEL MODELO ONTOLÓGICO

Este capítulo está dividido en dos parte, la primera parte trata sobre cómo conseguir el Modelo Independiente de la Plataforma (PIM), haciendo uso de una Ontología de Dominio, con lo cual se consigue una solución genérica que minimiza el tiempo de desarrollo y aumenta la reutilización de la interoperabilidad del producto que se desea obtener de este proyecto, también se hace una descripción sobre las herramientas que se utilizarán: PROTÉGÉ y OWL-DL. La segunda parte trata sobre el desarrollo metodológico, ahí se expone el alcance de la Ontología y la reutilización de Ontologías existentes, términos

importantes y otras características que se tendrán en cuenta para el desarrollo de la Ontología.

CAPÍTULO 4. PROPUESTA DE ARQUITECTURA DIRIGIDA POR MODELOS

Este capítulo está dividido en dos partes, la primera parte inicia con una introducción en la que se exponen los conceptos más importantes de la Arquitectura Dirigida por Modelos, posteriormente se explican los temas más relevantes que son usados para la recolección de la información estos temas son: Modelo Estándar de Control Interno (MECI) para las entidades del Estado, Código de Procedimiento Administrativo y de lo Contencioso Administrativo (CPACA) correspondiente a la Ley 1437 del 2.011 y la estrategia de Gobierno en Línea correspondiente en el Decreto Único Reglamentario del Sector Tecnologías de la Información y las Comunicaciones 1078 del 2.015. También dentro de la introducción se exponen los requerimientos funcionales y no funcionales de la arquitectura propuesta.

La segunda parte presenta la propuesta MDA, primero se explica la forma como se aplicará la Arquitectura Dirigida por Modelos (MDA) a nuestro proyecto y después cómo se hace la representación de los modelos CIM Y PIM de la arquitectura.

CAPÍTULO 5. DESARROLLO DEL PROTOTIPO DE ALTO NIVEL

En este capítulo se explica la forma como se realiza nuestro prototipo de alto nivel para la solución.

CAPÍTULO 7. CONCLUSIONES

En este capítulo se presenta en resumen las deducciones a las que llegamos una vez terminado el análisis y desarrollo del proyecto.

CAPÍTULO 8. PROSPECTIVA DEL TRABAJO DE GRADO

Este último capítulo da una mirada desde nuestro punto de vista sobre las causas que llevan a la generación de proyectos como este y la previsión de las situaciones que podrían derivarse de aplicar este tipo de soluciones a futuros proyectos semejantes al nuestro.

1.8. ESTUDIO DE SISTEMAS PREVIOS

TESIS DOCTORAL: "PROPUESTA PARA EL MODELADO DEL CONOCIMIENTO EMPRESARIAL" (Chalmeta Rosaleñ, 2007)

Este proyecto busca llevar a cabo el modelado del conocimiento empresarial para representar el conocimiento que las empresas tienen con el objetivo de procesarlo y utilizarlo cuando sea necesario, este enfoque se ha desarrollado a partir de la metodología **KM-IRIS** para la implantación de un sistema de gestión del conocimiento, a esta iniciativa la llamaron Propuesta **MDK** (Model Driven Knowledge).

Esta propuesta incluye los metamodelos y perfiles de **UML2** implementados para la representación del conocimiento empresarial y una guía de ayuda a las empresas en la elaboración de su mapa de conocimiento empresarial. Como fuente para el desarrollo de los metamodelos se usaron los requisitos y metamodelos definidos en los Proyectos Europeos **INTEROP** y **ATHENA**, y los lenguajes de modelado empresarial **UEML** y **POP*** para así lograr el intercambio de modelos entre empresas que utilizan distintas herramientas de modelado. Todo lo anterior dentro de un enfoque mda, modelando el conocimiento empresarial para obtener el **CIM**, para que los modelos obtenidos puedan ser transformados a **PIM** y **PSM**s.

Una vez finalizado este proyecto se ha logrado definir el conocimiento objetivo de los siguientes bloques conceptuales de conocimiento: ORGANIZACI´ON, PROCESO, PRODUCTO/SERVICIO y RECURSO predefinidos en la Metodología KM-IRIS. Y se han elaborado un conjunto de plantillas en las cuales se recoge para cada bloque el conocimiento objetivo que se desearía gestionar en una empresa tipo clasificado en diversas categorías y subcategorías ontológicas. A pesar de que cada empresa tiene sus propios objetivos y estrategias en estas plantillas se ha identificado el conocimiento general que puede estar presente en cualquier tipo de empresa y luego puede ser particularizado a las características intrínsecas y particulares de cada organización. Por lo tanto, estas plantillas pueden ser utilizadas como modelo de referencia en la fase I de identificación del conocimiento objetivo de la Metodología **KM-IRIS**.

También se han identificado los elementos de la fase II de extracción del conocimiento objetivo de la Metodología **KM-IRIS**. En concreto, se han identificado las variables de entrada necesarias para obtener el conocimiento objetivo identificado en la fase anterior, y las principales fuentes de conocimiento en las cuales se podría localizar este conocimiento en una empresa. El resultado recogido en forma de plantillas también puede ser usado a modo de modelo de referencia en la fase II de la citada metodología. Por último, y relacionado con esta fase, se ha presentado una síntesis de los procedimientos de extracción y cálculo que es necesario aplicar para la extracción del conocimiento, así como de los principales tipos de técnicas que pueden ser utilizados en cada uno de ellos.

Y por último se ha presentado la Propuesta **MDK** para el modelado del mapa de conocimiento de una empresa virtual con la finalidad de dotar a la fase III de representación del conocimiento objetivo de un lenguaje que permita tal fin.

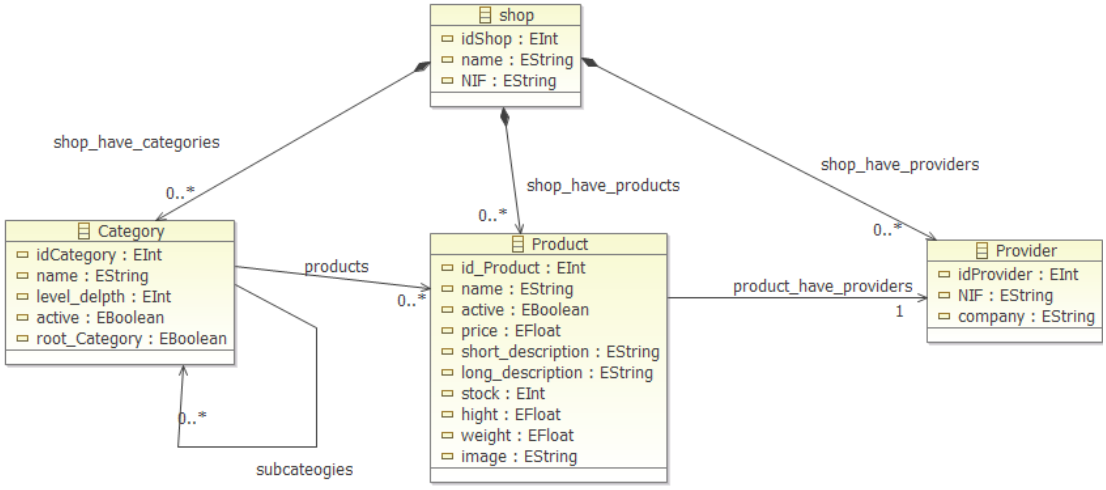
TESIS DOCTORAL: "METAMODELO DE PROCESOS DE EECOMMERCE PARA LOGRAR LA REUTILIZACION Y LA INTEROPERABILIDAD" (Tarazona Bermúdez, 2014)

Este proyecto propone un metamodelo de procesos E-Commerce con el cual se pretende conseguir la reutilización y la interoperabilidad, con el siguiente argumento: "Las soluciones de comercio electrónico disponibles actualmente en el mercado se caracterizan, en general, por el alto coste y la reducida interoperabilidad, esto obedece a las dificultades técnicas que ello implica. De otra parte está el interés delos proveedores de plataformas de comercio electrónico por mantener cuotas de mercado que imposibilitan a sus usuarios la migración de sus tiendas virtuales a otros proveedores. Esta situación dificulta que las empresas incursionen de manera decidida a utilizar internet como un canal de distribución formal".

Por ello se propone el desarrollo de un metamodelo para procesos específicos de comercio electrónico con lo cual se consiga la reutilización y la interoperabilidad, todo esto basado en una ontología de dominio, y la ayuda de principios propuestos por la Ingeniería Dirigida

por Modelos (MDE), específicamente la propuesta de la OMG, Arquitectura Dirigida por Modelos (MDA), para minimizar el grado de complejidad y costo para la gestión de la solución.

Ilustración 3 – Metamodelo proceso eCommerce



Fuente: (Tarazona Bermúdez, 2014)

PARTE 2. DESARROLLO DE LA INVESTIGACIÓN

CAPÍTULO 2. INVESTIGACIÓN PRELIMINAR

En este capítulo se presenta el estudio profundo del Proceso de Peticiones, Quejas y Reclamos en instituciones públicas del estado colombiano como de la normatividad vigente que lo rige, además de la definición de los requerimientos funcionales y no funcionales que debe tener en cuenta en la definición de la arquitectura como de la implementación de la solución que soporte el proceso PQRDs en las entidades del estado colombiano.

2.1. RECOLECCIÓN DE LA INFORMACIÓN

Con el fin de determinar las necesidades básicas y principales que debe contener un sistema integrado de PQRD, este proyecto se centró en diferentes frentes.

Una de esas fuentes se realizó sobre el estudio del Modelo Estándar de Control Interno (MECI) para las entidades del Estado, que se genera tomando como base el Artículo 1° de la Ley 87 de 1.993.

Del MECI podemos mencionar que es una herramienta que tiene como finalidad proporcionar una estructura para el control de la estrategia, la gestión y la evaluación de las entidades del Estado, para orientarlas hacia el cumplimiento de sus objetivos institucionales y su contribución a los fines esenciales del Estado.

En la siguiente grafica podemos evidenciar como el MECI se relaciona con el sistema integrado de PQRD.

Ilustración 4 - MECI y el proceso PQDR

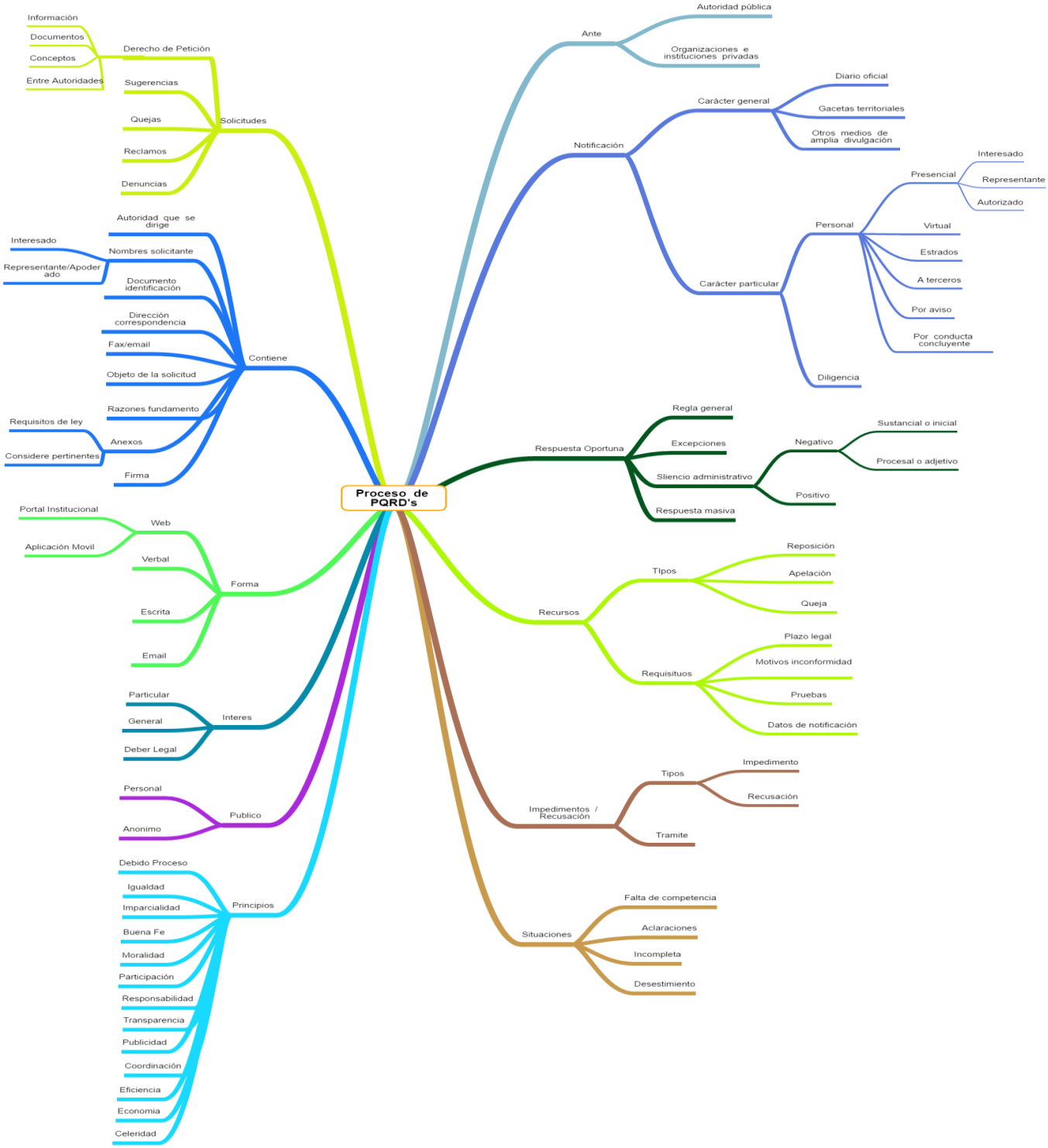
Fuente: (MECI, 2014)

Según lo anterior el sistema integrado PQRD puede relacionarse como una fuente de Información Primaria controlada por el Sistema de Información Tecnológico de cada una de las Entidades.

Otra fuente sobre la cual se realizó el estudio es el Código de Procedimiento Administrativo y de lo Contencioso Administrativo (CPACA) correspondiente a la Ley 1437 del 2011.

Dentro del Código se definen los principios, actores, términos de respuesta, entre otros aspectos para poder llevar un proceso PQRD. En el siguiente diagrama se muestra la distribución inicial de datos como resultado del estudio de la normatividad que rige el proceso de Peticiones, Quejas y Reclamos para las entidades de estado colombiano como de la revisión de casos de estudio de la definición e implantación de este proceso en instituciones públicas a nivel local como nacional, lo cual se identifican y representan en el mapa conceptual:

Ilustración 5 - Mapa conceptual del dominio



Fuente: Propia

El mapa conceptual construido con base al estudio; este nos sirvió en la construcción de la ontología y nos ayudara en la validación de los modelos propuestos por MDA.

Otra fuente sobre la cual se realizó el estudio es la estrategia de Gobierno en Línea correspondiente en el Decreto Único Reglamentario del Sector Tecnologías de la Información y las Comunicaciones 1078 del 2015.

Este Decreto Único comprende el cumplimiento de cuatro (4) propósitos:

- Lograr que los ciudadanos cuenten con servicios en línea de muy alta calidad.
- Impulsar el empoderamiento y la colaboración de los ciudadanos con el Gobierno.
- Encontrar diferentes formas para que la gestión en las entidades públicas sea óptima gracias al uso estratégico de la tecnología.
- Garantizar la seguridad y la privacidad de la información.

Dentro los lineamientos de la estrategia de Gobierno Electrónico, específicamente lo mencionado dentro del Manual de Gobierno en Línea en el apartado de TIC para Servicios en el punto de Sistema integrado de PQRD podemos encontrar los tres (3) frentes que presente cubrir.

El primer frente consiste en: “Busca garantizar que los usuarios cuenten con un canal de atención y comunicación con la entidad a través del sitio web, que permita realizar el seguimiento de PQRD y desarrollar acciones de mejoramiento continuo a partir de la evaluación de la satisfacción del usuario”. Se puede identificar que propone un sistema de tipo web que permite realizar todas las acciones relacionadas con PQRD teniendo en cuenta la ley y demás disposiciones vigentes.

El segundo frente consiste en: “Busca garantizar un canal de atención y comunicación de los usuarios con la entidad a través de tecnologías móviles, facilitando el seguimiento permanente y desarrollando acciones de mejoramiento continuo a partir de la evaluación de la satisfacción del usuario”. Se puede identificar que propone un sistema de tipo móvil que permite realizar todas las acciones relacionadas con PQRD teniendo en cuenta la ley y demás disposiciones vigentes.

El ultimo frente consiste en: “Busca integrar y centralizar las peticiones, quejas, reclamos y denuncias recibidas a través de los diferentes canales habilitados para tal fin y desarrollar acciones de mejoramiento continuo a partir de la evaluación de la satisfacción del usuario”. Se puede identificar que propone un sistema que integre las peticiones realizadas por el sistema de tipo Web y Móvil.

Dentro de estos frentes, la estrategia exige cumplir con los siguientes lineamientos:

- LI.GO.02 - Apoyo de TI a los procesos, el cual exige que debe identificar, definir y especificar las necesidades de sistematización y apoyo tecnológico a los procesos de la institución a partir del mapa de procesos del Modelo Integrado de Planeación y Gestión de la institución, de tal manera que desde su diseño se incorporen facilidades tecnológicas que contribuyan a lograr transversalidad, coordinación, articulación, mayor eficiencia y oportunidad a nivel institucional y sectorial para obtener menores costos, mejores servicios, menores riesgos y mayor seguridad.
- LI.INF.02 - Plan de calidad de los componentes de información, el cual exige que debe contar con un plan de calidad de los componentes de información que incluya etapas de aseguramiento, control e inspección, medición de indicadores de calidad, actividades preventivas, correctivas y de mejoramiento continuo de la calidad de los componentes.
- LI.INF.04 - Gestión de documentos electrónicos, el cual exige que debe contemplar el ciclo de vida de la gestión documental en la Arquitectura de Información.
- LI.INF.06 - Lenguaje común de intercambio de componentes de información, el cual exige el uso del lenguaje común para el intercambio de información con otras instituciones.
- LI.INF.09 - Canales de acceso a los Componentes de información, el cual exige garantizar los mecanismos que permitan el acceso a los servicios de información por parte de los diferentes grupos de interés, contemplando características de accesibilidad, seguridad y usabilidad.

- LI.INF.12 - Fuentes unificadas de información, el cual exige que debe garantizar la existencia de fuentes únicas de información, para que el acceso sea oportuno, relevante, confiable, completo, veraz y comparable.
- LI.INF.14 - Protección y privacidad de Componentes de información, el cual exige que debe incorporar, en los atributos de los Componentes de información, la información asociada con los responsables y políticas de la protección y privacidad de la información, conforme con la normativa de protección de datos de tipo personal y de acceso a la información pública.
- LI.INF.15 - Auditoría y trazabilidad de Componentes de información, el cual exige que debe definir los criterios necesarios para asegurar la trazabilidad y auditoría sobre las acciones de creación, actualización, modificación o borrado de los Componentes de información.
- LI.SIS.01 - Definición estratégica de los sistemas de información, el cual exige que debe definir la arquitectura de los sistemas de información teniendo en cuenta las relaciones entre ellos y la articulación con los otros dominios del Marco de Referencia.
- LI.SIS.07 - Guía de estilo y usabilidad, el cual exige que debe definir una guía de estilo y usabilidad única, que establezca los principios para el estilo de los componentes de presentación, estructura para la visualización de la información y procesos de navegación entre pantallas, entre otros.
- LI.SIS.09 – Interoperabilidad, el cual exige debe habilitar en sus sistemas de información aquellas características funcionales y no funcionales, necesarias para interactuar con la Plataforma de Interoperabilidad del Estado colombiano, partiendo de los flujos de información registrados en el directorio de Componentes de información y las necesidades de intercambio de información con otras instituciones.
- LI.SIS.10 – Implementación de Componentes de información, el cual exige que los sistemas de información deben funcionar sobre la Arquitectura de información definida para la institución y debe dar soporte a los componentes de información allí incluidos.

- LI.SIS.15 - Plan de capacitación y entrenamiento para los sistemas de información, el cual exige que debe contar con planes de capacitación y entrenamiento a los usuarios, que faciliten el uso y apropiación de los sistemas de información.
- LI.SIS.20 - Plan de calidad de los sistemas de información, el cual exige que debe contar con planes de calidad de los componentes de software de sus sistemas de información. Este Plan de Calidad debe formar parte del proceso de desarrollo de software.
- LI.SIS.21 - Criterios no funcionales y de calidad de los sistemas de información, el cual exige que debe tener en cuenta los requerimientos de la institución, las restricciones funcionales y técnicas, y los atributos de calidad.
- LI.SIS.22 - Seguridad y privacidad de los sistemas de información, el cual exige que debe incorporar aquellos componentes de seguridad para el tratamiento de la privacidad de la información, la implementación de controles de acceso, así como los mecanismos de integridad y cifrado de la información.
- LI.SIS.23 - Auditoría y trazabilidad de los sistemas de información, el cual exige que debe tener en cuenta mecanismos que aseguren el registro histórico para poder mantener la trazabilidad de las acciones realizadas por los usuarios.

Gracias a esta última fuente podemos tener más claros cuales con los lineamientos a seguir contenidos dentro del Marco de Referencia de Arquitectura TI definido por el estado Colombiano.

Adicional a las fuentes mencionadas, se realizó una revisión a las páginas de algunas entidades del Estado con el fin de determinar que otros conceptos, actores, términos, tiempos, lineamientos debiéramos tener en cuenta.

La revisión tuvo en cuenta lo siguiente:

- Información acerca de una PQRD
- Información de contacto
- Sección de preguntas frecuentes
- Informes de gestión PQRD

- Proyectos resultado de la evaluación de las PQRD
- Registro de la PQRD
- Seguimiento de la PQRD

Las entidades tenidas en cuenta fueron:

- Ministerio de Tecnologías de la Información y las Comunicaciones - MinTic
- Superintendencia de Sociedades - SuperSociedades
- UDistrital - Universidad Distrital Francisco José de Caldas
- Superintendencia de Puertos y Transporte - SuperTransporte
- Instituto Colombiano de Crédito Educativo y Estudios Técnicos en el Exterior - ICETEX

Teniendo en cuenta los aspectos para la revisión y las entidades se presenta a continuación una tabla donde refleja el resultado:

Ilustración 6 - Resultado de la revisión

Puntos a revisar / Entidad	Información acerca de una PQRD	Información de contacto	Sección de preguntas frecuentes	Informes de gestión PQRD	Proyectos resultado de la evaluación de las PQRD	Registro de la PQRD	Seguimiento de la PQRD
MinTic	Cuenta con la información	Cuenta con la información	Cuenta con la información	Cuenta con la información	Cuenta con la información, pero se encuentra dentro de los informes	Cuenta con el registro	Cuenta con la opción de consulta
SuperSociedades	Cuenta con la información	Cuenta con la información	No cuenta con la información	Cuenta con la información	No cuenta con la información	Cuenta con el registro	No se pudo ubicar la opción
UDistrital	Cuenta con la información	Cuenta con la información	No cuenta con la información	Cuenta con la información	Cuenta con la información	Cuenta con el registro	Cuenta con la opción de consulta
SuperTransporte	No cuenta con la información	Cuenta con la información	Cuenta con la información	No cuenta con la información	No cuenta con la información	No cuenta con el registro	No se pudo ubicar la opción
ICETEX	Cuenta con la información	Cuenta con la información	Cuenta con la información	No se pudo ubicar la opción	No se pudo ubicar la opción	Cuenta con el registro	Cuenta con la opción de consulta

Fuente: Propia

2.2. REQUERIMIENTOS FUNCIONALES

Después de la actividad de recolección de la información podemos continuar con la definición de los requerimientos funcionales para el sistema. Se entiende como requisitos funcionales aquellos que describen las funciones que el software va a ejecutar; por ejemplo ajustarse a un formato de texto o modular una señal. Se conocen también como capacidades.

Estos requerimientos se presentaran con ayuda de las siguientes herramientas:

2.2.1. LISTA DE REQUERIMIENTOS DE ALTO NIVEL

Para este punto se definieron la lista de requerimientos de alto nivel y la lista de actores.

Ilustración 7 - Lista de requerimientos de alto nivel

ID	Requerimiento	Actores	Prioridad
RQ1	Registrar la PQRD	Ciudadano	Alta
RQ2	Consultar la PQRD	Ciudadano Profesional Universitario	Alta
RQ3	Responder la PQRD	Profesional Universitario Lider Funcionario	Alta
RQ4	Realizar Seguimiento a la PQRD	Profesional Universitario	Alta
RQ5	Crear Hoja de Ruta	Líder	Media
RQ6	Responder la PQRD con base a una Hoja de Ruta	Profesional Universitario	Media
RQ7	Asignar la PQRD al area encargada	Funcionario	Alta
RQ8	Consultar lista de PQRD	Funcionario	Alta
RQ9	Cambiar el estado a la PQRD en el seguimiento	Funcionario	Alta
RQ10	Generar Informe de PQRD	Lider	Media
RQ11	Publicar información relacionada con PQRD	Profesional Universitario	Baja
RQ12	Publicar información de contacto para la atención de la PQRD	Profesional Universitario	Baja
RQ13	Crear Pregunta Frecuente	Profesional Universitario	Baja
RQ14	Editar Pregunta Frecuente	Profesional Universitario	Baja
RQ15	Consultar Pregunta Frecuente	Ciudadano	Baja

Fuente: Propia

Ilustración 8 - Lista de actores

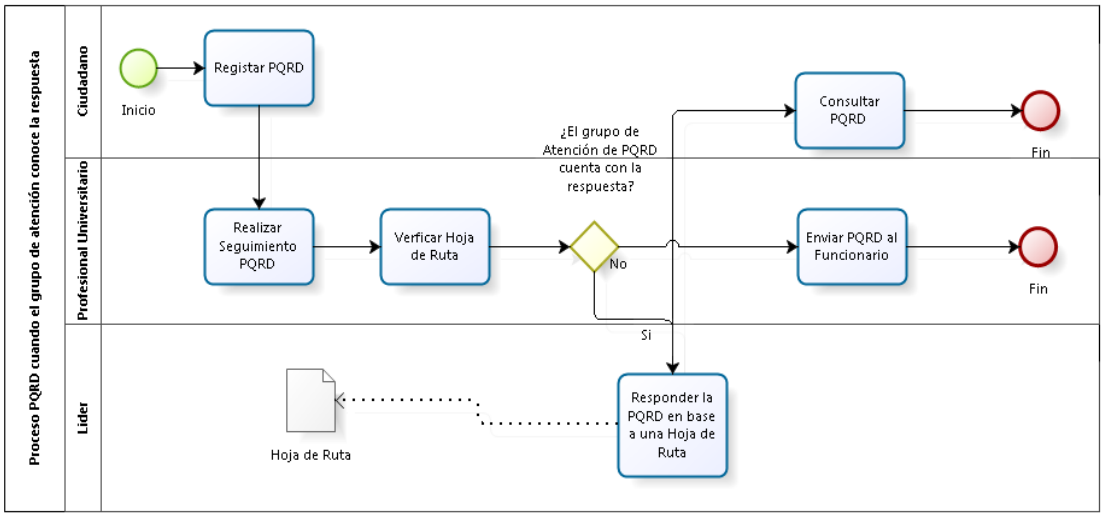
ID	Nombre Actor
1	Ciudadano
2	Lider
3	Profesional Universitario
4	Funcionario

Fuente: Propia

2.2.2. MODELO BPM DE ALTO NIVEL

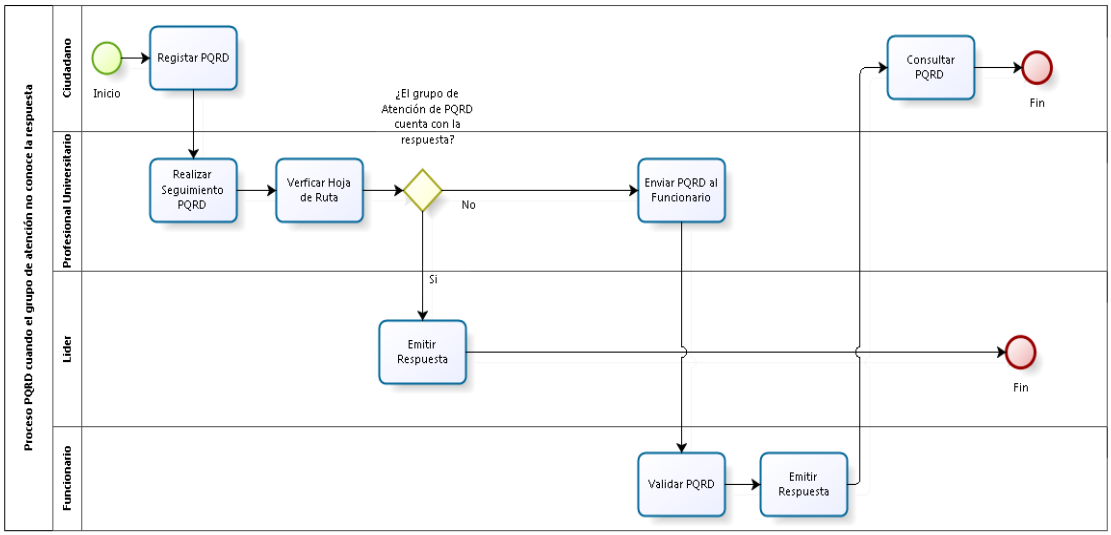
Para este punto se definieron dos (2) diagramas utilizando BPM; el primero para el proceso cuando el grupo de personas que hacen parte de la atención de las PQRD conoce la respuesta y el segundo para cuando no tiene la respuesta.

Ilustración 9 - Proceso PQRD cuando se conoce la respuesta



Fuente: Propia

Ilustración 10 - Proceso PQRD cuando NO se conoce la respuesta



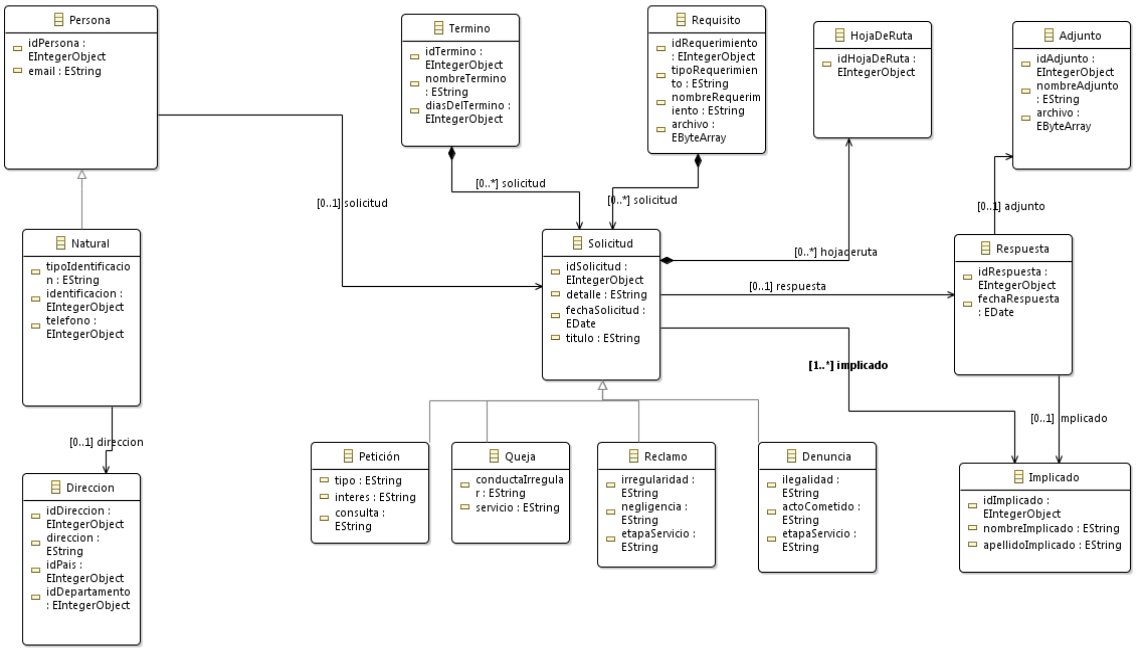
Fuente: Propia

2.2.3. MODELO DE DOMINIO

El modelo de dominio (o también llamado modelo conceptual) se crea con el fin de representar el vocabulario y los conceptos clave del dominio del problema. También se puede definir como una representación de clases conceptuales del mundo real, no de componentes de software. Dentro de este modelo se presentan las relaciones entre los conceptos.

A continuación el modelo de dominio definido para las PQRD con ayuda de la tecnología EMF de Eclipse Project:

Ilustración 11 - Modelo Dominio PQRD



Fuente: Propia

2.3. REQUERIMIENTOS NO FUNCIONALES

A continuación se listan los requerimientos no funcionales que deben ser tenidos en cuenta:

- Administración:
  - ✓ El acceso a las funcionalidades de administración y configuración se deben realizar dentro de las instalaciones de la Entidad.
- Disponibilidad:
  - ✓ El sistema debe contar con una disponibilidad máxima proponiendo un 95 % con un tiempo de atención por semana de 9 horas X 6 días.
- Capacidad:
  - ✓ El tiempo máximo de respuesta por transacción debe ser de 5 segundos.

- ✓ El sistema debe poder encriptar y utilizar firmas digitales para la comunicación y el intercambio de información entre el Ciudadano y la Entidad.
- ✓ El sistema debe contemplar el ciclo de vida de la gestión documental de aquellos documentos que participan dentro de los procesos que apoya el sistema.

- Usabilidad:
  - ✓ El sistema debe ofrecer herramientas de Ayudas Contextuales
  - ✓ El sistema debe ofrecer soporte para el uso de la herramienta Convertic[1] a los usuarios que presenten discapacidad visual, al momento de utilizarlo.
  - ✓ El sistema debe presentar un mensaje de información al usuario cuando presenten algún error de conectividad o del sistema mismo.
  - ✓ El sistema debe presentar la ayuda o guía necesaria en aquellas funcionalidades que se consideren críticas.
  - ✓ El sistema debe adaptar la apariencia de las páginas web al dispositivo que se esté utilizando para visualizarla (Responsive Web Design).
- Escalabilidad:
  - ✓ El sistema debe permitir cambios de hardware y software (hacia atrás o hacia adelante) con el fin de soportar una carga determinada teniendo en cuenta el coste, tiempo y complejidad.
  - ✓ El sistema debe no debe estar diseñado para que funcione en una sola combinación de hardware y software.
- Mantenibilidad:
  - ✓ El sistema debe tener mecanismos de adaptación e identificación de nuevas funcionalidades o componentes.

[1] Descripción de la herramienta en http://www.mintic.gov.co/portal/604/w3-article-5139.html

- Desempeño:
  - ✓ El sistema debe utilizar los recursos de manera óptima. Esto se mide teniendo en cuenta las medidas que ya el proveedor ha realizado (whitepapers).
- Seguridad:
  - ✓ El sistema debe garantizar la autenticación de usuarios.
  - ✓ El sistema debe garantizar la autorización de usuarios.
  - ✓ El sistema debe poseer mecanismos para la prevención y detección de intrusos, gestión de vulnerabilidades, ataques.
  - ✓ El sistema debe incorporar la información asociada con los responsables y políticas de la protección y privacidad de la información, conforme con la normativa de protección de datos de tipo personal y de acceso a la información pública.
- Auditoría y Control:
  - ✓ El sistema debe llevar el registro de las actividades de cada usuario. Llámese actividad a operaciones relacionadas con creación, consulta, modificación y eliminación de registros.
  - ✓ El sistema debe llevar el registro en un log que almacene como mínimo: modulo afectado, fecha hora, nivel de la actividad, descripción, registro antes del cambio y registro después de cambio.

CAPÍTULO 3. CONSTRUCCIÓN DEL MODELO ONTOLÓGICO

La especificación de un metamodelo independiente de la plataforma (PIM) basado en una ontología de dominio permite la categorización del conocimiento sobre el dominio del proceso de Peticiones, Quejas y Reclamos (Atención al Ciudadano) definido para las entidades del estado colombiano, lo que permite modelar una solución de software genérica que minimice el tiempo de desarrollo como que permita el aumento de la reutilización como de la interoperabilidad del producto de este trabajo.

En las secciones siguientes se describen las etapas de la solución al problema planteado en el proyecto:

La familiarización con el Proceso de Peticiones, Quejas y Reclamos en instituciones públicas del estado colombiano teniendo como base la normatividad vigente sobre el tema, la selección de la herramienta de construcción de ontologías de acuerdo a las propuestas en el anteproyecto como del lenguaje utilizado y los principios de diseño del metamodelo.

En la sección 2.2 Desarrollo Metodológico se presenta el desarrollo de la ontología siguiendo la metodología "**Ontology Development 101**" propuesta en (Natalya F. Noy, 2000).

3.1. FASE INICIAL

En esta fase se realizó el estudio profundo de metodologías, lenguajes y herramientas de la Web Semántica que más se adecuaron a nuestro proyecto para llegar a la definición de las más pertinentes en cada caso, como se describe a continuación.

3.1.1. ESTUDIO DEL DOMINIO

El estudio del dominio definido en el proyecto dejo en evidencia que la mayor parte del proceso se rige bajo las mismas normas y bajo mismos preceptos, y que de entidad a entidad si bien existen variaciones se evidencia no una diferenciación sustancial en el marco de referencia sobre el cual trabajan más bien sobre el flujo de trabajo que siguen, por tal razón este ayuda a ratificar nuestra hipótesis inicial de establecer una definición global, estandarizada y consistente que permita la implementación de sistemas de información que respalde el proceso en cuestión en las entidades del Estado Colombiano (Ver detalle en Capitulo 2- Investigación Preliminar).

3.1.2. PRINCIPIOS DE DISEÑO DEL MODELO ONTOLÓGICO

Se tuvieron en cuenta los siguientes principios de diseño como base para la construcción de la ontología y que se tuvieron en cuenta durante el desarrollo.

- **Abstracción**: Teniendo en cuenta el dominio del proyecto (Proceso de Peticiones, Quejas y Reclamos) y el carácter genérico que se plantea para la arquitectura (aplicable a todas las entidades del estado colombiano), es necesario mantener el modelo construido lo más simple posible sin perder la completitud ni la capacidad de expresión por medio de una ontología refinada lo más posible.

- **Jerarquía**: Las clases definidas por medio de los conceptos contemplados para la Ontología, deben seguir una forma taxonómica donde cada nodo debe ser hijo de un concepto más abstracto además ordenados en relación a la pertinencia entre ellos.

- **Flexibilidad**: Debe existir la posibilidad de adaptar el metamodelo ontológico construido a nuevas necesidades con la posibilidad de extenderlo o modificarlo sin que esto represente un coste elevado en la construcción de la aplicación software producto de la definición de la Arquitectura Dirigida por Modelos planteada.

3.1.3. HERRAMIENTA SELECCIONADA: PROTÉGÉ

La herramienta seleccionada como apoyo a la construcción de la ontología de dominio es Protégé por su amplia utilización académica como a nivel productivo, en implementación de soluciones de conocimiento, como por su amplia documentación y soporte, y por su intuitiva interfaz gráfica además del hecho de que es una herramienta libre y de software abierto basado en Java, que permite además la utilización de plugins externos que aumentan su funcionalidad.

Otras características importantes de la herramienta son:

- Compatible con estándares del W3C
- Interfaz de usuario personalizable
- Visualización de apoyo
- Apoyo a refactorización ontología
- Interfaz directa con razonadores

Además, se instalaron y utilizaron los plugins ***OWLViz 4.1.2*** (Horridge, 2010) y ***OntoGraf 1.0.1*** (Falconer, 2010) que permiten navegar de forma interactiva sobre las relaciones y clases de la ontología permitiendo filtrar los nodos para una mejor visualización, como del razonador ***FaCT++*** (Dmitry, 2004) que nos permite evaluar las relaciones existentes entre las clases de la ontología para el lenguaje OWL-DL.

3.1.4. LENGUAJE SELECCIONADO: OWL DL

En el marco conceptual se definió OWL como el lenguaje de marcado para publicar y compartir datos usando ontologías, y se explicó las diferencias existentes entre las tres variantes de este lenguaje recomendado del W3C (W3C).

De las variantes definidas por el lenguaje, fue seleccionado OWL DL, ya que posee todo el vocabulario OWL completo, permitiendo máxima expresividad mientras conserva la propiedad de ser computable y resoluble, y provee las propiedades necesarias para la

definición del modelo como son la exclusión mutua de clases, los operadores de unión y complemento, las restricciones de cordialidad y propiedades de exclusión, entre otras.

3.2. DESARROLLO METODOLÓGICO

La idea central de la sección es obtener un modelo ontológico robusto y en lo posible estático pero simple para cumplir con el propósito de metamodelo genérico del proceso de Peticiones, Quejas y Reclamos para las entidades del estado colombiano, por tal razón se hace uso de la guía propuesta para la construcción de ontologías OWL en Protégé (Stanford, 2016), ya que se apoya sobre ejemplos prácticos utilizando la herramienta, además, de que ilustra el paso a paso del desarrollo de una ontología genérica.

A continuación se describe el proceso definido por la guía definida en "**Ontology Development 101**" (Natalya F. Noy, 2000), metodología que plantea una estrategia iterativa de desarrollo en 7 fases, que presenta gran flexibilidad y simplicidad, para la ontología de dominio del Proceso de PQRs.

3.2.1. DOMINIO Y ALCANCE DE LA ONTOLOGÍA

EL dominio de la ontología es el Proceso de Peticiones, Quejas y Reclamos para las Entidades del Estado Colombiano, de tal forma que cualquier institución que cumpla labores públicas y/o requiera regirse bajo la normatividad vigente respecto a este tema estarán en condiciones de hacer uso del modelo construido gracias al nivel de abstracción propuesto en los principios de diseño del metamodelo.

Por lo tanto, se tiene como objetivo: identificar y plasmar los conceptos principales y sus relaciones que cubran de manera genérica y única los distintos subprocesos que engloban en las organizaciones el dominio de las PQRs.

3.2.2. REUTILIZACIÓN DE ONTOLOGÍAS EXISTENTES

En el desarrollo ontológico es necesario realizar una revisión previa de trabajos realizados sobre el mismo dominio o con un propósito similar con el fin de llegar a encontrar similitudes y diferencias entre estos y el que se está realizando, para tal caso de reusar lo máximo posible ya sea por la definición propuesta, su construcción o su propósito. Para este caso, se consultaron trabajos similares y sobre el mismo dominio el cual arrojo pocos resultados relevantes como los presentados en (Servetto & López), (PJoanneum & Salhofer, 2012) o en (Zdravkovic, 2010).

Además, se consultaron diseños ontológicos genéricos para *e-Government*, por medio de una propuesta centrada en datos de gobierno de EE.UU en (OE-Gov, 2010), que no fueron útiles para nuestro propósito.

De la investigación realizada al no encontrarse trabajos útiles y potenciales, que permitieran hacer reúso de las ontologías se procede con el desarrollo del modelo semántico desde cero.

3.2.3. TÉRMINOS IMPORTANTES EN LA ONTOLOGÍA

Luego de definir el dominio y de acuerdo con lo encontrado con la revisión del estado del arte, se proponen una serie de términos propios y generales en el proceso de las Peticiones, Quejas y Reclamos.

A continuación se presenta la terminología relevante para el dominio del proyecto, sobre estos términos se construyeron las clases y relaciones del modelo ontológico.

Ilustración 12 - Términos importantes

Términos	
Actor	Hoja de Ruta
Notificación	Interés
Presentación	Recurso
Respuesta	Requisitos
Solicitud	Denuncia
Derecho de Petición	Queja
Reclamo	Sugerencia
Anónimo	Personal
Email	Escrita
Verbal	Web
Deber legal	General
Particular	Autoridad Pública
Institución Privada	Apelación
Reposición	Anexos
Autoridad	Pruebas
Motivo Inconformidad	Objeto
Pruebas	Respuesta Masiva
Solicitante	Regla General
Silencio Administrativo	Aclaraciones
Desistimiento	Falta de Competencia

Fuente: Propia

3.2.4. CLASES Y JERARQUÍA

En esta fase se definen las entidades del modelo a partir de la información recolectada, apoyados del mapa conceptual definido en la sección de Estudio de Dominio (Ilustración 3 – Mapa conceptual del dominio), y de forma iterativa se llega a la definición final de una taxonomía para el dominio, para lo cual se siguió una estrategia inicial de construcción de forma top-down y en iteraciones posteriores un refinamiento de esta en forma bottom-up.

3.2.4.1. Taxonomía de primer nivel de la ontología

La taxonomía a nivel superior defina está compuesta por las clases: *Actor, Forma, Hoja de Ruta, Interés, Notificación, Presentación, Recurso, Requisitos, Respuesta, Situaciones y Solicitud*.

Ilustración 13 - Taxonomía de primer nivel de la ontología

Fuente: Propia

3.2.4.2. Taxonomía para Actor

La taxonomía correspondiente a los Actores se compone de: *Anónimo, Personal.*

Ilustración 14 - Taxonomía para Actor

Fuente: Propia

3.2.4.3. Taxonomía para Forma

La taxonomía correspondiente a la Forma de Solicitud se compone de: *Email, Escrita, Verbal y Web*.

Ilustración 15 - Taxonomía para Forma

Fuente: Propia

3.2.4.4. Taxonomía para Interés

La taxonomía correspondiente al Interés de la solicitud se compone de: *Deber legal, General y Particular*.

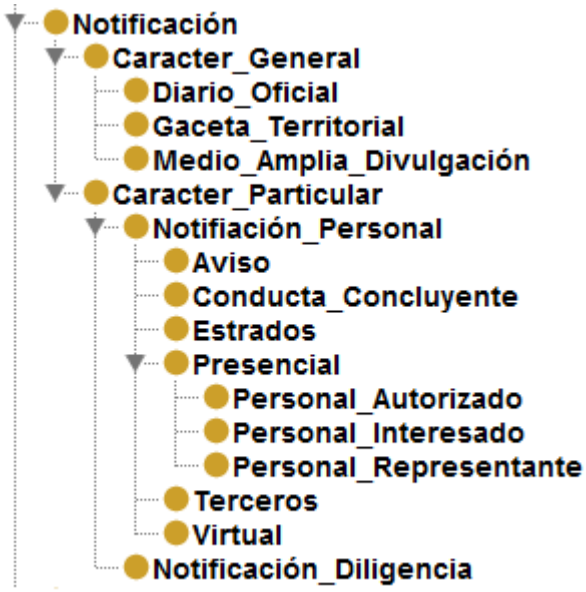
Ilustración 16 - Taxonomía para Interés

Fuente: Propia

3.2.4.5. Taxonomía para Notificación

La taxonomía correspondiente a la Notificación de la respuesta se compone de: *Carácter General, Carácter Particular*.

Ilustración 17 - Taxonomía para Notificación

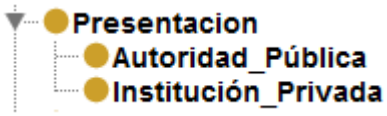


Fuente: Propia

3.2.4.6. Taxonomía para Presentación

La taxonomía correspondiente a la Presentación de la solicitud se compone de: *Autoridad Pública, Institución Privada*.

Ilustración 18 - Taxonomía para Presentación

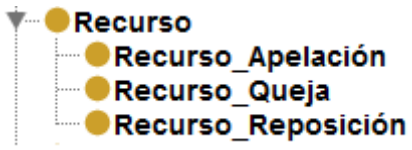


Fuente: Propia

3.2.4.7. Taxonomía para Recurso

La taxonomía correspondiente al Recurso contra los Actos Administrativos se compone de: *Apelación, Queja y Reposición.*

Ilustración 19 - Taxonomía para Recurso



Fuente: Propia

3.2.4.8. Taxonomía para Requisitos

La taxonomía correspondiente a los Requisitos se compone de: *Anexos, Autoridad, Dirección, Firma, Identificación, Motivo Inconformidad, Objeto, Pruebas, Razones y Solicitante.*

Ilustración 20 - Taxonomía para Requisitos

Fuente: Propia

3.2.4.9. Taxonomía para Respuesta

La taxonomía correspondiente a las Respuestas se compone de: *Regla General, Excepciones, Respuesta Masiva, Silencio Administrativo*.

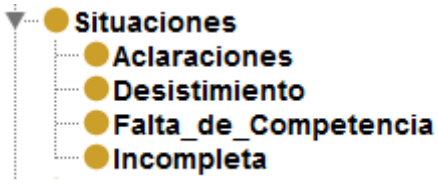
Ilustración 21 - Taxonomía para Respuesta

Fuente: Propia

3.2.4.10. Taxonomía para Situaciones

La taxonomía correspondiente a las Situaciones excepcionales se compone de: *Aclaraciones, Desistimiento, Falta de Competencia, Incompleta*.

Ilustración 22 - Taxonomía para Situaciones



Fuente: Propia

3.2.4.11. Taxonomía para Solicitud

La taxonomía correspondiente a las Solicitudes se compone de: *Denuncia, Derecho de Petición, Queja, Reclamo, Sugerencia.*

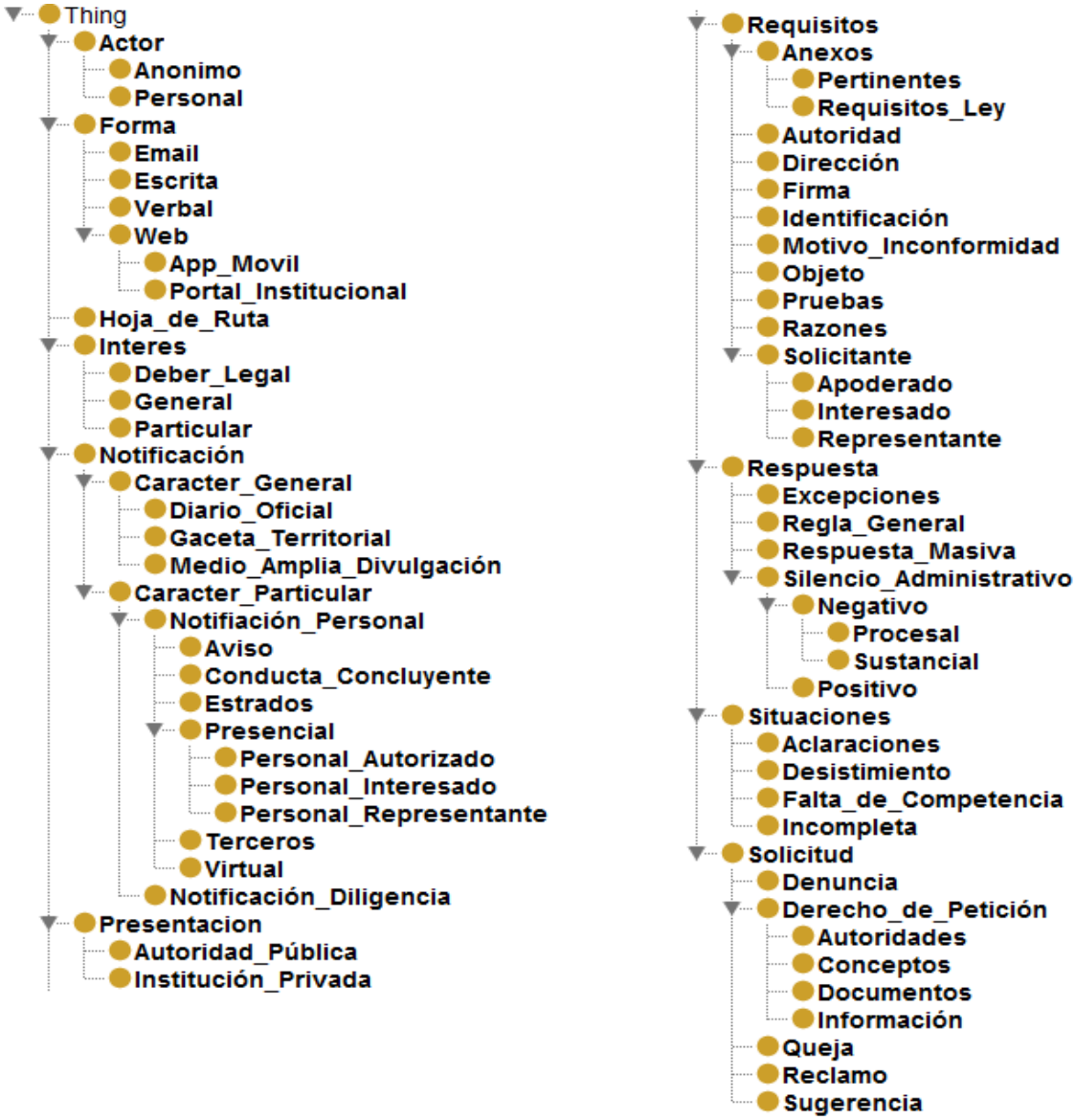
Ilustración 23 - Taxonomía para Solicitud

Fuente: Propia

3.2.4.12. Taxonomía de la Ontología

La definición completa del modelo ontológico definido gracias a las modificaciones y refinamientos efectuados en las iteraciones realizadas en esta construcción se refleja en la siguiente figura:

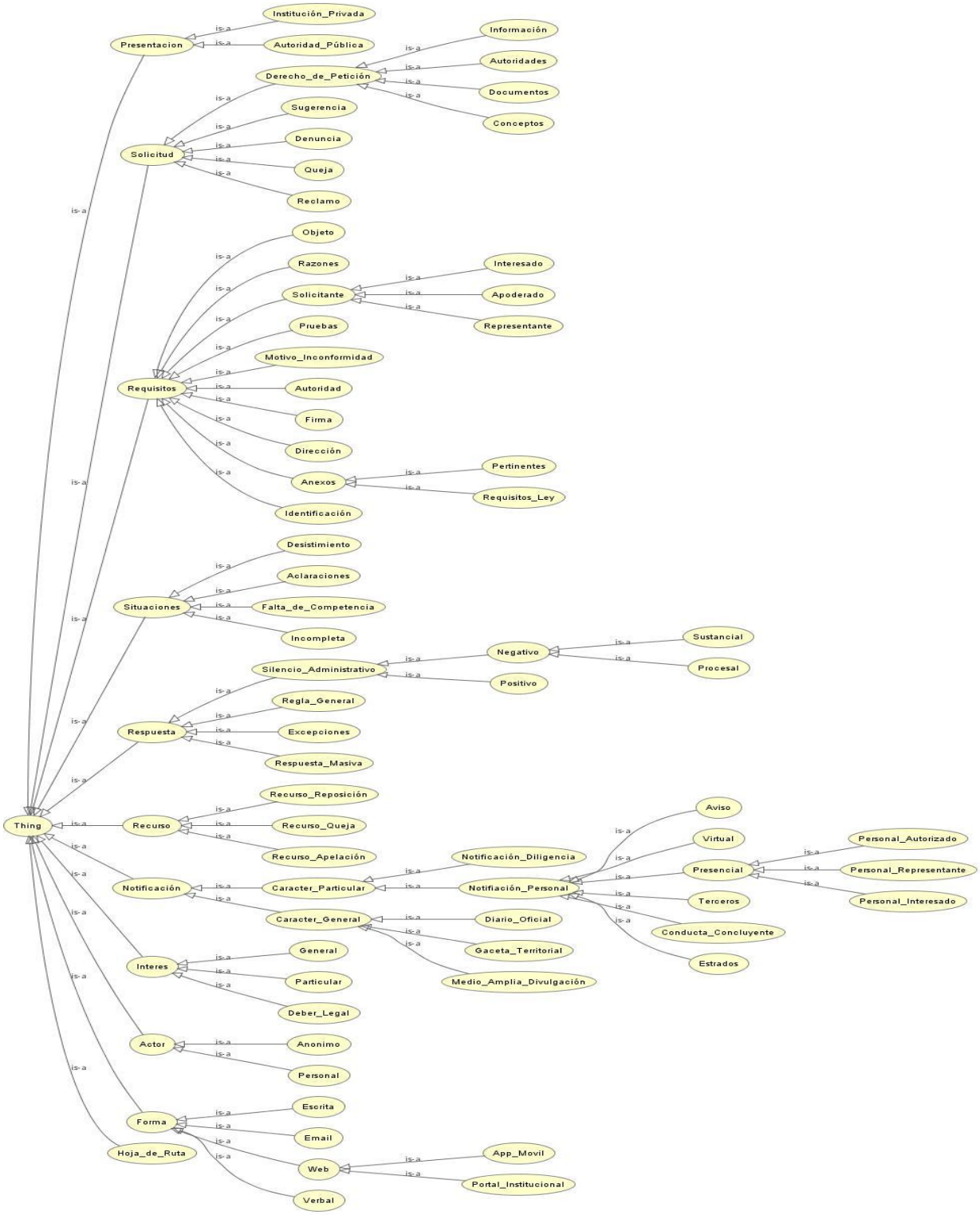
Ilustración 24 - Taxonomía de la Ontología



Fuente: Propia

En la Figura siguiente se realiza una descripción detallada de las clases y subclases que componen el modelo ontológico, esta taxonomía construida consta de 76 clases, y la visualización de estas clases esta graficada por medio del plugin de Protégé OWLViz.

Ilustración 25 - Taxonomía grafica de la ontología



Fuente: Propia

3.2.5. PROPIEDADES

Las propiedades sirven para definir la estructura interna de las clases definidas y su interrelación, estas fueron definidas simultáneamente al proceso de construcción de la taxonomía de la ontología.

3.2.5.1. PROPIEDADES DE DATOS

Corresponden a los atributos propios de cada clase, cada una de estas propiedades posee un Dominio y un Rango, donde el Dominio es la definición de esta propiedad y el Rango, se restringe a representar el tipo de dato que debe poseer las instancias de esta clase.

En la Figura siguiente se detalla la estructura de propiedades de datos del modelo, las cuales fueron ordenadas en una jerarquía de propiedades con el fin de proporcionar extensibilidad del modelo de datos. Algunas propiedades de datos poseen como dominio una clase general (de primer nivel), de las que las clases hijas heredan esta propiedad, además existen otras propiedades genéricas, que comparten varias de las clases del modelo.

Ilustración 26 - Propiedades de datos

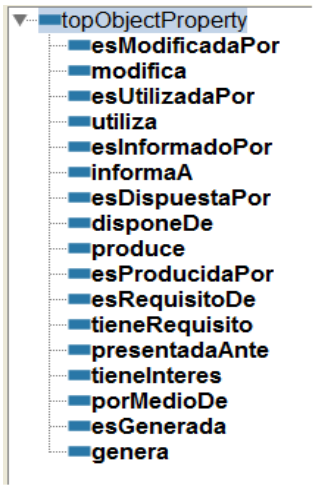
Fuente: Propia

3.2.5.2. PROPIEDADES DE OBJETOS

Corresponden a las relaciones entre dos clases expresando algún tipo de nexo, donde cada propiedad de objeto especifica una entidad (o conjunto de entidades) como Dominio, y otra entidad (o conjunto de entidades) como Rango.

En la Figura siguiente se observan las relaciones definidas en el modelo ontológico, donde la mayoría de relaciones de tipo binaria, e inversa funcional.

Ilustración 27 - Propiedades de objetos

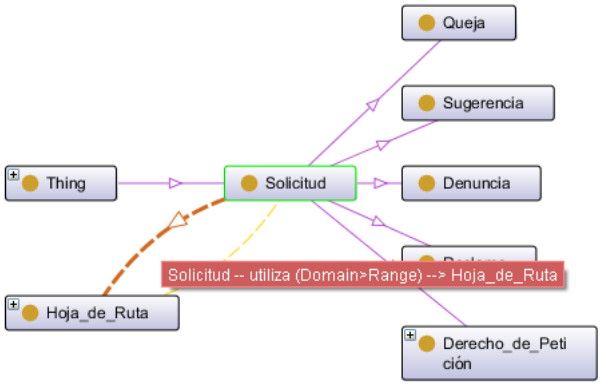


Fuente: Propia

Para realizar la definición de propiedades existentes del modelo ontológico, se establecieron las relaciones para las clases de primer nivel de acuerdo al listado anterior de la siguiente forma:

- Para la clase de *Solicitud* que se relaciona con la clase *Hoja de Ruta* se manejan las propiedades “utiliza” y “esUtilizadaPor”.

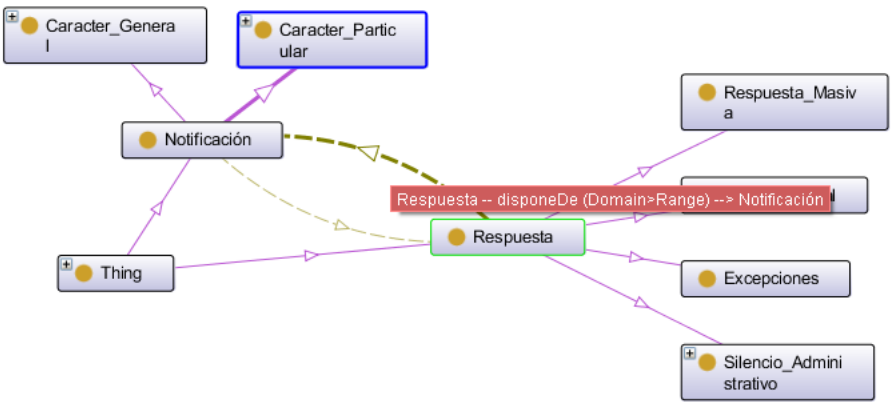
Ilustración 28 - Relación utiliza



Fuente: Propia

- Para la clase de *Respuesta* que se relaciona con la clase *Notificación* se manejan las propiedades “disponeDe” y “esDispuestaPor”.

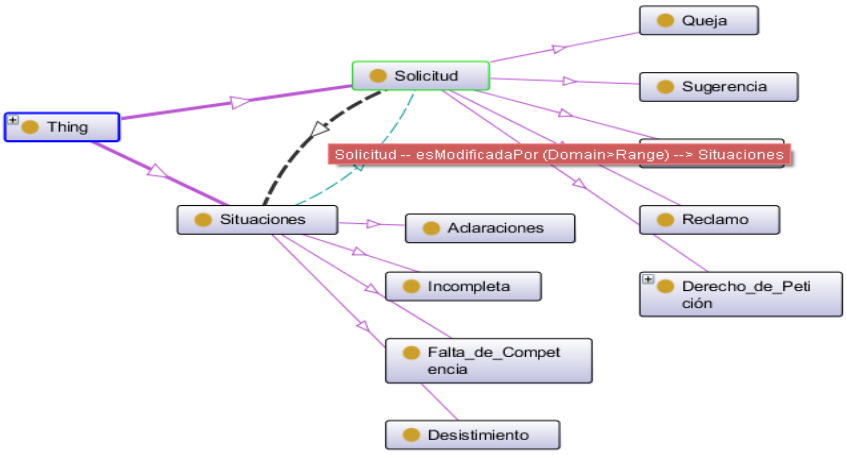
Ilustración 29 - Relación DisponeDe



Fuente: Propia

- Para la clase de *Solicitud* que se relaciona con la clase *Situaciones* se manejan las propiedades “modifica” y “esModificadaPor”.

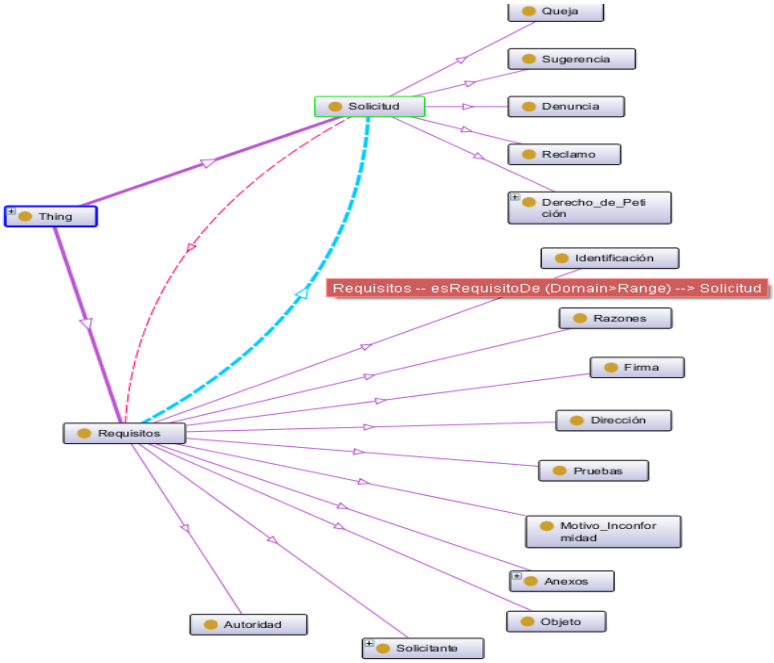
Ilustración 30 - Relación esModificadaPor



Fuente: Propia

- Para la clase de *Solicitud* que se relaciona con la clase *Requisitos* se manejan las propiedades “requiere” y “esRequisitoDe”.

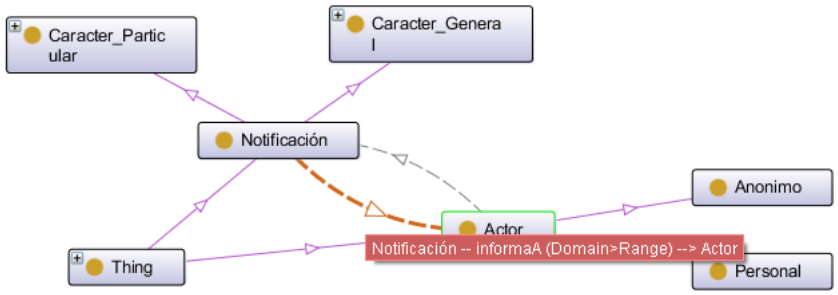
Ilustración 31 - Relación eRequisitoDe



Fuente: Propia

- Para la clase de *Notificación* que se relaciona con la clase *Actor* se manejan las propiedades “informaA” y “esInformadoPor”.

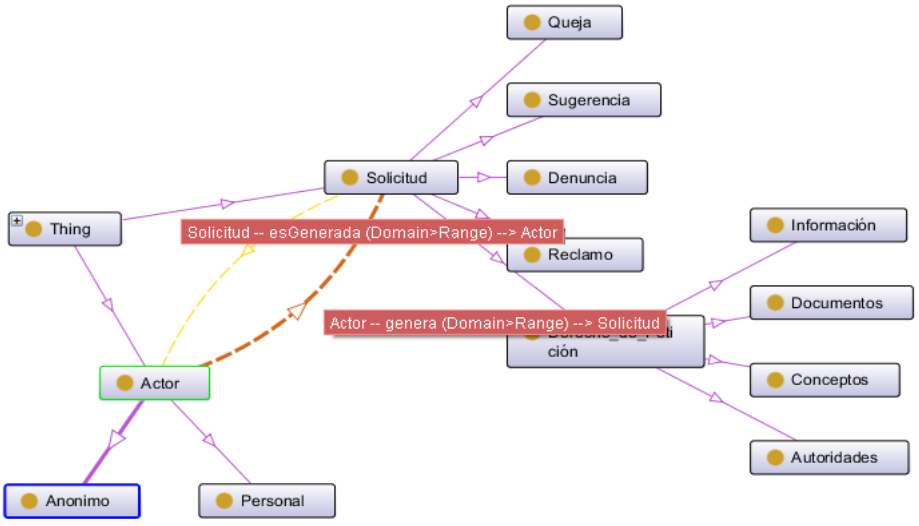
Ilustración 32 - Relación informaA



Fuente: Propia

- Para la clase de *Solicitud* que se relaciona con la clase *Actor* se manejan las propiedades “generea” y “esGeneradaPor”.

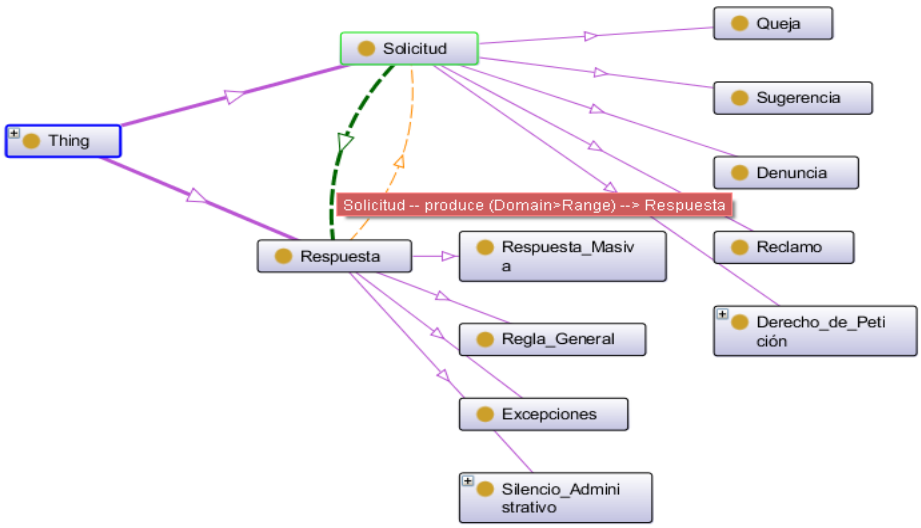
Ilustración 33 - Relación esGenerada



Fuente: Propia

- Para la clase de *Solicitud* que se relaciona con la clase *Respuesta* se manejan las propiedades “produce” y “esProducidaPor”.

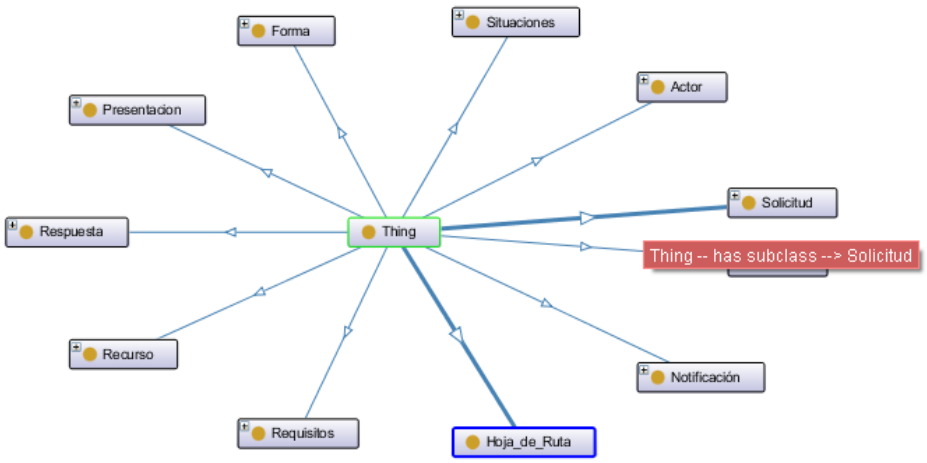
Ilustración 34 - Relación produce



Fuente: Propia

Para todas las clases del modelo sin importar el nivel en el que se encuentre tiene una propiedad (relación común) que se origina desde la clase principal de la ontología (Thing), y termina con la clase de ultimo nivel, esta relación a que se hace referencia es la de *has subclass*, generada al crear la taxonomía de la ontología.

Ilustración 35 - Relación hasSubclass



Fuente: Propia

3.2.6. RESTRICCIONES

Las restricciones se componen de restricciones de tipo de valor, de valores admitidos, cardinalidad, de dominio y de rango y otras de comportamiento de relaciones.

Las restricciones establecidas para el modelo correspondiente a la *Exclusión Mutua de Clases*, determina que si las clases A y B son excluyentes, una instancia de A no puede ser instancia de B. Son clases de este tipo las Solicitudes (**Denuncia, Derecho de Petición, Queja, Reclamo, Sugerencia**) y sus subclases.

Existen además restricciones de cardinalidad de algunas propiedades, como es el caso de la clase **Solicitud** y **Respuesta** donde una Solicitud solo *Produce* una Respuesta, donde posee una restricción de cardinalidad de 1, pero una Respuesta *es Producida Por* mínimo una Solicitud y máximo 10.

Las restricciones para las propiedades de datos corresponden a la especificación de clases como *Dominio* y los tipos de datos como *Rango*, para cada una de las clases definidas en el modelo ontológico.

3.2.7. CREACIÓN DE INSTANCIAS

Una vez descrito completamente el modelo semántico se procede a generar un conjunto de instancias que sirvan para poblar la ontología con la información detallada de lo recolectado en las fases anteriores de estudio del dominio y que funcionaran como la información parametrizada en el momento de la construcción de la aplicación.

Para el caso de la Ontología PQRDs se omite la instanciación puesto que el poblado será ejecutado en el momento de construcción de la aplicación que soportara el Proceso de Peticiones, Quejas y Reclamos en las entidades del estado colombiano.

CAPÍTULO 4. PROPUESTA DE ARQUITECTURA DIRIGIDA POR MODELOS

4.1. INTRODUCCION

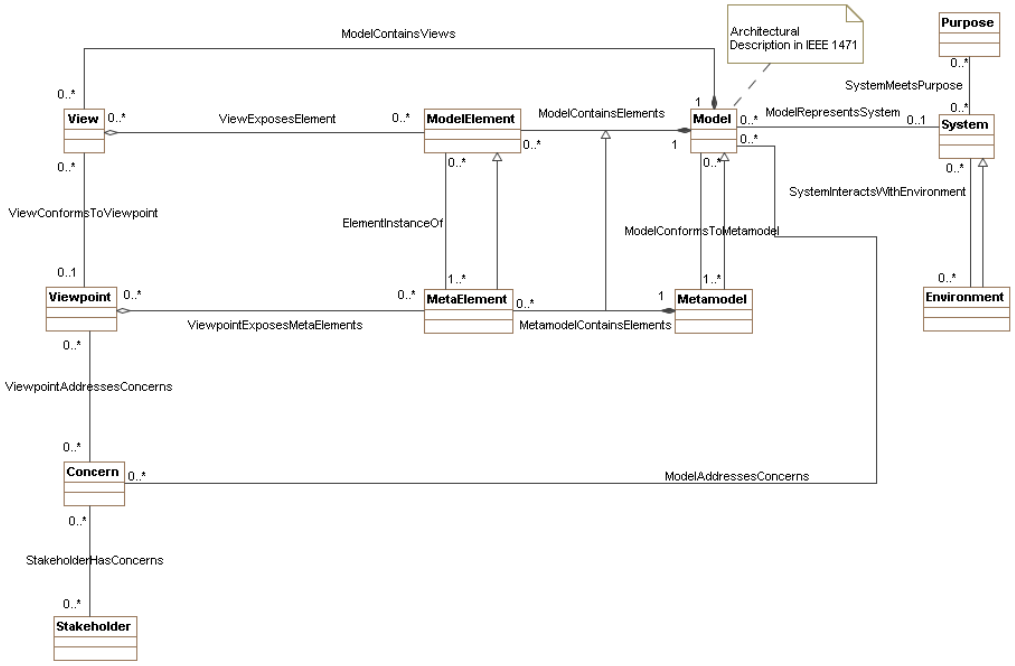
Como producto de las mejores prácticas en Ingeniería de Software surge la arquitectura dirigida por modelos MDA (o en ingles Model-Driven Architecture). Este movimiento es promovido por la OMG a partir del año 2001. Dentro de la promoción se desarrolló la guía de MDA, en el 2001 aparece la versión 1 y en el 2014 aparece la versión 2.

Podemos mencionar que MDA no es un proceso de desarrollo, no es una implementación o simplemente no es una generación de código fuente. MDA es un concepto que propone basar el desarrollo de software en modelos específicos utilizando diferentes tecnologías como UML, XMI, MOF, entre otros. MDA es solamente otro pequeño paso en el largo camino para el mejoramiento de la ingeniería como disciplina.

MDA contiene conceptos con el cual trabaja y es bueno tenerlos claros antes de comenzar a trabajar para tener un mayor entendimiento y tener el contexto donde se desenvuelve.

- *System*: Este concepto se presenta teniendo en cuanto que algún sistema ya existe o se está planeando. Un sistema puede incluir cualquier cosa desde un programa, la combinación de diferentes partes de diferentes sistemas, sistemas distribuidos, una organización, personas, etc. Puede entenderse como una colección de partes y relaciones que al momento de organizarse pueden lograr un propósito.
- *Model*: es la descripción o especificación de un sistema y su ambiente para el cumplimiento de un propósito. Dentro de un modelo podemos representar el un negocio, un dominio, un software o cualquier otro aspecto de un sistema.

Ilustración 36 Las relaciones entre los modelos, metamodelos y sistemas



Fuente: (OMG, The MDA Foundation Model, 2010)

- *Model-Driven:* es el medio que provee el contexto para el uso de modelos dentro del análisis, el diseño, la construcción, el despliegue, la operación, el mantenimiento y la modificación.
- *Architecture:* es la especificación de las partes y conectores de un sistema, teniendo en cuenta las reglas paras las interacciones de las partes con los conectores que la usan. También podemos definirla como un conjunto de reglas que definen la estructura de un sistema y las interrelaciones entre sus partes.
- *Viewpoint:* especifica un conjunto reutilizable de criterios para la construcción, selección y presentación de una parte de la información acerca de un sistema, enfocado a las preocupaciones de unos stakeholdres particulares. También se puede definir como la técnica de abstracción que tiene en cuenta concepto de arquitectura con el fin de satisfacer las particularidades de un sistema.
- *Abstraction:* trata de representar un modelo simple tratando de suprimir el detalle que no es relevante.

Dentro de la guía, de MDA la descripción de la misma es guiada desde el concepto de viewpoint. MDA contiene tres (3) viewpoints: Computation independent, Platform independent, Platform specific.

Recordemos que un viewpoint es la forma de abstracción que consigue presentar un conjunto de conceptos arquitectónicos y unas reglas con el fin de concentrarse en las particularidades de un sistema. Lo que MDA busca con estos tres (3) viewpoints es poder observar desde diferentes puntos de vista un sistema.

El Computation independent viewpoint tiene como objetivo presentar el sistema y su entorno, no incluye otros detalles de su estructura. En algunos casos, éste es llamado modelo de dominio. Este viewpoint juega un rol muy importante ya que busca reducir la brecha entre los expertos relacionados con el dominio y los expertos relacionados con el diseño y la construcción que satisface los requisitos planteados por el dominio.

El Platform independent viewpoint se enfoca en la operación del sistema mientras que oculta los detalles para una plataforma en particular. Este presenta que parte de la especificación completa no cambia de una plataforma a otra.

El Platform specific viewpoint combina el Platform independent viewpoint con un enfoque adicional relacionado con el uso de una plataforma específica para un sistema.

También, podemos encontrar los tipos de modelos dentro de MDA. Estos modelos ayudan a representar cada uno de los viewpoint de MDA: Computation independent model (CIM), Platform independent model (PIM) y Platform specific model (PSM).

Un CIM es una vista de un sistema desde el Computation independent viewpoint. Este no presenta detalle de la estructura de un sistema.

Un PIM es una vista de un sistema desde el Platform independent viewpoint. Un PIM exhibe un grado específico de independencia de la plataforma de manera que sea adecuado para su uso con un número de diferentes plataformas de tipo similar.

Un PSM es una vista de un sistema desde el Platform specific viewpoint. Un PSM combina lo especificado dentro de un PSM con los datos que especifican la forma en que un sistema utiliza un tipo particular de plataforma.

La información anterior fue obtenida de la guía de OMG MDA. (OMG, 2014) (MDA Guide Version 1.0.1, 2003)

4.2. PROPUESTA MDA

4.2.1. PRESENTACIÓN DE CONCEPTOS

Ya teniendo claro los conceptos anteriormente expuestos, el objetivo de este numeral es presentar que de lo mencionado en la guía se va aplicar al proyecto. A continuación se presenta:

- Para el proyecto se aplicara la versión 2.0 de la guía, pero teniendo en cuenta la guía en la versión 1.
- Se utilizara Zooming in/out. Esto permite realizar modelos a mayor/menor nivel de detalle según se requiera. Este punto se verá reflejado en gran parte en el PIM.
- Se van a tener en cuenta en el modelo CIM la ontología resultado del capítulo 2, y se complementara con los procesos que se encuentra representados en BPM y el modelo de dominio resultado del numeral de la recolección de la información.
- Para la transformación de CIM a PIM se realizara de manera manual.
- Se van a tener en cuenta los modelos PIM se utilizaran modelos UML propuestos dentro del modelo 4+1 de Kruchten el cual encaja con el estándar de la IEEE 1471-2000 (Recommended Practice for Architecture Description of Software-Intensive Systems) que se utiliza para describir la arquitectura de un sistema software intensivo basado en el uso de múltiples puntos de vista.
    - ✓ Para la vista lógica: diagrama de clases.

- ✓ Para la vista de despliegue: diagrama de componentes y diagrama de paquetes.
- ✓ Para la vista de procesos: diagrama de actividad y diagrama de estado.
- ✓ Para la vista física: diagrama de despliegue.
- ✓ Para la vista de escenarios: diagrama de casos de uso.

- La transformación de PIM a PSM se realizara gracias a la herramienta Model2Roo.
- La transformación de PSM a código se realizara gracias a la tecnología Spring Roo. El prototipo estará construido en esta tecnología.

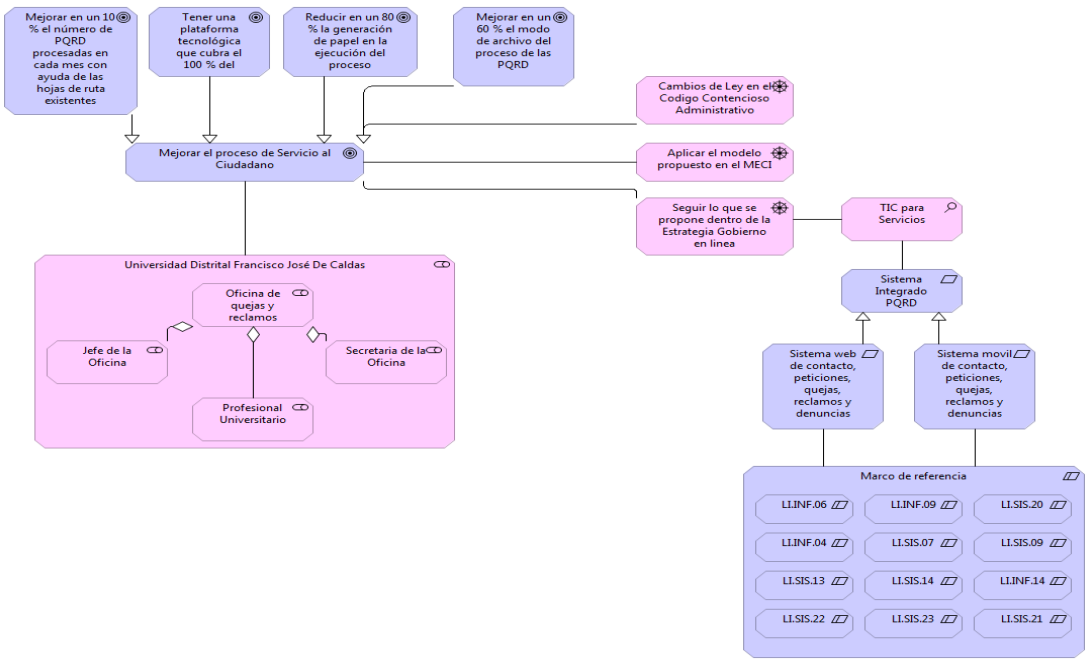
4.2.2. PRESENTACION DE MODELOS

4.2.2.1. MODELO INDEPENDIENTE DE LA COMPUTACIÓN (CIM)

Dentro de la guía menciona a CIM como el modelo que solamente representa el negocio, las personas, los lugares, las reglas, etc. Teniendo en cuenta lo anterior y aplicando conocimiento dado en la especialización podemos hacer semejanza con otros modelos que representan el negocio y que podemos utilizar para la representación del modelo. Uno de estos es Business Motivation Model (BMM) perteneciente a la especificación planteada por la OMG (Business Motivation Model™ (BMM™), 2015) y que podemos usar Archimate (OpenGroup, 2013) para su representación.

Con el BMM podemos identificar los factores que motivan el desarrollo de los planes de negocio de una organización. A continuación el modelo propuesto por el proyecto:

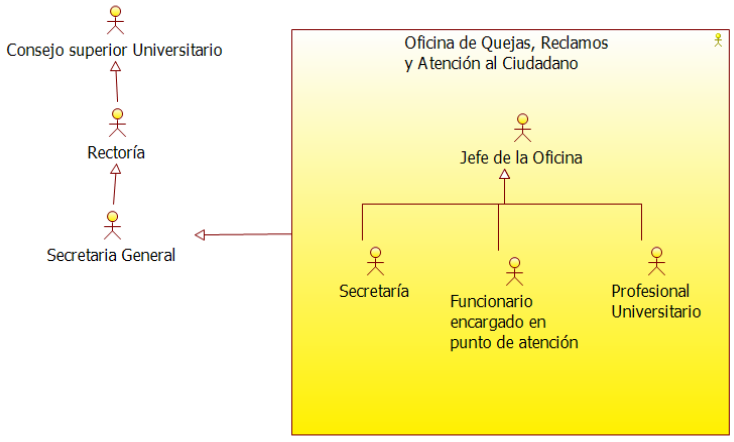
Ilustración 37 - BMM ejemplo



Fuente: Propia

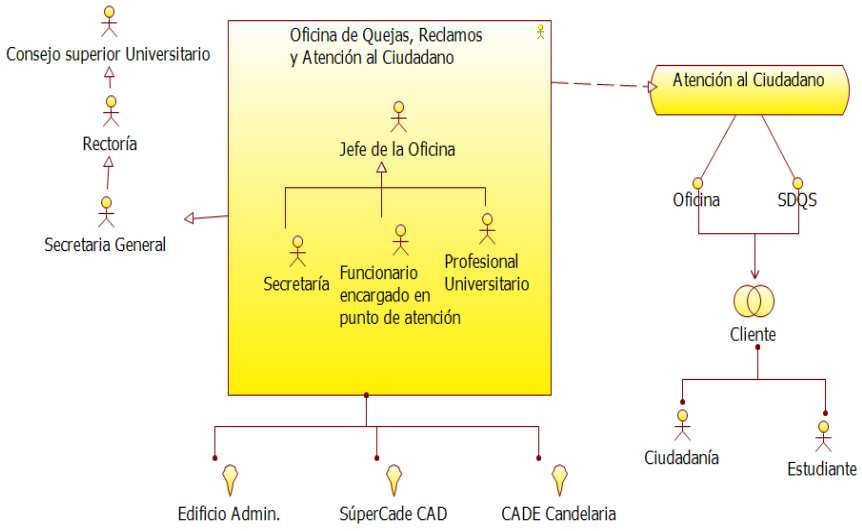
También podemos encontrar en Architmate los viewpoints relacionados con la arquitectura (Group, 2013). Dentro de estos podemos encontrar los viewpoints de la organización, Actor Cooperation, entre otros. A continuación el modelo propuesto por el proyecto:

Ilustración 38 - Organization Model



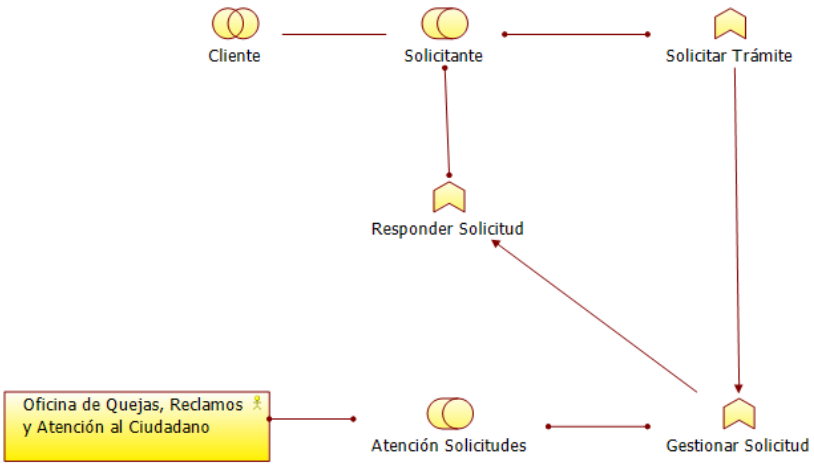
Fuente: Propia

Ilustración 39 - Actor Cooperation Model



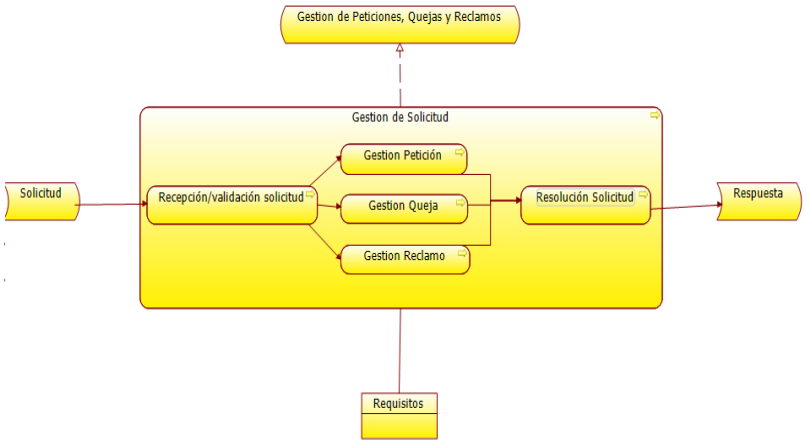
Fuente: Propia

Ilustración 40 - Business Function Model



Fuente: Propia

Ilustración 41 - Business Process Model

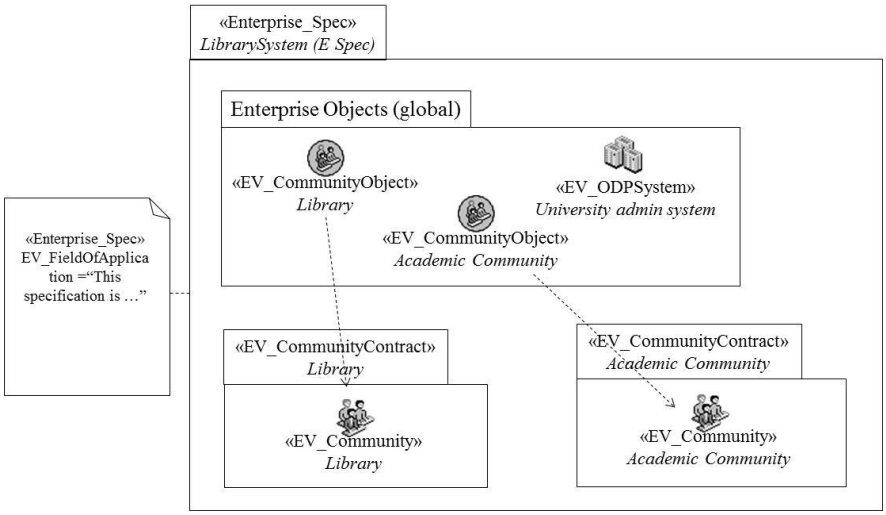


Fuente: Propia

Cabe aclarar que estos modelos están basados en la Oficina de Quejas, Reclamos y Atención al Ciudadano de la Universidad.

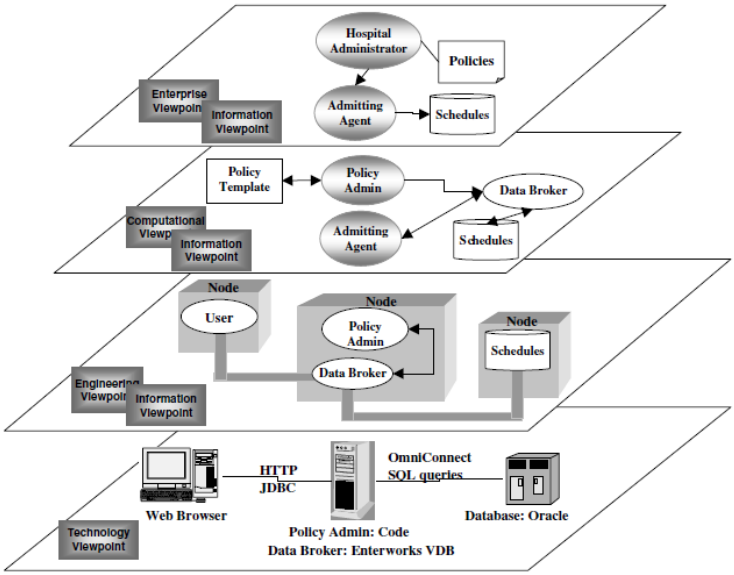
Otro modelo de referencia es el RM-ODP; el cual es un estándar internacional de arquitectura basada en objetos para su uso en la arquitectura de sistema distribuidos. Hace parte de una familia de estándares pertenecientes a la ISO/IEC and ITU-T (ITU-T Rec X.901-904 | ISO/IEC 10746). A continuación un ejemplo de un sistema de una biblioteca facilitado en la página (RM ODP):

Ilustración 42 - RM ODP Enterprise Viewpoint de un sistema para una biblioteca



Fuente: (RM ODP)

Ilustración 43 - RM ODP Viewpoints para un Hospital



Fuente: (RM ODP)

4.2.2.2. MODELO INDEPENDIENTE DE LA PLATAFORMA (PIM)

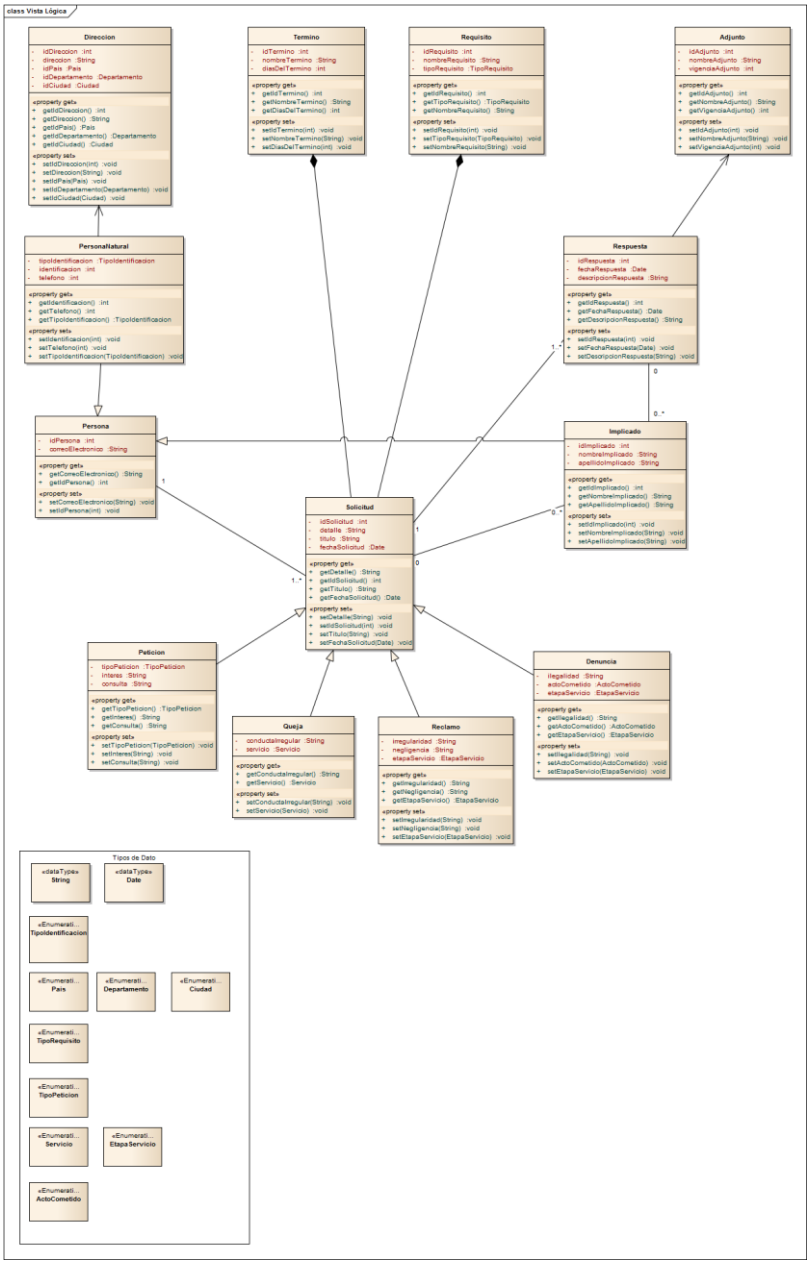
Como se mencionó anteriormente, la transformación de CIM a PIM se realizara manualmente. Dentro de la guía en la versión 1 (MDA Guide Version 1.0.1, 2003) en el numeral 4.1 el cual trata los grados y métodos de transformación de un modelo. Este cual menciona los cuatro (4) diferentes transformaciones entre ella la manual.

Al aplicar transformaciones entre modelo en algunos casos se tienen que tomar decisiones tanto de diseño como desarrollo que impiden hacer la transformación en su totalidad. Dentro de la investigación realizada para aplicarla en el proyecto se encontraron varias fuentes en donde se menciona en cómo hacer una de CIM a PIM pero deberían tener una serie de requisitos antes que al momento de cumplirlos se contradice con la guía; ejemplo (Y. Rhazali, 2014) desde el CIM plantean la posible existencia de diagramas de colaboración y de actividades. Mientras otras fuentes más especializadas (Dragan Djurić, 2005) propone el OMD (ONTOLOGY MODELING ARCHITECTURE) donde propone realizar el CIM de manera manual y en base a la ontología.

- **PARA LA VISTA LÓGICA**

Dentro de esta vista principalmente se presentan los requerimientos funcionales del sistema y de lo que el sistema debe hacer, las funciones y servicios que se han definido.

Ilustración 44 - Diagrama de Clases

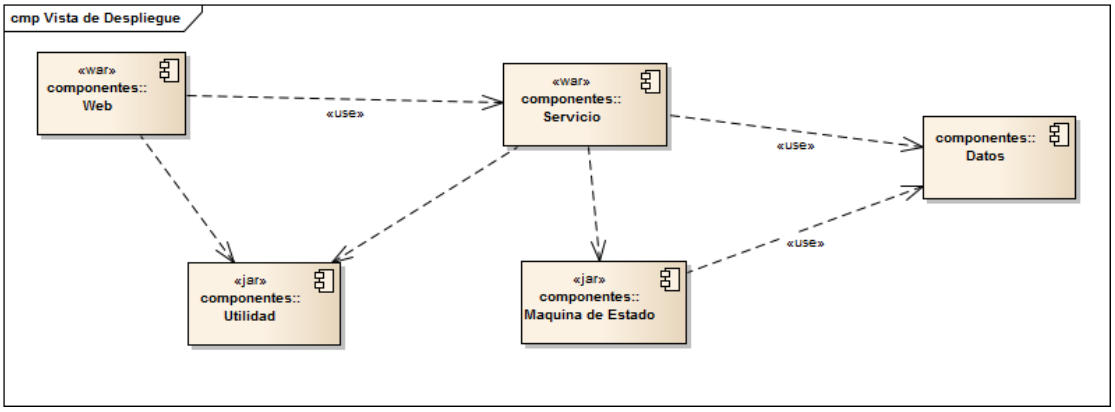


Fuente: Propia

- **PARA LA VISTA DE DESPLIEGUE**

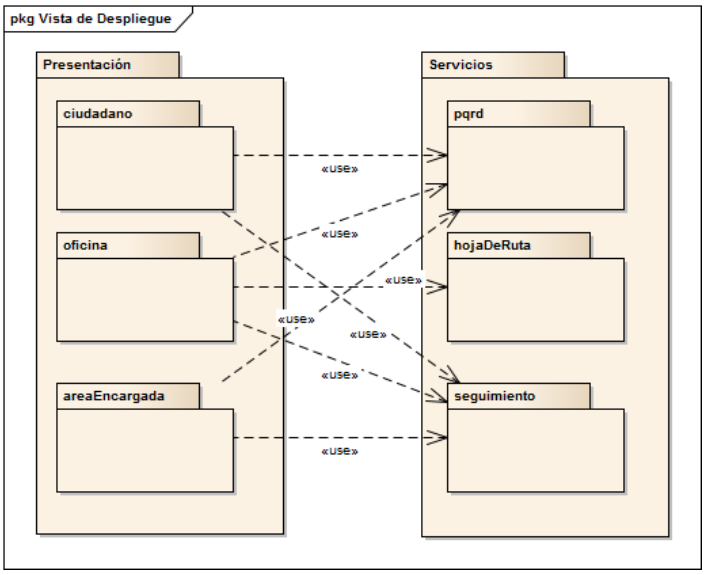
Esta vista se presenta como estará dividido el sistema, representado en sus componentes.

Ilustración 45 - Diagrama de componentes



Fuente: Propia

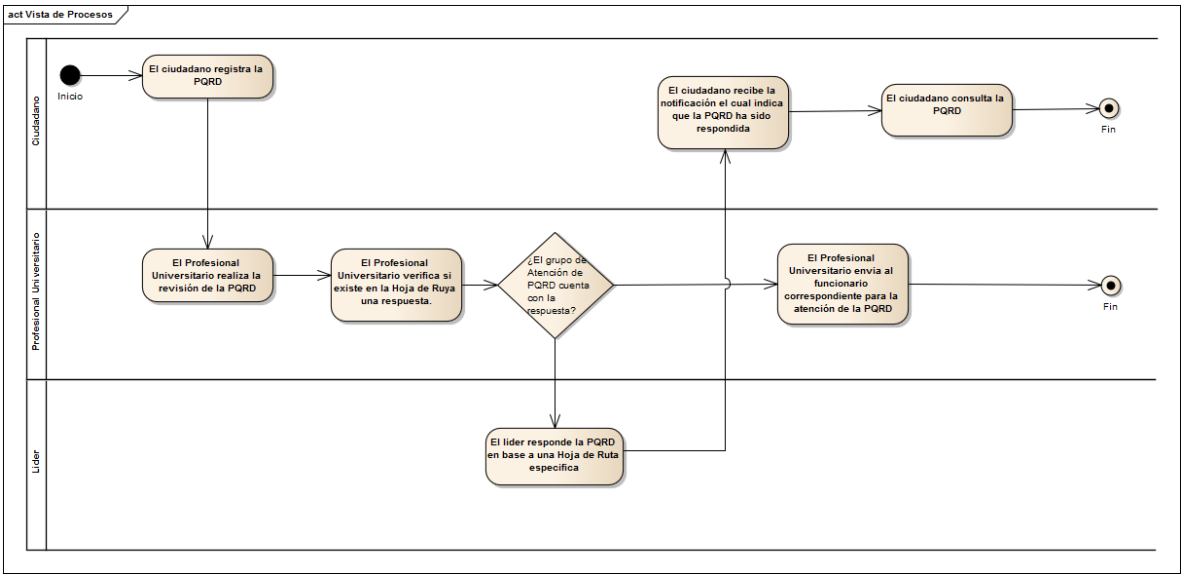
Ilustración 46 - Diagrama de paquetes



Fuente: Propia

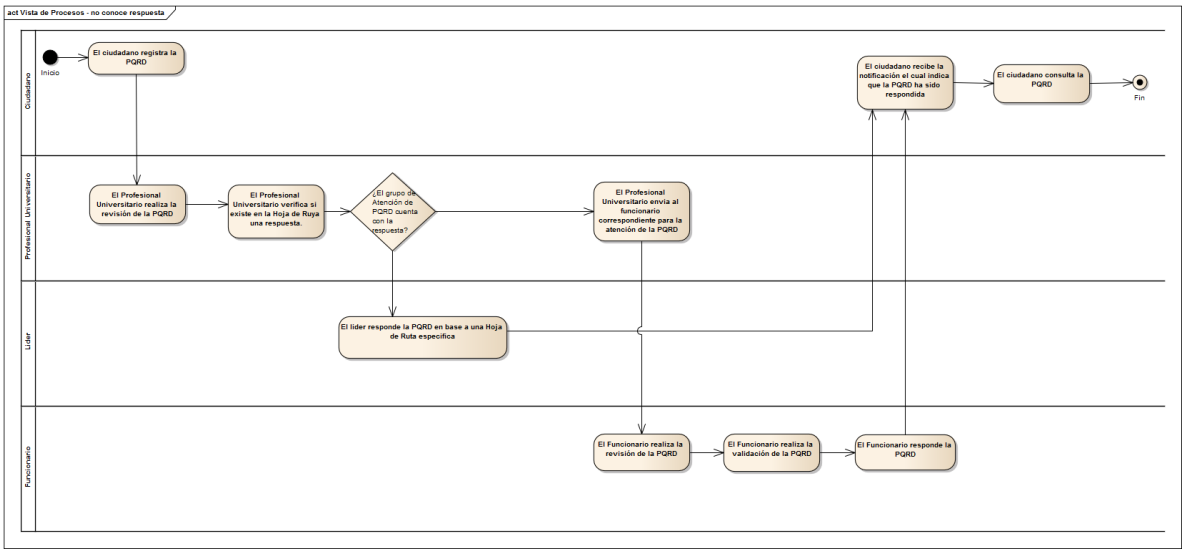
- **PARA LA VISTA DE PROCESOS**

Ilustración 47 Diagrama de actividades del proceso PQRD cuando el grupo de atención conoce la respuesta



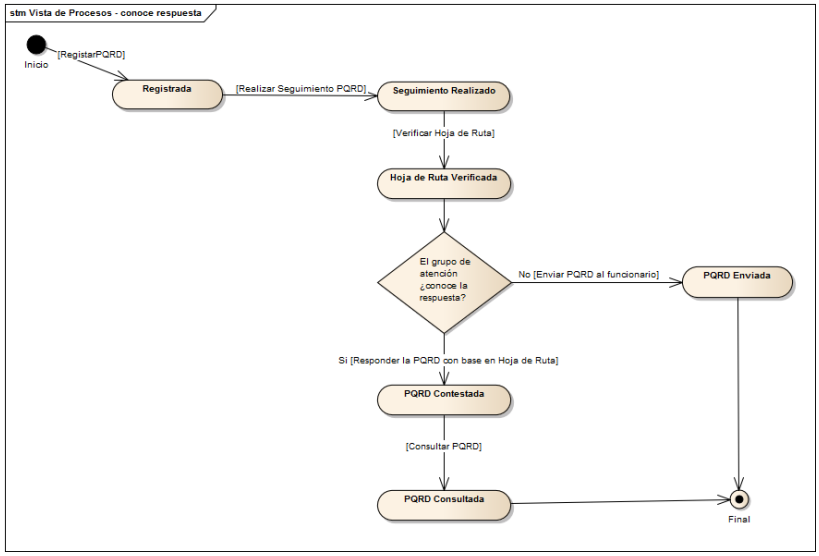
Fuente: Propia

Ilustración 48 Diagrama de actividades del proceso PQRD cuando el grupo de atención no conoce la respuesta



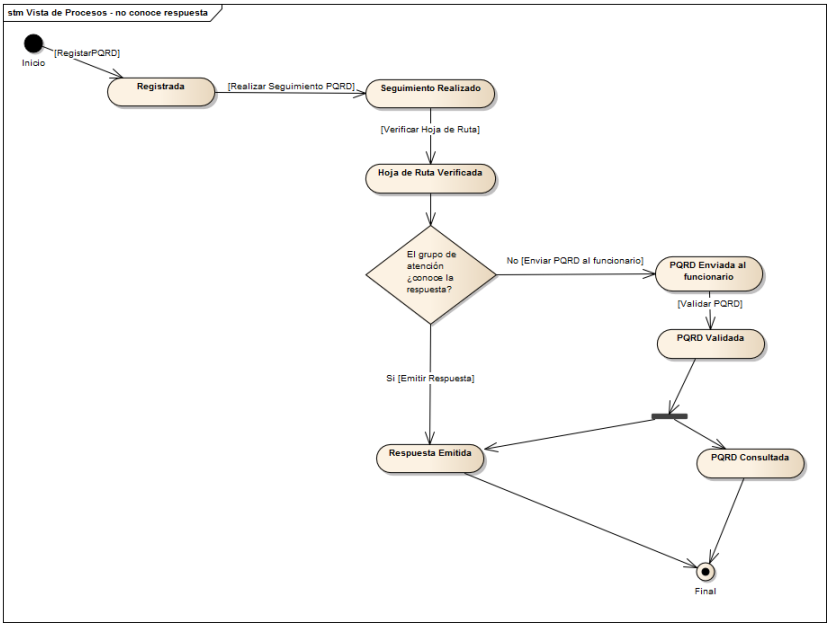
Fuente: Propia

Ilustración 49 - Diagrama de estado del proceso PQRD cuando el grupo de atención conoce la respuesta



Fuente: Propia

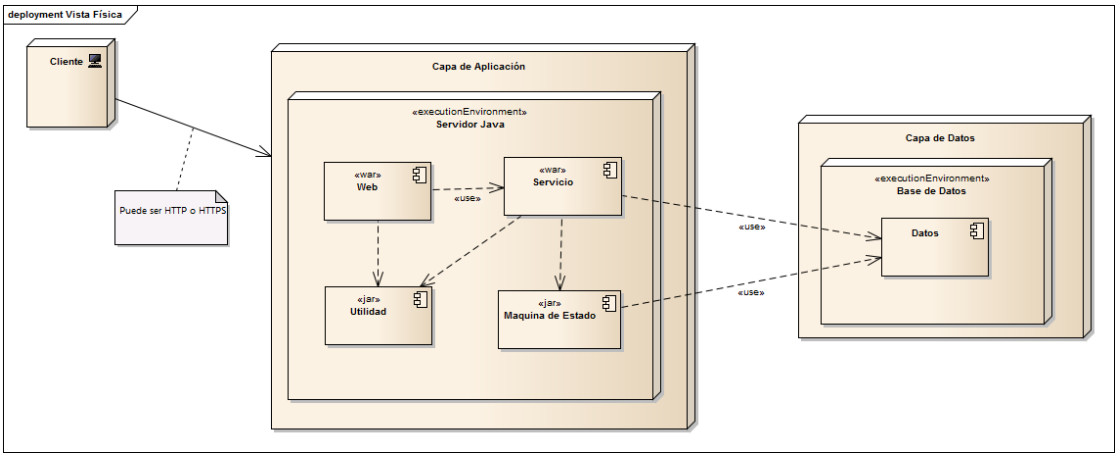
Ilustración 50 - Diagrama de estado del proceso PQRD cuando el grupo de atención no conoce la respuesta



Fuente: Propia

- **PARA LA VISTA FISICA**

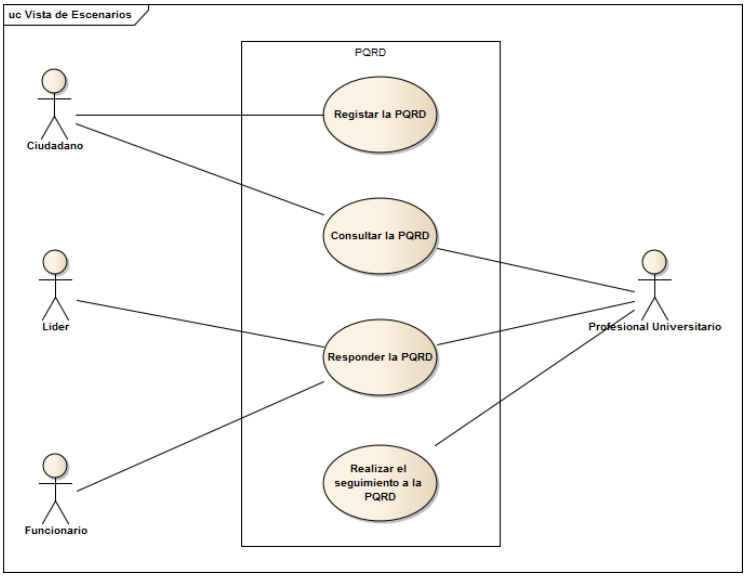
Ilustración 51 - Diagrama de despliegue



Fuente: Propia

- **PARA LA VISTA DE ESCENARIOS**

Ilustración 52 - Diagrama de casos de uso



Fuente: Propia

CAPÍTULO 5. DESARROLLO DEL PROTOTIPO DE ALTO NIVEL

Como se comentó en el capítulo anterior en la sección 4.2.1 Presentación de conceptos, la transformación de PIM a PSM se realizara gracias a la herramienta Model2Roo, con el fin de utilizar a Spring Roo como generador de código.

Dentro de la investigación se encontraron varias herramientas que ayudan a realizar la tarea de a partir de un diseño en UML, entre ellas podemos mencionar las siguientes:

- RooModeler: https://roomodeler.com/
- model2roo: https://code.google.com/archive/p/model2roo/
- GenMyModel: https://www.genmymodel.com/ teniendo en cuenta el articulo en el blog http://blog.sudobits.com/2013/07/14/genmymodel-online-uml-tool-for-software-architects-and-developers/ donde menciona a Spring Roo como generador de código.
- RooPlugout: https://gna.org/projects/roo-plugout/

Dentro de la investigación encontramos que la mayor herramienta para la tarea seria model2roo. Esta decisión se basa en que las demás herramientas se les encontró un detalle que no servía para el desarrollo del proyecto: RooModeler todavía se encuentra en pruebas, al igual que RooPlugout, sin mencionar que la primera y GenMyModel tienen un alto costo ya que la versión free solamente permite la creación de pocos objetos hasta las relaciones las tienen en cuenta en total solamente permitiría unas 10 clases y sus relaciones. Otro punto a anotar relacionado con herramientas ya existentes y están presentadas dentro de la las posibilidades que la OMG tiene en la página; esto debido a que ya se encuentran que están comprobadas y evaluadas no solamente por la OMG sino también por la comunidad, y el fin del proyecto era presentar herramientas de fácil acceso y de fácil uso, principalmente para presentar nuevas alternativas y dejar al lector la intriga de construir/diseñar nuevas herramientas. A continuación se presenta una tabla que es

resultado de un estudio reflejado en un artículo acerca de las herramientas mencionadas en la pagina de la OMG http://www.omg.org/mda/committed-products.htm

Ilustración 53 - Comparación herramientas presentadas por la OMG

Herramientas	*ArcStyler [15]*	*OptimalJ [16]*	*AndroMDA [17, 18]*	*Codagen Architect*	*Together Architect*
Entrada	UML	XMI v1.1	XMI v1.1	XMI v1.1, MDD de Racional, Visio, together.	XMI
Lenguaje de salida	Todos	En J2EE	Todos	Java,C#,C++, Visual Basic.	Java,C#,C++,Corba, Visual Basic 6,Visual Basic .Net
Estándares	XMI, UML, MOF, JMI	XMI, UML, MOF, XML, WSDL, J2EE	Struct, Netbeans MDR, antMybernate, Velocity, xDoclet y Maven.	Struct	XMI
Soporte CIM					*
Soporte PIM	√	√	√	*	*
Soporte PSM	*	√	√	*	√
CIM a PIM					
PIM a PSM		√	√		
PIM a Código	√			√	
PSM a Código		√	√		
Consistencia	√	√	*	√	*
Trazabilidad	√	√	*	*	*

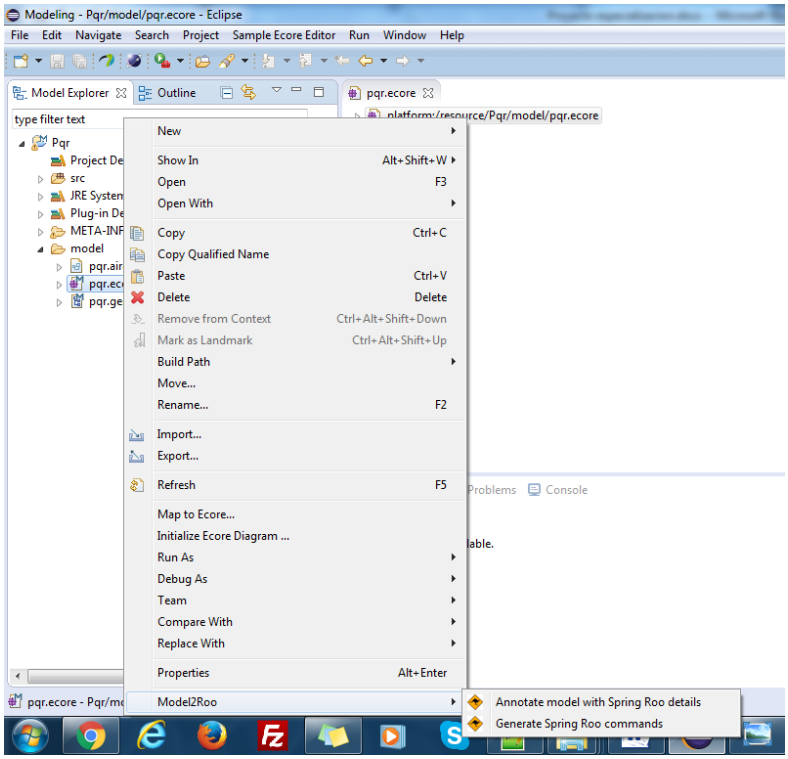
Fuente: (Edna D. LÓPEZ L., 2006)

Model2roo es una herramienta desarrollada con el fin de aprovechar las técnicas del desarrollo de software enfocado a la web y el desarrollo basado en modelos. Cabe tener en cuenta que esta herramienta en su versión actual no soporta los comandos de la última versión de Spring Roo (no soporta la versión 2.0). Como tal la herramienta se encuentra como plugin de Eclipse; presta soporte desde un diagrama construido con las EMF tools hasta un diagrama construido en el editor del proyecto Papyrus que se ha convertido en uno de los más usados por la comunidad para el diseño en UML https://eclipse.org/papyrus/. Hoy día model2roo se encuentra como proyecto dentro del code google.

Basta con acceder al enlace https://code.google.com/archive/p/model2roo/ y se puede acceder al proyecto a la zona de wiki donde se puede encontrar la información; y a la zona de descargas donde se evidencian los múltiples ejemplos y el plugin para Eclipse. Dentro de la documentación se encuentra la guía de como instalarlo. Una vez instalado es solamente hacerlo funcionar el proyecto. Dentro de proyecto se analizaron las formas de cómo generar la aplicación partiendo de un diagrama de clases en UML gracias a Papyrus o con las herramientas ECore tools. Se escogió la segunda debido a que tenemos la menor intervención humana, ya que en el plugin con Papyrus se tiene que adicionar manualmente los comandos y características relacionadas con Spring Roo. En cambio con las herramientas ECore no sucede esto, el mismo plugin realiza la adición de lo necesario. A continuación se presentaran los pasos desarrollados para llevar a cabo la transformación.

Este plugin contiene dos (2) opciones que se muestran a continuación en la imagen cuando se quiere realizar la transformación desde un modelo ECore:

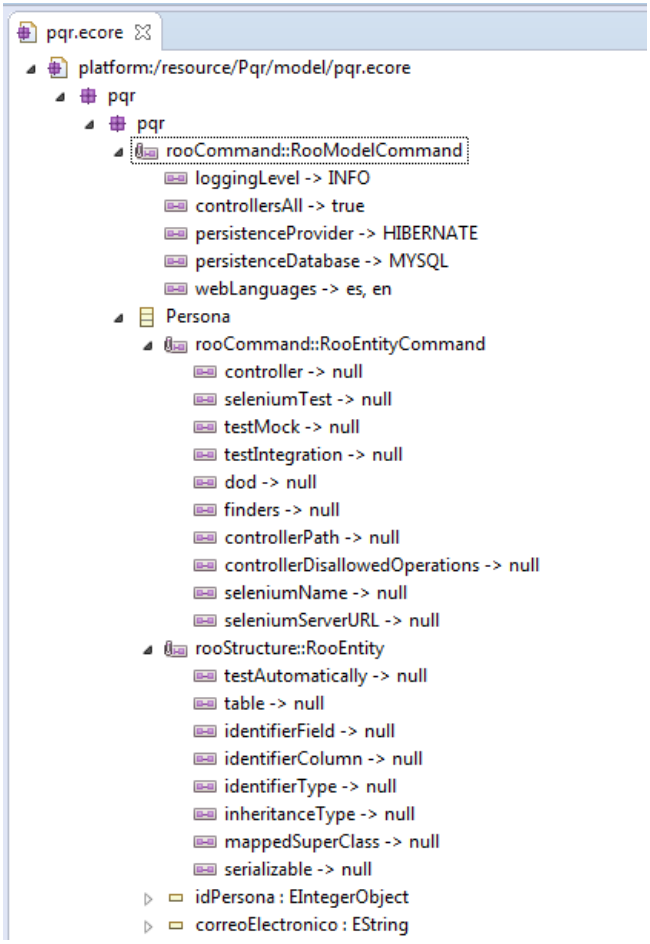
Ilustración 54 - Presentación de las opciones del plugin



Fuente: Propia

Primero se utiliza la opción *"Annote Model with Spring Roo details"*, gracias a esta opción se le dan características al modelo que tienen en cuenta comandos y características de Spring Roo. Estos existen a nivel de modelo en general y de cada uno de los elementos. Estos comandos y características están documentados dentro de la especificación de model2roo. Para el caso del proyecto se configuraron las opciones a nivel de modelo.

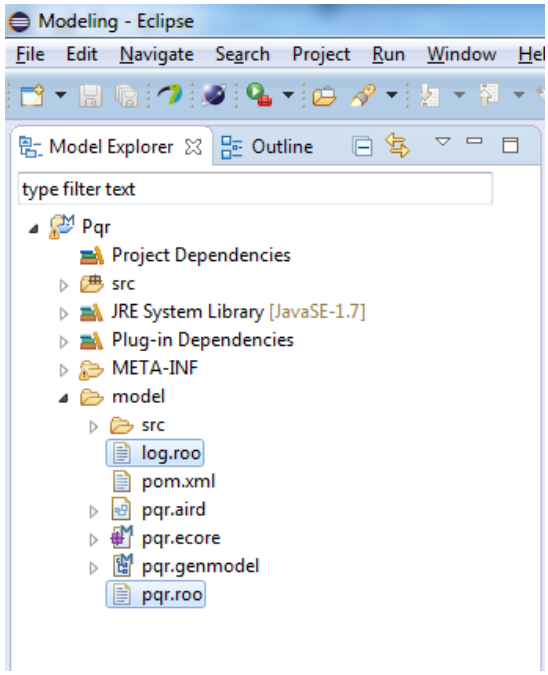
Ilustración 55 - Vista ECore con detalles de Spring Roo



Fuente: Propia

Y la segunda opción que se utiliza es *"Generate Spring Roo commands"*, gracias a este comando ya comienza a generar el script de Spring Roo.

Ilustración 56 - Generación Script Spring Roo



Fuente: Propia

Ilustración 57 - Script Spring Roo

pqr.roo

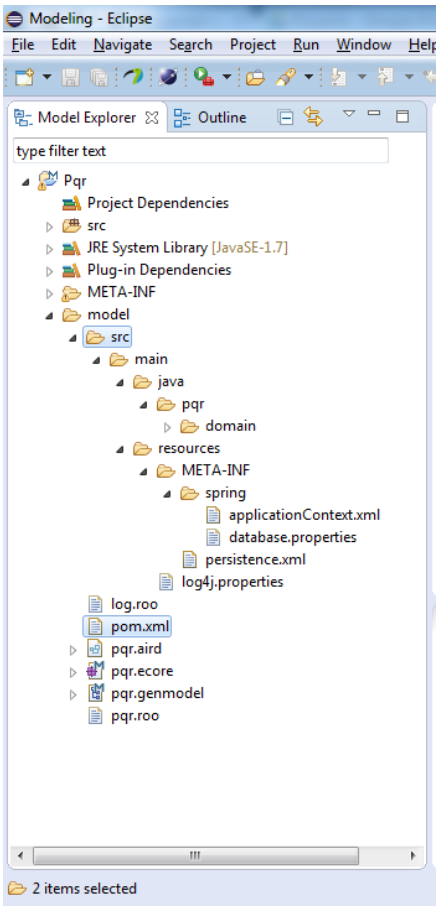
```
field string --fieldName tipoIdentificacion --class ~.domain.Natural
field string --fieldName identificacion --class ~.domain.Natural
field string --fieldName telefono --class ~.domain.Natural
field string --fieldName idDireccion --class ~.domain.Direccion
field string --fieldName direccion --class ~.domain.Direccion
field string --fieldName idPais --class ~.domain.Direccion
field string --fieldName idDepartamento --class ~.domain.Direccion
field string --fieldName idCiudad --class ~.domain.Direccion
field string --fieldName tipo --class ~.domain.Peticion
field string --fieldName interes --class ~.domain.Peticion
field string --fieldName consulta --class ~.domain.Peticion
field string --fieldName conductaIrregular --class ~.domain.Queja
field string --fieldName servicio --class ~.domain.Queja
field string --fieldName irregularidad --class ~.domain.Reclamo
field string --fieldName negligencia --class ~.domain.Reclamo
field string --fieldName etapaServicio --class ~.domain.Reclamo
field string --fieldName ilegalidad --class ~.domain.Denuncia
field string --fieldName actoCometido --class ~.domain.Denuncia
field string --fieldName etapaServicio --class ~.domain.Denuncia

// References
field reference --type ~.domain.Solicitud --fieldName solicitud  --class ~.domain.Persona
field reference --type ~.domain.Respuesta --fieldName respuesta  --class ~.domain.Solicitud
field reference --type ~.domain.Implicado --fieldName implicado  --class ~.domain.Solicitud
field set --type ~.domain.HojaDeRuta --fieldName hojaderuta  --class ~.domain.Solicitud
field reference --type ~.domain.Adjunto --fieldName adjunto  --class ~.domain.Respuesta
field reference --type ~.domain.Implicado --fieldName implicado  --class ~.domain.Respuesta
field set --type ~.domain.Solicitud --fieldName solicitud  --class ~.domain.Termino
field set --type ~.domain.Solicitud --fieldName solicitud  --class ~.domain.Requisito
field reference --type ~.domain.Direccion --fieldName direccion  --class ~.domain.Natural

// System commands
web mvc setup
web mvc language --code es
web mvc language --code en
logging setup --level INFO
web mvc all --package ~.web
```

Fuente: Propia

Ya una vez realizada la transformación de PIM a PSM gracias al proyecto model2roo, el paso siguiente es realizar el paso de PSM a código y el mismo proyecto nos ayudara en esto. Lo anterior también es otra de las razones del porque se seleccionó la herramienta.

En el paso dos (2) llamado *"Generate Spring Roo commands"* y mencionado anteriormente además de la generación del script Spring Roo y si todo se encuentra correcto genera la aplicación teniendo en cuenta la tecnología Maven. En la siguiente imagen se pude observar el pom y la carpeta de los fuentes del proyecto.

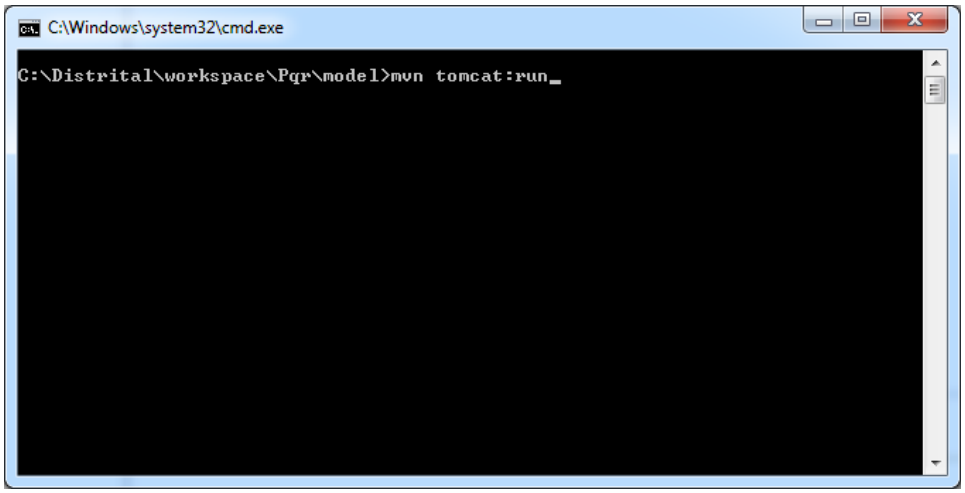
Ilustración 58 - Aplicación generada



Fuente: Propia

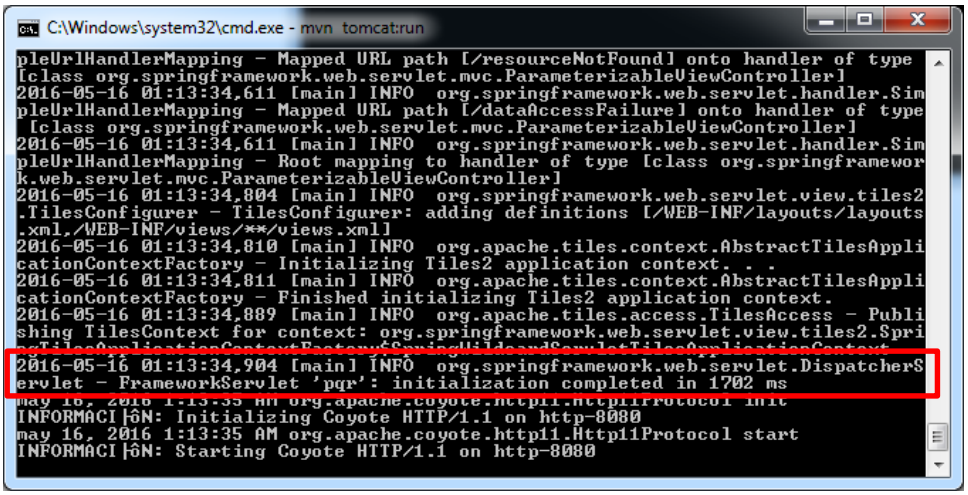
Antes de demostrar que la aplicación quedo construida se definen los valores para la conexión a la base de datos, esta configuración se realiza en el archivo database.properties. Una vez configurada la base de datos se ejecuta por medio de líneas de comando el siguiente comando con el fin de correr la aplicación. Como servidor de utilizar tomcat y para el despliegue nos apoyaremos en Maven.

Ilustración 59 - Comando ejecución aplicación



Fuente: Propia

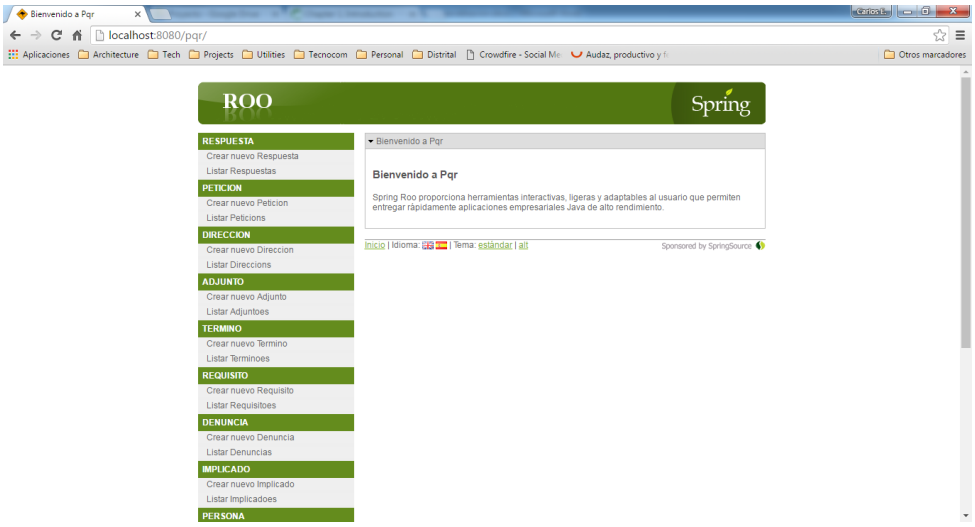
Ilustración 60 - Servidor en ejecución con la aplicación



Fuente: Propia

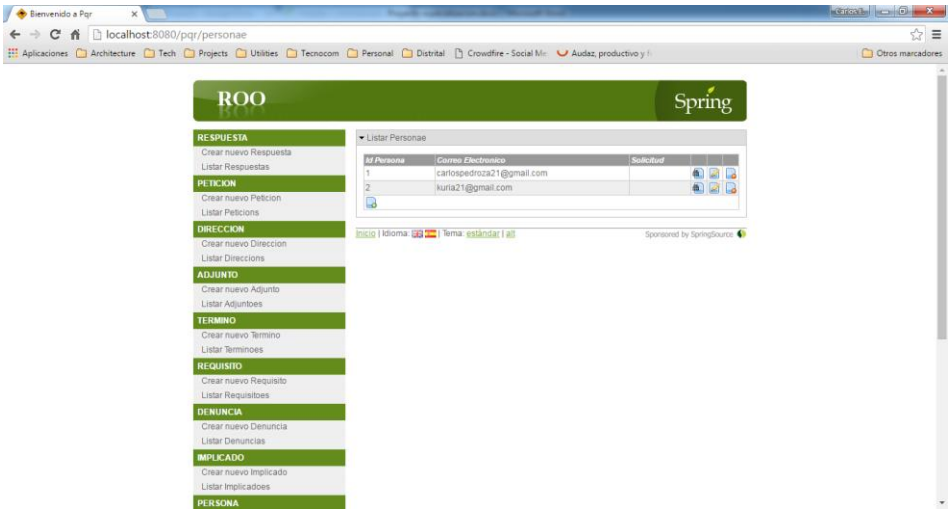
En la anterior imagen se evidencia que la aplicación se encuentra arriba en el contexto pqr. A continuación se presenta la aplicación desde el navegador.

Ilustración 61 - Aplicación desde el navegador



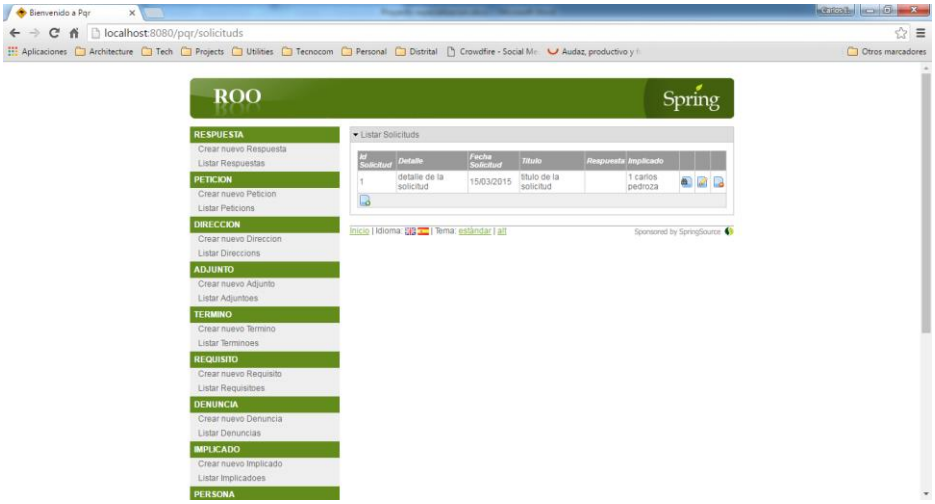
Fuente: Propia

Ilustración 62 - Consulta de Personas dentro de la aplicación



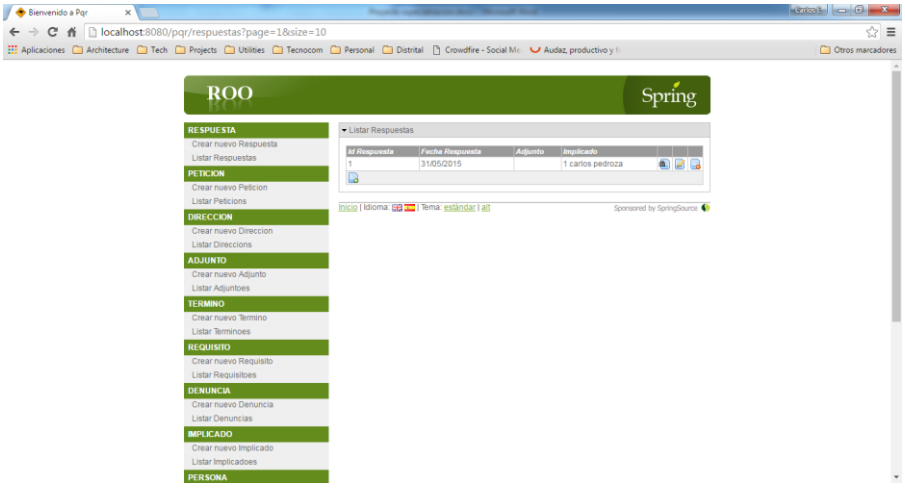
Fuente: Propia

Ilustración 63 - Consulta de Solicitudes dentro de la aplicación



Fuente: Propia

Ilustración 64 - Consulta de Respuesta dentro de la aplicación



Fuente: Propia

PARTE 3. CIERRE DE LA INVESTIGACIÓN

CAPÍTULO 7. CONCLUSIONES

7.1. VERIFICACIÓN, CONTRASTE Y EVALUACIÓN DE LOS OBJETIVOS

- La construcción de la ontología de dominio permite la categorización del conocimiento sobre el dominio del proceso de Peticiones, Quejas, Reclamos y Denuncias definido por la normatividad vigente para las entidades del estado colombiano, por medio de la definición de las clases, relaciones y restricciones; facilitando el proceso de modelado, permitiendo la creación de modelos más expresivos, verificables y estandarizados, lo que permite no sólo mejorar la calidad de los productos de software generados sino también la comunicación entre desarrolladores gracias al manejo de un lenguaje unificado, en fases posteriores en la implementación de la arquitectura propuesta.
- La utilización de MDA contribuye a la estandarización y uniformidad en la fase de diseño de la arquitectura gracias a la utilización de los conceptos y herramientas provistas por la propuesta de la OMG para la elaboración de los modelos definidos en el alcance del proyecto, reflejando de manera general la propuesta arquitectónica para el dominio de las PQRD que permita el reúso y adecuación a las necesidades propias de cada entidad del estado colombiano que requieran implementarla.
- La adición de los lineamientos de presentados dentro del Marco de Referencia sirve como instrumento dentro del proyecto para implementar la Arquitectura TI y habilitar la Estrategia de Gobierno en Línea. Con esto se busca habilitar las diferentes estrategias TIC pero en este proyecto se busca la estrategia TIC para servicios que contempla todo lo relacionado con el sistema integrado de PQRD.

7.2. SÍNTESIS DEL MODELO PROPUESTO

El papel de los modelos es fundamental en el desarrollo de software para proporcionar el reusó de los diferentes elementos de software generados a partir de la definición y abstracción de estos modelos definidos, lo cual proporciona características únicas para el desarrollo de software como son la abstracción del dominio del problema y la automatización en la generación de los componentes de software, gracias a esto es posible la implementación del sistemas de software para el proceso de Peticiones, Quejas y Reclamos en cualquier entidad del estado colombiano que permita la radicación por parte de la ciudadanía de sus peticiones, quejas, reclamos y denuncias ante la entidad que implemente la solución de software como del seguimiento y tramite al interior de la organización.

La investigación del proceso de Peticiones, Quejas, Reclamos y Denuncias tomando como base la legislación a la cual se deben regir permitió realizar la abstracción del modelo genérico para el proceso en las entidades del estado colombiano, de tal forma que a futuro dichos modelos podrán ser extendidos de acuerdo a necesidades futuras que puedan presentarse. Además, el uso de los modelos generados por medio de las trasformaciones definidas en el desarrollo del trabajo y por medio generadores de código automáticos como Spring Roo permiten la generación de elementos de datos y plantillas CRUD que agilizan el proceso de codificación y reducen el costo de mantenimiento.

Por último, cabe mencionar que el proyecto provee las herramientas para la implantación de la aplicación para manejo de PQRs a las nuevas entidades del estado Colombiano que se encuentran en las fases iniciales o busquen cumplir con los lineamientos de la estrategia de gobierno electrónico, por medio de la propuesta de una arquitectura dirigida por modelos (MDA) basándose en una ontología de dominio. Aprovechando los avances de la tecnología para garantizar una mejor comunicación e interacción con la ciudadanía, que permita además la prestación de más y mejores servicios por parte del Estado.

7.3. APORTES ORIGINALES

En la definición de la arquitectura para el proceso de Peticiones, Quejas, Reclamos y Denuncias en las entidades del estado colombiano se logró abordar una problemática común a muchas de estas instituciones de forma que el resultado fuera relevante y aplicasen sin mayor impacto a cada una de ellas, para lograrle fue necesario hacer uso de herramientas de la web semántica como de la propuesta de la OMG para abordar el Desarrollo Dirigido por Modelos, es así que entre los aportes relevantes del proyecto están:

- La construcción de una Ontología de dominio para el proceso de Peticiones, Quejas y Reclamos (Atención al Ciudadano), como metamodelo de más alto nivel de abstracción como apoyo transversal de definición de la arquitectura.
- La separación por medio de la Arquitectura Dirigida por Modelos (MDA) de la especificación de la funcionalidad del sistema de su implementación propiamente dicha sobre una plataforma tecnológica especifica.
- La utilización de una herramienta de desarrollo rápido de aplicaciones (RAD) como SpringRoo, que permite el desarrollo de aplicaciones JavaEE de forma muy rápida y cómoda, para la validación de la arquitectura propuesta.

CAPÍTULO 8. PROSPECTIVA DEL TRABAJO DE GRADO

8.1. LÍNEAS DE INVESTIGACIÓN FUTURA

A partir de la definición de la arquitectura quedan planteadas diferentes líneas de investigación complementarias entre las que resaltamos las siguientes:

- Refinamiento de los modelos generados para cada fase propuesta en MDA como de las transformaciones para llegar a cada uno de estos modelos.

- Definición de sistemas de gestión documental que permita administrar el flujo de documentos entrantes como generados en el proceso de recepción como contestación de las solicitudes del proceso.
- Integración e interoperabilidad entre plataformas de gestión de procesos PQRD para las entidades del estado colombiano, por medio de la definición de estándares y lineamientos únicos, que provean el fortalecimiento y la evolución de estas plataformas a un sistema único interoperable de gestión.
- Creación de un método de transformación de CIM to PIM a partir de este modelo.
- Creación de un método de transformación de PIM to PSM a partir de este modelo.
- Creación de un método de transformación de PIM to PSM a partir de este modelo teniendo en cuenta otras tecnologías como .NET, PHP, etc.
- La definición de metamodelos como artefactos que contribuya en el proceso de desarrollo de software para otros procesos aparte del de PQRs para las entidades del estado colombiano.

8.2. TRABAJOS FUTUROS

A partir del resultado de este trabajo se propone como trabajos futuros los siguientes:

- Implementación de una aplicación de software bajo la Arquitectura Dirigida por Modelos propuesta en el presente trabajo en una entidad pública del estado colombiano.
- Generación de un metamodelo con ayuda de Meta-Object Facility o MOF a partir de lo planteado en este trabajo.
- Aplicación de UML Profiles planteados por la OMG a partir de lo planteado en este trabajo.

BIBLIOGRAFIA

AndroMDA. (18 de Septiembre de 2014). *AndroMDA*. Obtenido de andromda.org: http://www.andromda.org/

Arquitectura TI. (2015). *Arquitectura TI Colombia.* Obtenido de mintic: http://www.mintic.gov.co/arquitecturati/630/w3-channel.html

Bacha, F., Oliveira, K., & Abed, M. (2011). Providing Personalized Information in Transport Systems: A Model Driven Architecture Approach. *IEEE*.

Bernaras, A., Laresgiti, I., & Corera, J. (1996). Building and reusing ontologies for electrical network applications. *Proc. European Conference on Artificial Intelligence (ECAI_96)*, (págs. 298-302). Budapest, Hungary.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*.

Borst, W. (1997). *Construction of Engineering Ontologies.* PhD Thesis, University of Tweenty.

Business Motivation Model™ (BMM™). (2015). Obtenido de http://www.omg.org/spec/BMM/1.3/PDF/

Castells, P. (s.f.). *La web semántica.* Obtenido de http://arantxa.ii.uam.es/~castells/

Chalmeta Rosaleñ, R. (2007). *Catàleg de la Biblioteca*. Recuperado el 01 de Mayo de 2016, de Catàleg de la Biblioteca: http://cataleg.uji.es/record=b1343729*cat

Corcho, O., Fernandez lopez, M., & Gomez perez, A. (2003). Methodologies, toolsand languages for building ontologies. Where is their meeting point? *Data and Knowledge Engineering*, 41-64.

Dirección Gobierno en Línea. (30 de Junio de 2015). *Dirección de Gobierno en Línea.* Obtenido de mintic: http://www.mintic.gov.co/portal/604/w3-propertyvalue-7612.html

Dmitry, T. (2004). *FaCT++*. Obtenido de FaCT++: http://owl.man.ac.uk/factplusplus/

Dragan Djurić, D. G. (2005). *Journal of Object Technology.* Obtenido de http://www.jot.fm

Edna D. LÓPEZ L., M. G. (2006). Proceso de Desarrollo de Software Mediante Herramientas MDA. *SISTEMAS, CIBERNÉTICA E INFORMÁTICA*.

Falconer, S. (2010). *OntoGraf*. Obtenido de OntoGraf: http://protegewiki.stanford.edu/wiki/OntoGraf

Fensel, D. (2001). Ontologies: Silver Bullet for Knowledge. *Springer Verlag*.

Fernandez lopez, M., & Gómez Pérez, A. (1997). METHONTOLOGY:From ontological art towards ontological engineering. *AAAI Symposium on Ontoligical Engineering.* Stanford.

GEL. (09 de Octubre de 2015). *ESTRATEGIA GOBIERNO EN LINEA*. Obtenido de gobiernoenlinea: http://estrategia.gobiernoenlinea.gov.co/623/w3-channel.html

Group, O. (2013). *Architecture Viewpoints*. Obtenido de http://pubs.opengroup.org/architecture/archimate2-doc/chap08.html

Gruber, T. (1993). Atranslation approach to portable ontology specification. *Knowledge Acquisition* , 199-220.

Grununger, M., & Fox, M. (1995). Mhetodology for the design and evaluation of ontologies. *Workshop on Basic Ontological Issues in Knowledge.* Montreal.

Guha, R., & Lenat, D. (1990). Cyc: A mid-Term report. *AI Magazine*, Volume 11, Number 3.

Horridge, M. (2010). *OWLViz*. Obtenido de OWLViz: http://protegewiki.stanford.edu/wiki/OWLViz

IEEE. (6 de Septiembre de 2000). RM-ODP ARCHITECT'S PRIMER.

Kotis, K., & Vouros, A. (2006). Human-centered ontology engineering: The hcome methodology. *Knowledge and Information System 10* , 109-131.

Ley 1437. (18 de Enero de 2011). Congreso de la República. Colombia.

Ley 1474. (12 de Julio de 2011). Congreso de la República. Colombia.

López, E. D., & González, M. L. (2006). Proceso de Desarrollo de Software Mediante Herramientas MDA . *SISTEMAS, CIBERNÉTICA E INFORMÁTICA - VOLUMEN 3 - NÚMERO 2* .

Lopez, G., Servetto, A. C., Echeverría, A., Jeder, I., Grossi, M. D., & Rey, J. (2011). Ontologías en Arquitecturas Dirigidas por Modelos. *XIII Workshop de Investigadores en Ciencias de la Computación*.

Manual GEL. (10 de Septiembre de 2015). *Manual Estrategia de Gobierno en Línea.* Obtenido de gobiernoenlinea: http://estrategia.gobiernoenlinea.gov.co/623/articles-7941_manualGEL.pdf

Marco Jurídico GEL. (2015). *Marco Jurídico Institucional de la Estrategia de Gobierno en Línea.* Obtenido de gobiernoenlinea: http://estrategia.gobiernoenlinea.gov.co/623/articles-7929_normatividad.pdf

MDA. (27 de Mayo de 2015). *MDA - The Architecture Of Choice For A Changing World*. Obtenido de Object Managment Group: http://www.omg.org/mda/

MDA Guide Version 1.0.1. (2003, Junio 12). *MDA Guide*.

MECI. (2014). *Manual Técnico del Modelo Estándar de Control Interno para el Estado Colombiano*. Obtenido de http://portal.dafp.gov.co/form/formularios.retrive_publicaciones?no=2162

Nájera, K. (2013). Desarrollo de software dirigido por modelos y ontologías. *Software Gurú, Núm. 39*.

Natalya F. Noy, D. L. (2000). Ontology Development 101: A Guide to Creating Your. *Stanford University*.

Nicola, A., Missikoff, M., & Navigli, R. (2005). A proposal for a unified process for ontology building: Upon. En K. Andersen, J. Debenham, & R. Wagner, *Lecture Notes in Computer Science* (págs. 655-664). Database and Expert Systems Applications.

OE-Gov. (2010). *Ontologies for e-Government*. Obtenido de Ontologies for e-Government: http://oegov.org/

OMG. (2001). *MDA - The Architecture Of Choice For A Changing World*. Obtenido de http://www.omg.org/mda/

OMG. (06 de Septiembre de 2010). The MDA Foundation Model. *The MDA Foundation Model*.

OMG. (Junio de 2014). OMG MDA Guide rev. 2.0. *OMG MDA Guide*.

OpenGroup. (2013). *ArchiMate® 2.1 Specification Motivation Extension*. Obtenido de http://pubs.opengroup.org/architecture/archimate2-doc/chap10.html

Orozco, M. D., Giraldo, O. W., & Trefftz, G. H. (2013). MDE; MDA; Transformaciones y DSLs. Una breve introducci´on. *Universidad EAFIT*.

Parrend, P., & David, B. (2005). Use of Ontologies as a Way to Automate MDE. *EUROCON*.

Pascuas, S., Mendoza, J., & Córdoba, E. (2014). Desarrollo Dirigido por Modelos (MDD) en el Contexto Educativo. *Scientia et Technica, Vol. 20, No. 2*.

PJoanneum, F., & Salhofer, P. (2012). Semantic MDA for E-Government Service Development. *System Science (HICSS), 2012 45th Hawaii International Conference*.

Poole, J. (2001). Model-Driven Architecture: Vision, Standards And Emerging Technologies. *Workshop on Metamodeling and Adaptive Object Models*.

R. Neches, R. F. (1991). Enabling technology for knowledge sharing. *AI Magazine*, 36-56.

RM ODP. (s.f.). Obtenido de http://www.rm-odp.net/

Servetto, A., & López, G. (s.f.). Ontología de Datos de Sistemas de Gestión para Arquitecturas Dirigidas por Modelos. *Facultad de Ingeniería de la Universidad de Buenos Aires*, 2011.

Siddiqui, F., & Alam, A. (2010). Aspect Weaving in MDA for Ontology Modelling. *Second International Conference on Computer Engineering and Applications*.

Standford, U. (s.f.). *http://protege.stanford.edu/overview/*. Obtenido de http://protege.stanford.edu/overview/

Stanford, U. (2016). *Protégé*. Obtenido de Protégé: http://protege.stanford.edu/

Sure, Y., & Studer., R. (2002). *On-To-Knowledge Methodology*. On-To-Knowledge deliverable 18,Institute AIFB, University of Karlsruhe.

Swartout, B., Ramesh, P., Knight, K., & Russ, T. (1997). Toward Distributed Use of Large-Scale Ontologies. *AAAI Symposium on Ontological Engineering.* Stanford.

Tarazona Bermúdez, G. M. (18 de Septiembre de 2014). *Repositorio Institucional Universidad de Oviedo*. Recuperado el 01 de Mayo de 2016, de Repositorio

Institucional Universidad de Oviedo: http://dspace.sheol.uniovi.es/dspace/handle/10651/29869

The Eclipse Fundation. (2015). *Graphical Modeling Project (GMP).* Obtenido de eclipse.org: https://eclipse.org/modeling/gmp/

Truyen, F. (2005). *Implementing Model Driven Architecture using Enterprise Architect.* Obtenido de http://www.sparxsystems.com.au/: http://www.sparxsystems.com.au/downloads/whitepapers/EA4MDA_White_Paper_Practice.pdf

Uschold, M., King, M., Mooralee, S., & Zorgios, Y. (1996). *The enterprise ontology.* The University of Edinburgh: Technical Report AIAI-TR-195. Artificial Intelligence.

W3C. (s.f.). *WORLD WIDE WEB CONSORTIUM*. Obtenido de http://www.w3.org/XML/

Wongthongtham, P., E, C., Dillon, T., & Sommerville, E. (2006). Ontology-based multi-site software development methodology and tools. *Journal of Systems Architecture, vol. 52, no.11*, 640-653.

Y, D., & S, F. (2002). Ontology research and development. Part 1 . *Journal of Information Science*, 123-36.

Y. Rhazali, Y. H. (2014). Transformation Method CIM to PIM: From Business. *International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:8, No:8*.

Zapata, C., Giraldo, G., & Urrego, G. (2010). Ontologies in software engineering: approaching two great knowledge areas. *Scielo vol.9 no.16*.

Zdravkovic, J. (2010). A Model-Driven Approach for Designing E-Services Using Business Ontological Frameworks. *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*.