



**EDUCACIÓN**  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO  
NACIONAL DE MÉXICO

Instituto Tecnológico de Orizaba

**DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN**

**OPCIÓN I.- TESIS**

**TRABAJO PROFESIONAL**

**“Desarrollo del módulo para la transformación  
de los modelos XMI de cliente y de navegación  
al metamodelo MOF para la herramienta SODRA”**

**QUE PARA OBTENER EL GRADO DE:  
MAESTRO EN SISTEMAS  
COMPUTACIONALES**

**PRESENTA:**

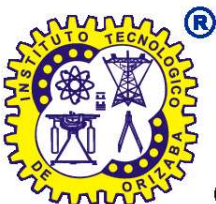
*I.T.I. Carlos Heriberto Hernández Jácome*

**DIRECTOR DE TESIS:**

*Dr. Ulises Juárez Martínez*

**CODIRECTOR DE TESIS:**

*M.C.E. Beatriz Alejandra Olivares Zepahua*



**ORIZABA, VERACRUZ, MÉXICO.**

**JUNIO 2023**



**EDUCACIÓN**  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO  
NACIONAL DE MÉXICO

Instituto Tecnológico de Orizaba  
División de Estudios de Posgrado e Investigación

Orizaba, Veracruz, **03/junio/2023**  
Dependencia: **División de Estudios de  
Posgrado e Investigación**  
Asunto: **Autorización de Impresión**  
OPCION: I

**C. CARLOS HERIBERTO HERNÁNDEZ JÁCOME**  
**CANDIDATO A GRADO DE MAESTRO EN:**  
**SISTEMAS COMPUTACIONALES**  
**P R E S E N T E.-**

De acuerdo con el Reglamento de Titulación vigente de los Centros de Enseñanza Técnica Superior, dependiente de la Dirección General de Institutos Tecnológicos de la Secretaría de Educación Pública y habiendo cumplido con todas las indicaciones que la Comisión Revisora le hizo respecto a su Trabajo Profesional titulado:

" Desarrollo del módulo para la transformación de los modelos XMI de cliente y de navegación al metamodelo MOF para la herramienta SODRA"

comunico a Usted que este Departamento concede su autorización para que proceda a la impresión del mismo.

**ATENTAMENTE**

Excelencia en Educación Tecnológica  
CIENCIA - TÉCNICA - CULTURA

*Cauhtémoc Sánchez R.*

**DR. CAUHTÉMOC SÁNCHEZ RAMÍREZ**  
**JEFE DE LA DIVISIÓN DE ESTUDIOS**  
**DE POSGRADO E INVESTIGACIÓN**



OG-13-F06



Av. Oriente 9 Núm.852, Colonia Emiliano Zapata. C.P. 94320 Orizaba, Veracruz.  
Tel. 01 (272)1105360 e-mail: dir\_orizaba@tecnm.mx tecnm.mx | orizaba.tecnm.mx



**2023**  
**Francisco**  
**VILLA**

Orizaba, Veracruz, **24/abril/2023**  
Asunto: Revisión de trabajo escrito

**C. CUAUHTÉMOC SÁNCHEZ RAMÍREZ**  
**JEFE DE LA DIVISIÓN DE ESTUDIOS**  
**DE POSGRADO E INVESTIGACIÓN**  
**PRESENTE.-**

Los que suscriben, miembros del jurado, han realizado la revisión de la Tesis del (la) C.

**CARLOS HERIBERTO HERNÁNDEZ JÁCOME**

La cual lleva el título de:

**Desarrollo del módulo para la transformación de los modelos XMI de cliente y de navegación al metamodelo MOF para la herramienta SODRA**

Y concluyen que se acepta.

**ATENTAMENTE**  
Excelencia en Educación Tecnológica®  
CIENCIA - TÉCNICA - CULTURA®

**PRESIDENTE: DR. ULISES JUÁREZ MARTÍNEZ**



---

FIRMA

**SECRETARIO: M.C.E. BEATRIZ A. OLIVARES ZEPAHUA**



---

FIRMA

**VOCAL: M.C. MA. ANTONIETA ABUD FIGUEROA**



---

FIRMA

**VOCAL SUP.: DRA. LISBETH RODRÍGUEZ MAZAHUA**



---

FIRMA

TA-09-18



## *CARTA DE ORIGINALIDAD*

En la ciudad de Orizaba, Veracruz, el día 22 del mes de junio del año 2023, el (la) que suscribe Carlos Heriberto Hernández Jacome, alumno (a) del programa de Maestría en Sistemas Computacionales con número de control M19011613, manifiesta que es autor (a) del trabajo de tesis titulado “Desarrollo del módulo para la transformación de los modelos XMI de cliente y de navegación al metamodelo MOF para la herramienta SODRA” y declara que el trabajo es original, ya que su contenido es producto de su directa contribución intelectual. Todos los datos y las referencias a materiales ya publicados están debidamente identificados con su respectivo crédito e incluidos en las notas bibliográficas y en las citas que se destacan como tal y, en los casos que así lo requieran, se tienen las debidas autorizaciones de quienes poseen los derechos patrimoniales. Por lo tanto, se hace responsable de cualquier litigio o reclamación relacionada con derechos de propiedad intelectual, exonerando de toda responsabilidad al Tecnológico Nacional de México / Instituto Tecnológico de Orizaba.

  
Carlos Heriberto Hernández Jacome  
Nombre y firma autógrafa



## CARTA DE CESIÓN DE DERECHOS

En la ciudad de Orizaba, Veracruz, el día 22 del mes de junio del año 2023, el (la) que suscribe Carlos Heriberto Hernández Jacome alumno (a) del programa de Maestría en Sistemas Computacionales con número de control M19011613, manifiesta que es autor (a) intelectual del trabajo de tesis bajo la dirección del Dr. Ulises Juárez Martínez y ceden los derechos del trabajo intitulado “Desarrollo del módulo para la transformación de los modelos XMI de cliente y de navegación al metamodelo MOF para la herramienta SODRA” al TecNM/Instituto Tecnológico de Orizaba para su difusión, con fines académicos y de investigación.

Queda estrictamente prohibido reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del Tecnológico Nacional de México/Instituto Tecnológico de Orizaba. Este puede obtenerse escribiendo a la siguiente dirección: [msc@orizaba.tecnm.mx](mailto:msc@orizaba.tecnm.mx). Si el permiso se otorga, cualquier usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



Carlos Heriberto Hernández Jacome  
Nombre y firma

## **Agradecimiento**

Dedico el presente trabajo a las personas que han estado presentes desde mi familia, amigos y compañeros, estoy seguro que sin ellos, no habría logrado este gran paso. A mis padres, agradezco el tiempo dedicado a mi educación con grandes valores que me inspiran cada día a ser una mejor versión de mí, gracias a ustedes soy el resultado de lo que han construido en estos años, agradezco su confianza en mí, son mi fuerza en los tiempos difíciles. A mis hermanas, Karla y Gloria, por ser ejemplo a seguir y apoyarme en mi etapa de crecimiento con paciencia. Mi esfuerzo es dedicado a Cesar Antonio, espero que mis acciones sean ejemplo de superación y trabajo. A Sandra García, por acompañarme este tiempo y que me recordaba todos los días que no importaba el reto, todo lo que quisiera lo lograría. A mi grupo de amigos de infancia y adolescencia, por recordarme el motivo de tantos sacrificios y conseguir la admiración de ustedes, gracias a ustedes nunca estuve solo. A mi grupo de amigos de profesión por compartir enseñanzas y apoyo en momentos académicos y personales difíciles. Por último, pero no menos importante, mis amigos de robótica por la confianza y enseñanzas, estaré agradecido eternamente por las grandes experiencias de vida.

Al M.C. Silvestre Gustavo Peláez Camarena, que en paz descansé, tuve la oportunidad de conocer una persona con grandes ideas y la comprensión de un mundo diferente a partir de su sabiduría, me hubiera gustado compartir más tiempo con usted y fuera testigo del trabajo de SODRA terminado. A mi director de tesis, el Dr. Ulises Juárez Martínez, gracias por adoptarme y apoyarme a realizar este proyecto, su dirección fue de mucha ayuda. A la M.C.E. Beatriz Olivares Zepahua, sin su ayuda la realización de este proyecto no sería posible. Así mismo, agradezco a cada uno de los miembros del jurado por su tiempo en cada revisión y retroalimentación. Al Consejo Nacional de Ciencia y Tecnología, CONACYT, por el apoyo de la beca de manutención otorgada para la realización de mis estudios.

## Índice general

Índice de tablasxii

Índice de listadosxiii

Tabla de acrónimosxiv

Resumenxvi

Abstractxvii

Introducciónxviii

Capítulo 1. Antecedentes1

1.1 Marco teórico1

1.1.1 Grafo1

1.1.2 Teoría de grafos1

1.1.3 Modelo2

1.1.4 SODRA2

1.1.5 Modelo del cliente3

1.1.6 Modelo navegacional6

1.1.7 Diagrama8

1.1.8 MOF (*Meta Object Facility*, Facilidad de objeto metamodelo)9

1.1.9 XML10

1.1.10 XMI (*XML Metadata Interchange*, Intercambio de metadatos XML)10

1.1.11 QVT (*Query/View/Transformation*, Consulta/Vista/Transformación)11

1.1.12 ATL (*ATLAS Transformation Language*, Lenguaje de Transformación ATLAS)12

1.1.13 ANTLR13

1.1.14 Java14

1.2 Situación tecnológica, económica y operativa de la empresa15

1.3	Planteamiento del problema	16	
1.4	Objetivo general y específicos	17	
1.4.1	Objetivo general	18	
1.4.2	Objetivos específicos	18	
1.5	Justificación	18	
Capítulo 2. Estado de la práctica			20
2.1	Trabajos relacionados	20	
2.2	Análisis comparativo	27	
2.3	Propuesta de solución	36	
Capítulo 3. Aplicación de la metodología			39
3.1	Descripción de la solución	39	
3.2	Metodología de desarrollo en Espiral	40	
3.2.1	Primera iteración	40	
3.2.2	Segunda iteración	67	
3.2.3	Tercera iteración	72	
3.2.4	Cuarta iteración	74	
Capítulo 4. Resultados			80
4.1	Planteamiento del caso de estudio	80	
4.1.1	Generación de metamodelo MOF del modelo de cliente	81	
4.1.2	Generación de código a partir del metamodelo MOF del modelo cliente	83	
4.1.3	Generación de metamodelo MOF del modelo navegacional	85	
Capítulo 5. Conclusiones y recomendaciones			88
5.1	Conclusiones	88	



## **índice de figuras**

- Figura 1.1 Representación de un grafo simple con variables V, E2
- Figura 1.2 Notación del modelo cliente de Gámez et al. [7]4
- Figura 1.3 Modelo del cliente en la herramienta SODRA6
- Figura 1.4 Modelo navegacional en la herramienta SODRA8
- Figura 1.5 Esquema General de Metamodelos9
- Figura 1.6 Configuración inicial de ATL de Jouault et al. [16]12
- Figura 1.7 Fases de las aplicaciones de lenguajes [18]14
- Figura 3.1 Esquema de la solución propuesta39
- Figura 3.2 Arquitectura del sistema46
- Figura 3.3 Proceso de Compilación/Traducción48
- Figura 3.4 Estructura general del modelo XMI48
- Figura 3.5 Estructura del nodo xml:properties del modelo XMI49
- Figura 3.6 Estructura del nodo sodra:model del modelo XMI50
- Figura 3.7 Nodo de concepto51
- Figura 3.8 Nodo de relación51
- Figura 3.9 Nodo de nota51
- Figura 3.10 Nodo de lista52
- Figura 3.11 Nodo de proceso52
- Figura 3.12 Nodo navegacional52
- Figura 3.13 Nodo de menú52
- Figura 3.14 Nodo de consulta53
- Figura 3.15 Nodo de lista53
- Figura 3.16 Nodo de proceso53
- Figura 3.17 Nodo de nota53
- Figura 3.18 Estructura detallada del nodo sodra:model del modelo XMI54
- Figura 3.19 Estructura del nodo sodra:diagram del modelo XMI navegacional a la derecha y modelo XMI de cliente a la izquierda55
- Figura 3.20 Ejemplo de la obtención de tokens de una etiqueta XML57
- Figura 3.21 ATS de modelo XMI59

Figura 3.22 Archivos Java generados por ANTLR4 para el reconocimiento y recorrido del del contenido de los archivos XMI61

Figura 3.23 Paquete gram del módulo del generador MOF62

Figura 3.24 Diagrama de clases del paquete sodra.moduloMOF.gram63

Figura 3.25 Diagrama de clases del paquete sodra.moduloMOF.symbols64

Figura 3.26 Estructura de la aplicación al finalizar el primer ciclo de la espiral67

Figura 3.27 AST de la representación intermedia67

Figura 3.28 Fragmento del diagrama de clases de la representación intermedia del cliente68

Figura 3.29 Conjunto de clases abstractas para la generación de código70

Figura 3.30 Estructura de la aplicación al finalizar el segundo ciclo de la espiral72

Figura 3.31 Diagrama de clases para servicio REST73

Figura 3.32 Estructura de la aplicación al finalizar el tercer ciclo de la espiral74

Figura 3.33 Arquitectura del sistema SODRA75

Figura 3.34 Ventana pop-up con opciones para la exportación del modelo.76

Figura 3.35 Diagrama de paquetes de la herramienta SODRA78

Figura 3.36 Estructura de la aplicación al finalizar el cuarto ciclo de la espiral79

Figura 4.1 Modelo de cliente desarrollado en SODRA para caso de estudio de Agenda81

Figura 4.2 Notación del diagrama del cliente del caso de estudio82

Figura 4.3 Exportación de MOF en SODRA Modelo de cliente desarrollado en SODRA para caso de estudio de Agenda82

Figura 4.4 Archivos resultantes de la exportación MOF del diagrama del cliente83

Figura 4.5 Proyecto generado en EMF a partir del metamodelo agenda.xmi exportado de SODRA84

Figura 4.6 Creación EMF Editor plug-ins85

Figura 4.7 Modelo navegacional desarrollado en SODRA para caso de estudio de Agenda86

Figura 4.8 Notación del diagrama navegacional del caso de estudio86

Figura 4.9 Exportación de MOF en SODRA Modelo navegacional desarrollado en SODRA para caso de estudio de Agenda87

Figura 4.10 Archivos resultantes de la exportación MOF del diagrama navegacional87

## **Índice de tablas**

Tabla 1.1 Notaciones con grafos del modelo navegacional de Gámez et al. [1]	7
Tabla 2.1 Análisis comparativo de los artículos relacionados.	27
Tabla 2.2 Alternativa de solución.	36
Tabla 3.1 Requerimiento de usuario	41
Tabla 3.2 Requerimientos del sistema relativos al requerimiento de usuario	141
Tabla 3.3 Requerimientos del sistema relativos al requerimiento de usuario	242
Tabla 3.4 Requerimientos del sistema relativos al requerimiento de usuario	442
Tabla 3.5 Requerimientos del sistema relativos al requerimiento de usuario	442
Tabla 3.6 Análisis de los principales riesgos	44
Tabla 3.7 Estrategia para la gestión del riesgo	45
Tabla 3.8 Equivalencia de representación intermedia en nodos y diagrama de clases según el modelo cliente y navegacional	69

## **Índice de listados**

Listado 1.1 Estructura básica de XML10

Listado 1.2 Estructura básica de XMI11

Listado 3.1 Fragmento extraído de SodraXMILexer.g458

Listado 3.2 Archivo completo SodraXMIParser.g460

Listado 3.3 Comprobación de consistencia en EscuchaVuelta1.java65

Listado 3.4 Comprobación de verificación en EscuchaVuelta2.java66

Listado 3.5 Generación de modelo MOF71

Listado 3.6 Fragmento de código de diagrama.js75

Listado 3.7 Fragmento de código de proyecto.php77

## Tabla de acrónimos

<b>Término</b>	<b>Concepto</b>
<b>SODRA</b>	Sistema de Objetos de Diagramación Relacional Amigable
<b>ANTLR</b>	“ANOther Tool for Language Recognition”, Otra Herramienta para el Reconocimiento de Lenguaje
<b>AJAX</b>	“Asynchronous JavaScript And XML”, JavaScript asíncrono y XML
<b>API</b>	Application Programming Interface”, Interfaz de Programación de Aplicaciones
<b>AST</b>	“Abstract Syntax Tree”, Árbol de Sintaxis Abstracto
<b>CASE</b>	“Computer Aided Software Engineering”, Ingeniería de Software Asistida por Computadora
<b>CIM</b>	“Computer Integrated Manufacturing”, Manufactura Integrada por Computadora
<b>EBNF</b>	"Extended Backus–Naur Form", Notación de Backus-Naur Extendida
<b>EMF</b>	“Eclipse Modeling Framework”, Marco de modelado de Eclipse
<b>GNU</b>	"GNU's Not Unix", GNU No es Unix
<b>GPL</b>	"GNU General Public License", Licencia Pública General de GNU
<b>ID</b>	Identificador
<b>IDE</b>	“Integrated Development Environment”, Entorno de Desarrollo Integrado
<b>JPG</b>	
<b>JSON</b>	“JavaScript Object Notation”, Notación de objetos de JavaScript
<b>LL</b>	"Left-to-right" y "Leftmost derivation", De izquierda a derecha y Derivación a la izquierda
<b>M2M</b>	"Model to Model", Modelo a Modelo
<b>MDA</b>	“Model Driven Architecture”, Arquitectura dirigida por modelos
<b>MDE</b>	“Model-driven Development Environment”, Entorno de desarrollo dirigido por el modelo
<b>MOF</b>	"Meta-Object Facility", Mecanismo de Meta-Objeto
<b>OMG</b>	“Object Management Group”, Grupo de Administración de Objetos
<b>PHP</b>	"Hypertext Preprocessor", Preprocesador de Hipertexto



<b>PIM</b>	"Platform Independent Models", Modelos independientes de la plataforma
<b>PO</b>	Pequeña Organización
<b>PSM</b>	"Platform Specific Model", Modelo específico de plataforma
<b>QVT</b>	"Query/View/Transformation", Consulta/Vista/Transformación
<b>REST</b>	"Representational State Transfer", Transferencia de Estado Representacional
<b>UML</b>	"Unified Modeling Language", Lenguaje Unificado de Modelado UML
<b>XMI</b>	"XML Metadata Interchange", XML de Intercambio de Metadatos XML
<b>XML</b>	"eXtensible Markup Language", Lenguaje de marcado extensible

## Resumen

SODRA (Sistema de Objetos de Diagramación Relacional Amigable) es una herramienta que permite la generación de diagramas del cliente y modelo de navegación utilizando la notación XMI (Extensible Markup Language Metadata Interchange), lo que es fundamental para el desarrollo de software y facilita la labor de los diferentes roles que participan en el proceso de análisis. MDA (Model Driven Architecture) propone un proceso de desarrollo basado en la realización y transformación de modelos, con el objetivo de automatizar el mapeo entre modelos para optimizar tiempo, recursos económicos y humanos, y en el cual la abstracción, automatización y estandarización son los principios fundamentales.

El objetivo del presente trabajo es realizar un análisis de soluciones, tecnologías y antecedentes para el módulo de transformación de modelos XMI a MOF para la herramienta SODRA utilizando técnicas de generación de código, con el fin de permitir la transición entre la generación de los diagramas y su representación en un lenguaje de Meta-modelado. Además, se propone el desarrollo de un módulo que genere el Metamodelo MOF a partir de los diagramas Cliente/navegacional, utilizando técnicas de generación de código como ANTLR4, lo que permitirá a los desarrolladores migrar los modelos desarrollados en SODRA a plataformas más robustas como EMF (Eclipse Modeling Framework) y optimizar los tiempos de desarrollo y corrección de errores durante el mismo.

La idea de la automatización para el mapeo entre modelos es el uso de lenguajes de transformación, que utilizan semántica y operaciones especiales para resolver la problemática observada mediante este trabajo. En resumen, el trabajo propone el desarrollo de un módulo de transformación de modelos XMI a MOF para la herramienta SODRA, utilizando técnicas de generación de código para permitir la transición entre la generación de los diagramas y su representación en un lenguaje de Meta-modelado, lo que optimizará los tiempos de desarrollo y corrección de errores en el proceso de desarrollo de software.

## **Abstract**

SODRA (Friendly Relational Diagram Object System) is a tool that enables the generation of client diagrams and navigation models using XMI (Extensible Markup Language Metadata Interchange) notation, which is essential for software development and facilitates the work of different roles involved in the analysis process. MDA (Model Driven Architecture) proposes a development process based on the creation and transformation of models, with the aim of automating the mapping between models to optimize time, economic resources, and human effort, where abstraction, automation, and standardization are fundamental principles.

The objective of this work is to conduct an analysis of solutions, technologies, and background for the XMI to MOF model transformation module for the SODRA tool using code generation techniques. The goal is to enable the transition between diagram generation and representation in a Meta-modeling language. Additionally, the development of a module is proposed to generate the MOF Metamodel from the client/navigation diagrams using code generation techniques like ANTLR4. This will allow developers to migrate the models developed in SODRA to more robust platforms such as EMF (Eclipse Modeling Framework) and optimize development time and error correction during the process.

The idea of automation for model mapping involves the use of transformation languages that employ semantics and special operations to address the observed issues in this work. In summary, this work proposes the development of an XMI to MOF model transformation module for the SODRA tool using code generation techniques. This will enable the transition between diagram generation and representation in a Meta-modeling language, thereby optimizing development time and error correction in the software development process.

## **Introducción**

La herramienta empleada en este trabajo es SODRA donde se construyen y generan los modelos de cliente y navegacional con el uso de una notación definida por Carreón en [1], la cual está fundamentada en el primer principio de la MDA que es la representación directa haciendo énfasis en el dominio del problema más que en la tecnología. Cruz et al. [2] presenta la Arquitectura del generador de aplicaciones Web en código PHP para la herramienta SODRA basado en los modelos de cliente y navegacional, donde destacan la generación de código esqueleto a partir de los modelos mencionados, con el fin de realizar una mejora significativa en los tiempos para los desarrolladores, cumpliendo así el segundo principio de MDA: la automatización. Con base en este principio y el trabajo realizado en [2], se ha propuesto el uso de los metamodelos que provee SODRA en los estándares XMI, transformando el modelo en un metamodelo MOF basado en una simplificación de las capacidades de modelado clase UML2, constituyendo en los medios la posibilidad de integrarse con herramientas robustas de ayuda al desarrollo.

Los modelos son la estructura de un proyecto de software que comienza por la idea y la consolidación de esta, mediante los modelos se contemplan los requisitos funcionales, el modelo de negocio, entre otros, pero la transición entre cada uno toma tiempo en las etapas de análisis y diseño. Además, si ocurre un error en alguna etapa o artefacto deben actualizarse en cadena repercutiendo en el tiempo, en la calidad y económicamente. La propuesta MDA fortalece y agiliza el papel que juegan las herramientas CASE en el desarrollo de soluciones. Cuando aparecen nuevas funcionalidades como el cambio de modelos, verificación de consistencia, transformación de modelos y gestión de metamodelos, entre otros, deben ser soportadas por las herramientas. Desde la perspectiva de MDA, el papel de las herramientas CASE es fundamental para apoyar el proceso de desarrollo de una manera coherente y sistemática, visto como un proceso de transformación del modelo.

La idea de la automatización para el mapeo entre modelos es el uso de lenguajes de transformación, los cuales usan semántica y operaciones especiales, resolviendo la problemática observada mediante este trabajo al analizar y comparar las posibles soluciones.

Este trabajo se organiza en cinco capítulos principales: en el capítulo uno se abarca el marco teórico, planteamiento del problema, objetivo general y específicos y la justificación; el capítulo dos contempla el estado de la práctica y un análisis comparativo para la propuesta de solución específica, la descripción de la solución, el cronograma de actividades, los lenguajes de programación y los lenguajes de transformación; el capítulo tres abarca la metodología de desarrollo de software con la solución propuesta, presenta la arquitectura del módulo y explicación de cada uno de sus componentes, también el proceso de transformación explica las fases usadas del proceso de compilación juntos con artefactos creados para el desarrollo de la solución con ANTLR4.

## **Capítulo 1. Antecedentes**

En este capítulo se muestran los conceptos que tienen relación con el presente proyecto de tesis. Además, se expone la descripción del problema actual que da lugar a la continuación de este proyecto, se presentan el objetivo general y los objetivos específicos del mismo, y por último, se concluye con la explicación de la justificación detrás del proyecto.

### **1.1 Marco teórico**

En este apartado, se explican los principales conceptos que están relacionados con el presente trabajo.

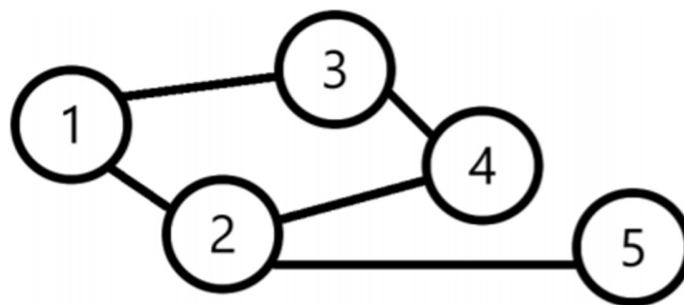
#### **1.1.1 Grafo**

Un grafo es una descripción visual que puede mostrar información en un contexto espacial. Específicamente, como el punto entre dos coordenadas geográficas y las trazas resultantes inventa figuras geométricas. En una perspectiva diferente, según [3], un grafo es una colección de elementos que están interconectados para crear una relación entre ellos. Esta colección de elementos puede estar compuesta por vértices, enlaces y nodos de gráficos dirigidos o no dirigidos, según las necesidades y aplicaciones específicas, y se elegirá uno de ellos para su uso.

#### **1.1.2 Teoría de grafos**

La teoría de grafos, descrita en [4], es un campo de la ciencia exacta e informática que se encarga de comprender y estudiar los detalles de los grafos para aplicarlos en diferentes áreas, como álgebra, teoría de conjuntos, topología y aritmética básica. Estos grafos están compuestos por dos variables:  $V$  y  $E$ .  $V$  representa el número de nodos y  $E$  representa el número de enlaces entre ellos para formar una representación gráfica [4]. Se utilizan los nodos  $V = \{1,2,3,4,5\}$  y  $E = \{\{1,2\}, \{2,4\}, \{3,4\}\}$  para representar gráficamente un gráfico  $\{2,5\}, \{1,3\}$ , descrito en la Figura 1.1. Existen varios tipos de gráficos, como gráficos dirigidos, mixtos, múltiples, simples y no dirigidos. Sin embargo, se presenta un gráfico dirigido que representa el contexto básico de este trabajo.





**Figura 1.1 Representación de un grafo simple con variables  $V, E$**

### **1.1.3 Modelo**

Booch et al. [5] define un modelo como una simplificación de la realidad, que consiste en representar elementos de manera simplificada para facilitar su comprensión antes de su construcción, a través del uso de notaciones, esquemas o información adicional. Estos modelos tienen elementos generales correctamente definidos, que ayudan a construir, resolver e identificar la problemática.

Pressman [6] sostiene que para resolver problemas reales de desarrollo de software, es necesario adoptar una estrategia de desarrollo que acompañe al proceso, los métodos, las herramientas y las fases genéricas ya descritas, que se conoce como modelo. Este modelo se determina de acuerdo con la naturaleza del proyecto, la aplicación, los métodos y las herramientas que se utilizarán, así como los controles y las entregas que se requieren.

En el desarrollo de software, se realiza un ciclo de resolución de problemas, que consta de cuatro etapas distintas: estado actual, definición de problemas, desarrollo técnico e integración de soluciones. A veces, estas cuatro etapas se aplican no solo para la solución de un problema general, sino también para la solución de pequeños problemas más específicos en un modelo recursivo.

### **1.1.4 SODRA**

Según lo presentado por Carreón [1], se ha desarrollado una herramienta llamada SODRA (Sistema de Objetos de Diagramación Relacional Amigable), la cual se basa en los artefactos de los modelos del cliente y navegación para generar un entorno de desarrollo amigable e intuitivo de los diagramas a través de una

aplicación web. Esta herramienta es capaz de generar los diagramas de cliente y navegacionales y produce la notación XMI correspondiente para la representación de estos diagramas, así como su correspondiente imagen en formato JPG.

Un ejemplo de la herramienta SODRA usado en la práctica es que un equipo de desarrollo de software está trabajando en la creación de una aplicación web compleja y necesita visualizar los diferentes diagramas de cliente y navegación para poder planificar mejor la estructura de la aplicación.

En este caso, el equipo podría utilizar SODRA para generar de manera rápida y eficiente los diagramas correspondientes a partir de los artefactos de los modelos existentes. Esto les permitiría tener una vista clara de los diferentes componentes de la aplicación y cómo se relacionan entre sí, lo que a su vez les ayudaría a tomar decisiones informadas sobre la arquitectura y el diseño de la aplicación.

Otro ejemplo de uso de SODRA podría ser en el contexto de la enseñanza de la programación. Un profesor podría utilizar la herramienta para generar diagramas de cliente y navegación que muestren cómo funciona un programa específico, lo que ayudaría a los estudiantes a entender mejor los conceptos clave de programación y cómo se relacionan entre sí.

En resumen, SODRA es una herramienta versátil y útil para cualquier persona que necesite generar diagramas de cliente y navegación de manera eficiente y efectiva.

#### **1.1.5 Modelo del cliente**

De acuerdo con la propuesta de Gámez et al. [7], se propone el uso de conjuntos y grafos como artefactos para el modelado del contenido y la navegación, respectivamente. Esto conduce a la creación de un modelo de clases con una notación basada en la teoría de conjuntos, y un modelo de navegación con una notación que utiliza símbolos propios de la navegación en una aplicación Web.

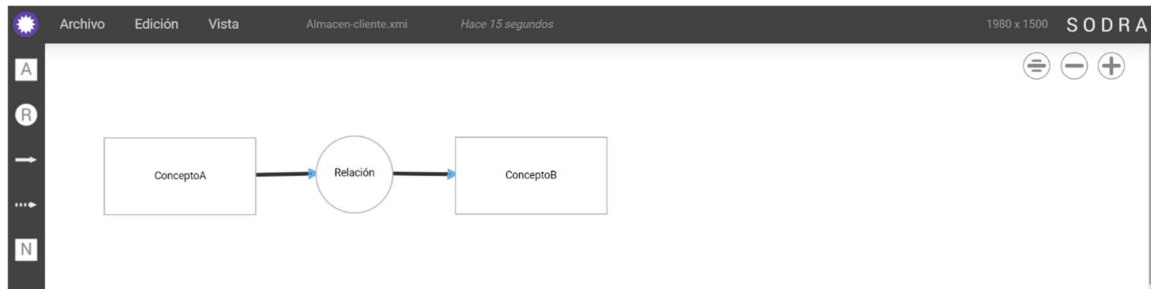
Para ilustrar, en la notación del modelo de clases basado en conjuntos, se utiliza un bloque que incluye el nombre de la clase en el primer elemento, seguido de un conjunto de atributos en el segundo elemento, y finalmente los métodos que contiene la clase en el tercer elemento. Por ejemplo, en un modelo de clases para



- La primera letra de la clase debe ser mayúscula y los atributos o métodos deben separarse por comas.
- Si no hay atributos o métodos, se representarán utilizando el conjunto vacío.
- Si la clase es abstracta, su nombre comenzará con "Abs\_".
- Si se trata de una interfaz, su nombre comenzará con "In\_".
- Para representar una clase que implementa una interfaz, se utilizará el símbolo de implicación.
- El símbolo de unión de conjuntos se usará para representar la herencia.
- Los símbolos  $\in$  y  $\subset$  se emplearán para representar la composición y agregación de clases, respectivamente.
- El símbolo  $\cap$ , junto con sus diferentes multiplicidades (1-1, 1-\*, -), se usará para establecer la asociación entre clases.
- Para representar una clase de asociación, se concatenará el nombre de la primera clase, seguida de la clase de asociación y finalizando con la segunda clase.

Por ejemplo, para representar una clase de cliente que hereda de una clase abstracta "Persona", se utilizará el símbolo de unión de conjuntos. Si la clase "Cliente" tiene una asociación de "1 a muchos" con la clase "Pedido", se utilizará el símbolo  $\cap$  con la multiplicidad "1-\*". Si se tiene una clase de asociación "DetallePedido" que conecta la clase "Pedido" y la clase "Producto", se representará como "Pedido-DetallePedido-Producto".

También, dentro del modelo del cliente se incluyen componentes como el rectángulo, los cuales indican los nodos conceptuales; los círculos, que representan los nodos relacionales; y las flechas, que indican las conexiones entre los nodos conceptuales y relacionales. La Figura 1.3 presenta una ilustración de estos elementos en la herramienta SODRA.





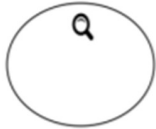

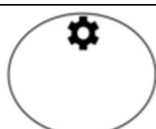
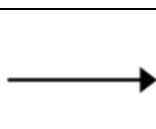

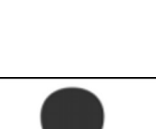
**Figura 1.3 Modelo del cliente en la herramienta SODRA**

### 1.1.6 Modelo navegacional

El modelo de navegación para la Web es específico para determinar cómo están conectadas las diferentes páginas de un sitio web. Según la Universidad Ludwig-Maximilians de München [9], este modelo está compuesto por nodos y enlaces, donde los nodos representan unidades de navegación y están conectados por enlaces. Estos nodos pueden estar en diferentes páginas o en la misma página, y esto depende del nivel de abstracción que se utilice para representar el flujo y los elementos del modelo.

En cuanto al modelo navegacional, se utilizan símbolos que representan las diferentes pantallas y se establecen las relaciones entre ellas, mostrando cómo se puede navegar entre ellas. Por ejemplo, en una aplicación de compras en línea, se podría tener una pantalla de inicio, seguida de una pantalla de búsqueda de productos, una pantalla de detalles del producto y una pantalla de pago, y se mostraría cómo el usuario puede pasar de una pantalla a otra en una secuencia lógica y coherente.

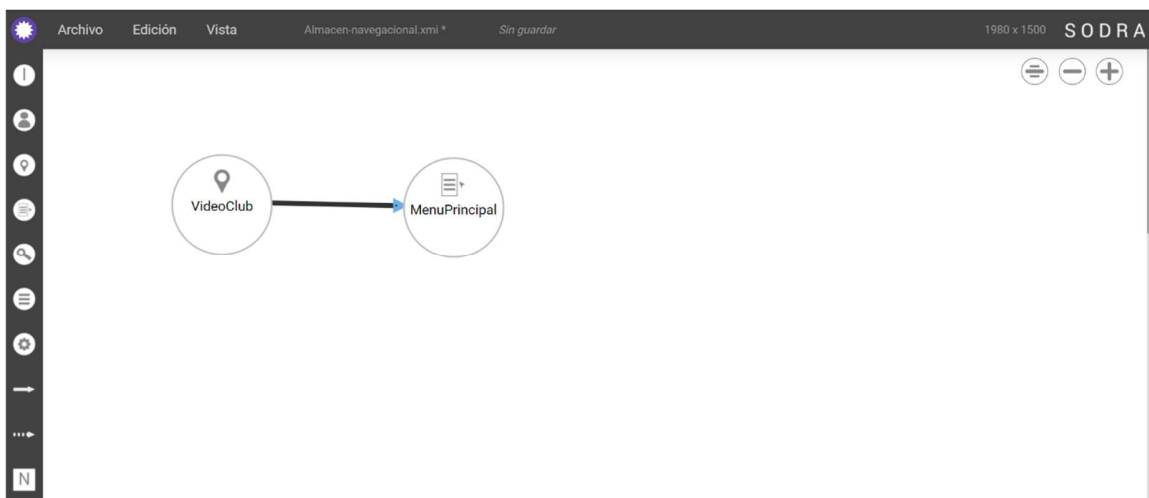
**Tabla 1.1 Notaciones con grafos del modelo navegacional de Gámez et al. [1]**

Símbolo	Nombre	Descripción
	<i>Navigation Node</i>	Representa un nodo de navegación y se muestra la información al usuario.
	<i>MenuNode node</i>	Representa un menú para manejar diferentes opciones de navegación.
	<i>Query Node</i>	Representa puntos de la navegación donde solicita información al usuario.
	<i>Index Node</i>	Representa puntos de navegación en una lista de posibles resultados a visualizar.
	<i>Process Node</i>	Representan procesos/tareas a realizar en la aplicación.
	<i>Navigation Link</i>	Representan un <i>link</i> de navegación de un nodo de navegación a otro, no se usan con nodos de proceso.
	<i>Process link</i>	Representa un <i>link</i> de proceso que va de un nodo de proceso a otro, no se usan para nodos de navegación (para nodos de navegación se usan <i>links</i> de navegación).
	<i>User Item</i>	Sirve para indicar qué usuario tiene acceso a determinado destino de navegación

En el artículo [10], se define el modelo navegacional como las distintas rutas que los usuarios pueden seguir dentro de una aplicación de acuerdo con su rol asignado. Para representar este modelo se utiliza una notación basada en grafos, donde se



describen cada uno de los elementos utilizados y su representación formal definida en [8]. La notación incluye la representación de cada nodo, el tipo de nodo y los enlaces entre ellos. Para representar un nodo se utiliza "(nombre\_nodo)" y para especificar el tipo de nodo se utiliza "{propiedades}" según se indica en la tabla 1.1. Los enlaces entre nodos se representan utilizando "-[EnlaceNavegacional]->" para enlaces de navegación y "-[EnlaceProceso]->" para enlaces de proceso. La Figura 1.4 muestra un ejemplo simple de la relación entre un videoclub y un menú principal, utilizando esta notación.



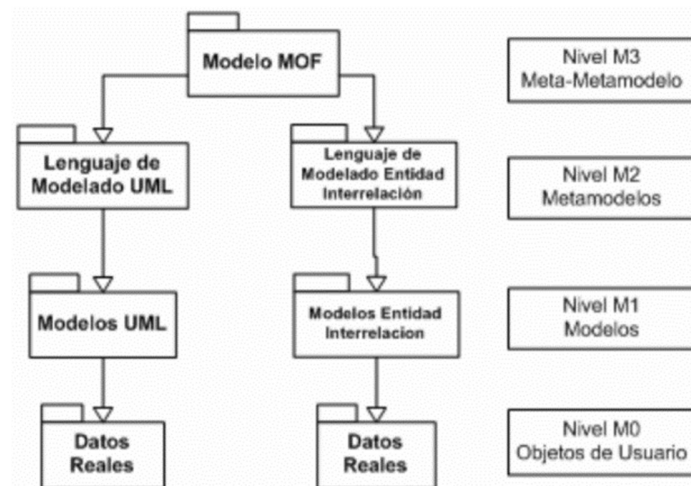
**Figura 1.4 Modelo navegacional en la herramienta SODRA**

### 1.1.7 Diagrama

Según la definición en [11], es una representación gráfica de un modelo, que extiende la estructura, comportamiento o combinación de varios conceptos del sistema. Esto incluye elementos como nombres, atributos, variables y operaciones. En el campo de la informática existen varios diagramas, como los diagramas UML [12], que en conjunto proporcionan una perspectiva visual de los problemas completados o resueltos. Un modelo representa una serie de pasos como punto de partida para operaciones repetibles, como hacer vidrio a partir de un modelo. La diferencia entre estos dos conceptos es que el modelo se aplica a todas las etapas del desarrollo de software, mientras que el diagrama proporciona una explicación intuitiva de algunas partes del sistema.

### 1.1.8 MOF (*Meta Object Facility, Facilidad de objeto metamodelo*)

MOF es una norma internacional que proporciona la base para la definición de metamodelos en la familia de lenguajes MDA (Arquitectura Basada en Modelos) de *Object Management Group* [13] y se basa en una simplificación de las capacidades de modelado de clase de UML2 (*Unified Modeling Language version 2, Lenguaje de modelado unificado versión 2*). Además de proporcionar los medios para la definición del metamodelo, agrega capacidades básicas para la gestión del modelo en general, incluidos identificadores, una capacidad de etiqueta genérica simple y operaciones reflectantes que se definen genéricamente y son aplicadas independientemente del metamodelo.



**Figura 1.5 Esquema General de Metamodelos**

Existen tres versiones, la primera denominada 1.4 surgió en abril del 2002, posteriormente la versión 2.0 en enero 2006 y finalmente y actual versión 2.4.2 publicada en abril de 2014, esta versión se publicó formalmente por ISO (*International Organization for Standardization, Organización Internacional de Normalización*) como el estándar de la edición 2014: ISO/IEC (Comisión Electrotécnica Internacional, *International Electrotechnical Commission*) 19508, además contiene la representación OCL (Lenguaje de restricción de objetos, *Object Constraint Language*) de las restricciones que se aplicarán a un modelo UML para

validar si es un metamodelo CMOF (Instalación completa de Metaobjeto, Complete Meta-Object Facility) válido y ejecutable en el entorno Eclipse OCL.

### 1.1.9 XML

Se trata de un tipo de texto jerárquico o con estructura de árbol que contiene información específica para describir elementos tales como configuraciones, datos y facturas, de manera que pueda ser entendido por aplicaciones que recopilan su contenido con diversos fines. Esta especificación define diferentes aspectos relacionados con la descripción de objetos en XML, como la representación de objetos mediante elementos y atributos XML, mecanismos estándar para vincular objetos dentro de un mismo archivo o entre archivos, validación de documentos XML mediante esquemas XML, e identidad de objeto que permite hacer referencia a objetos de otros objetos en términos de ID y UUID. XMI aborda estos problemas mediante la especificación de reglas de producción EBNF para crear documentos XML y esquemas que permiten compartir objetos de manera coherente. En la siguiente sección se muestra la sintaxis básica de un documento XML.

```
1  <?xml version= "1.0 " encoding= "UTF-8 "?>
2  <raiz>
3  <elemento>Contenido</elemento>
4  <time id="2017-01-01 16:32">Domingo 1 de enero de 2017 a las 16:32</time>
5  <hijo nombre="Fernando" edad="60">
6  <nieto nombre="Juan Manuel" edad="35">
7  <bisnieto nombre="Carlos" edad="5"></bisnieto>
8  </nieto>
9  </hijo>
10 </raiz>
```

### **Listado 1.1 Estructura básica de XML**

#### **1.1.10 XMI (XML Metadata Interchange, Intercambio de metadatos XML)**

La especificación descrita por Object Management Group [14] define los aspectos necesarios para describir objetos en XML, tales como la representación de objetos en términos de elementos y atributos XML, los mecanismos estándar para vincular objetos dentro de un archivo o entre archivos, la validación de documentos XML mediante esquemas XML, y la identidad de objetos para permitir la referencia de objetos desde otros objetos utilizando identificadores y identificadores únicos universales (UUIDs). XMI proporciona soluciones a estos problemas a través de

reglas de producción EBNF que permiten la creación de documentos XML y esquemas que comparten objetos de manera consistente. Han sido publicadas cuatro versiones de XMI: 2.0 en mayo de 2003, 2.1.1 en diciembre de 2007, 2.4.2 en 2014 y la versión 2.5.1 en junio de 2015, todas ellas relacionadas con los estándares MOF y XMI existentes. En el listado 1.2 se muestra un ejemplo de documento XMI que representa el modelo de un automóvil.

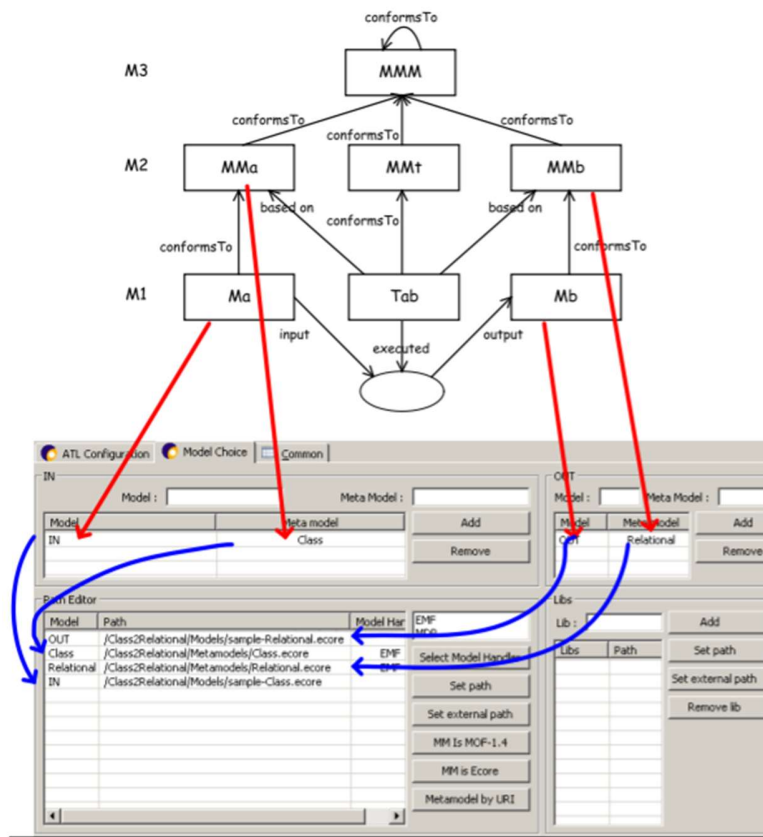
```
1 <?xml version= "1.0" encoding= "UTF-8"?>
2 <xsd:schema
3 xmlns:xmi="http://www.omg.org/XMI"
4 targetNamespace="http://www.ejemplo.com/EspacioAutos"
5 xmlns:cds="http://www.ejemplo.com/EspacioAutos">
6 <xsd:complexType name="Auto">
7 <xsd:choice minOccurs="0" maxOccurs="unbounded">
8 <xsd:element name="marca" type="xsd:string" />
9 <xsd:element name="modelo" type="xsd:string" />
10 <xsd:element ref="xmi:Extension" />
11 </xsd:choice>
12 <xsd:attribute ref="xmi:id" />
13 <xsd:attribute name="marca" type="xsd:string" />
14 <xsd:attribute name="modelo" type="xsd:string" />
15 </xsd:complexType>
16 <xsd:element name="Autos" type="cds:Auto" />
17 </xsd:schema>
```

### **Listado 1.2 Estructura básica de XMI**

#### **1.1.11 QVT (Query/View/Transformation, Consulta/Vista/Transformación)**

En [15] se especifica cómo QVT proporciona la arquitectura, los idiomas, las asignaciones operativas y el lenguaje central para la especificación de consulta, vista y transformación de MOF 2.0 (QVT). La especificación define tres lenguajes de transformación relacionados: relaciones, mapeos operativos y núcleo.

La conformidad del lenguaje QVT se especifica a lo largo de dos dimensiones ortogonales: la dimensión del lenguaje y la dimensión de interoperabilidad. Cada dimensión especifica un conjunto de niveles con nombre. Cada intersección de los niveles de las dos dimensiones especifica un punto de conformidad QVT válido. Todos los puntos de conformidad son válidos por sí mismos, lo que implica que no existe una noción general de “conformidad QVT”. En cambio, una herramienta indicará qué puntos de conformidad implementa. Algunas versiones formales se especifican desde 1.0 en abril de 2008 a la versión 1.3 publicada en junio de 2016.



**Figura 1.6 Configuración inicial de ATL de Jouault et al. [16]**

### 1.1.12 ATL (ATLAS Transformation Language, Lenguaje de Transformación ATLAS)

En el trabajo de Jouault et al. [16] ATL se aplica en el contexto del patrón de transformación, como se observa en Figura 1.2, un modelo de origen *Ma* se transforma en un modelo de destino *Mb* de acuerdo con una definición de transformación *mma2mmb.atl* escrita en el lenguaje ATL. Estos tres elementos son modelos que se ajustan respectivamente a los metamodelos *MMa*, *MMb* y ATL.

Todos los metamodelos se ajustan al metamodelo (MOF en el contexto de los estándares OMG). Los modelos y metamodelos de origen y destino pueden expresarse en XMI. Los metamodelos también usan la notación KM3 más conveniente [14].

### 1.1.13 ANTLR

ANTLR es una herramienta muy útil en proyectos que involucran el análisis de archivos de texto estructurados o binarios. Se utiliza tanto en proyectos a gran escala como en proyectos a pequeña escala, como lectores de archivos de configuración. Al utilizar ANTLR, se puede generar una aplicación de lenguaje que puede ser utilizada para la lectura, procesamiento, ejecución o traducción de los archivos de texto.

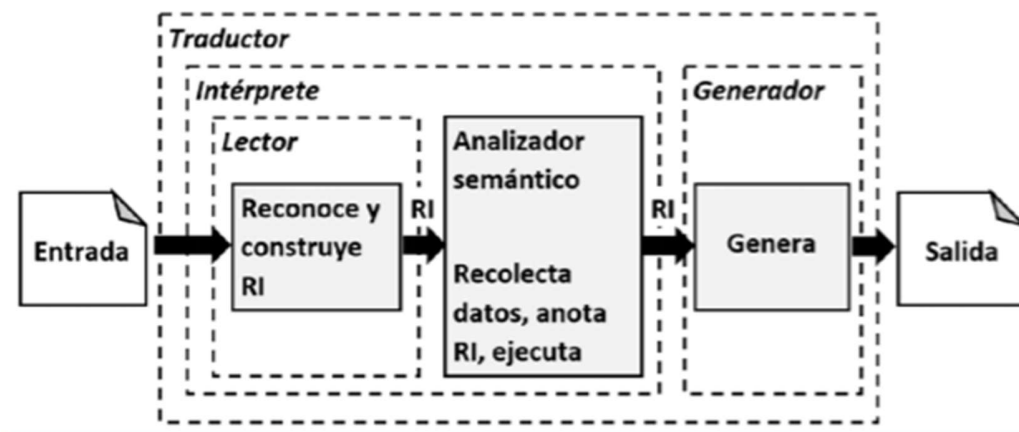
Para generar un analizador sintáctico, se necesita una gramática que describa el lenguaje que se quiere analizar. A partir de esa gramática, ANTLR puede generar un analizador sintáctico que pueda construir árboles de análisis sintáctico, que son la representación de las coincidencias entre la gramática y los datos de entrada. Además, ANTLR también puede generar *walkers*, que son estructuras utilizadas para visitar los nodos de los árboles sintácticos y ejecutar códigos específicos de la aplicación.

ANTLR tiene la capacidad de aceptar cualquier gramática dada sin tener conflictos gramaticales o de ambigüedad. Su funcionamiento se basa en el uso de la tecnología de análisis *AdaptiveLL()* o también conocida como *All()*, la cual realiza el análisis de la gramática dinámicamente en tiempo de ejecución en lugar de hacerlo de forma estática previo a la ejecución del analizador sintáctico generado. Al tener acceso a las sentencias de entrada, ANTLR puede reconocer de forma apropiada las sentencias a través de la gramática [17].

Un ejemplo de uso de ANTLR es en el análisis de un archivo de texto que contiene una lista de nombres y números de teléfono en un formato específico. La gramática para este lenguaje podría ser:

```
phonebook: entry+ ;
entry: name phone ;
name: LETTER+ ;
phone: DIGIT+ '-' DIGIT+ '-' DIGIT+ ;
```

A partir de esta gramática, ANTLR podría generar un analizador sintáctico que pueda leer el archivo de texto y construir árboles de análisis sintáctico. Además, se podrían generar *walkers* para recorrer los árboles y ejecutar códigos específicos de la aplicación, como agregar los nombres y números de teléfono a una base de datos.



**Figura 1.7 Fases de las aplicaciones de lenguajes [18]**

En la imagen 1.8 se ilustran los distintos pasos que conforman las aplicaciones de lenguaje, en los cuales se puede observar la forma en que un traductor recibe una entrada y produce una salida, combinando los componentes fundamentales de un lector y un generador [18].

En resumen, ANTLR es una herramienta muy útil para el análisis de archivos de texto estructurados o binarios. Al utilizar una gramática, se puede generar un analizador sintáctico que pueda construir árboles de análisis sintáctico y *walkers* para ejecutar códigos específicos de la aplicación. Con ANTLR, se pueden analizar gramáticas sin conflictos gramaticales o de ambigüedad y reconocer de forma apropiada las sentencias a través de la gramática.

#### **1.1.14 Java**

Java es un lenguaje de programación y plataforma informática que fue lanzado por primera vez en 1995 por Sun Microsystems. Gracias a su amplia funcionalidad y robustez, se utiliza en una gran cantidad de aplicaciones, lo que facilita el desarrollo de software. Es posible descargar este lenguaje de forma gratuita desde la página oficial de Java [19]. La versión más reciente de Java incluye mejoras significativas

en el rendimiento, la estabilidad y la seguridad de las aplicaciones que se ejecutan en los equipos.

Java es un lenguaje de programación de alto nivel y orientado a objetos, que se ha convertido en uno de los lenguajes más populares en el desarrollo de aplicaciones empresariales y de software en general. Una de las características que lo ha hecho tan popular es su portabilidad, lo que significa que el código escrito en Java se puede ejecutar en diferentes sistemas operativos sin necesidad de hacer cambios en el código fuente.

Java se utiliza en una amplia gama de aplicaciones, desde aplicaciones web y móviles hasta sistemas de gestión empresarial y aplicaciones de procesamiento de datos en tiempo real. La tecnología de Java se compone de dos partes principales: el lenguaje de programación Java y la plataforma Java.

La plataforma Java es un conjunto de herramientas y bibliotecas que los desarrolladores pueden utilizar para crear y ejecutar aplicaciones Java. La plataforma incluye la Máquina Virtual de Java (JVM), que es responsable de ejecutar el código Java, así como bibliotecas para tareas comunes de programación, como la manipulación de datos y la gestión de red. Además, Java cuenta con una gran comunidad de desarrolladores y una amplia gama de recursos en línea, lo que lo hace fácil de aprender y utilizar.

Java es utilizado en empresas de todo el mundo para desarrollar aplicaciones de alta calidad, robustas y escalables. Una de las razones de su popularidad es que Java se considera un lenguaje seguro y confiable. Java cuenta con una serie de características de seguridad incorporadas en su diseño, lo que lo hace menos susceptible a errores y vulnerabilidades comunes en otros lenguajes de programación.

## **1.2 Situación tecnológica, económica y operativa de la empresa**

El proyecto se realizó en la División de Estudios de Posgrado e Investigación (DEPI) del Instituto Tecnológico de Orizaba (dicha institución es una de las instituciones educativas más importantes a nivel nacional), en el programa de Maestría en



Sistemas Computacionales, que tiene el nivel de Consolidado ante el CONACYT, programa creado en el año 2006 y se encuentra vigente dentro del Programa Nacional de Posgrados de Calidad (PNPC) de CONACYT.

El programa de maestría cuenta con las instalaciones adecuadas para la realización de diversos tipos de proyectos que se relacionen con el área de sistemas computacionales, además de que cuenta con personal altamente capacitado profesionalmente para apoyar en la realización de los mismos.

Al encontrarse dentro del PNPC los alumnos pertenecientes a la Maestría en Sistemas Computacionales cuentan con una beca para solventar los gastos que surjan a lo largo del desarrollo del proyecto de tesis

### **1.3 Planteamiento del problema**

Los modelos son herramientas clave en el desarrollo de software ya que permiten la reutilización de elementos y simplifican el trabajo de los diferentes roles involucrados. En este contexto, MDA propone un proceso de desarrollo que se basa en la creación y transformación de modelos. La metodología se sustenta en tres principios fundamentales: abstracción, automatización y estandarización.

El proceso central de MDA es la transformación de modelos, que comienza con la creación de modelos que representan el problema a resolver, conocidos como Modelos de Información Común (CIM). Luego, se procede a la creación de modelos que describen una solución independientemente de la plataforma en la que se ejecutará, conocidos como Modelo independiente de Plataforma (PIM). Por último, se transforma el PIM en modelos específicos de plataforma (Platform-Specific Models o PSM) que describen cómo la solución se implementará en una plataforma en particular.

Actualmente, existe la necesidad de impulsar y aumentar la producción de software y con ello nacieron controladores de lenguajes específicos de dominio, además los ingenieros o desarrolladores de software detallan los sistemas con términos relevantes para el dominio en cambio de mostrarlos en conceptos de propósito general como UML. El uso de metamodelos, donde se expresan los conceptos y la estructura de los posibles modelos (es decir, sintaxis abstracta), se convierten

complejos según sus dominios, y la expresividad de los metamodelos son insuficientes para especificar con precisión el dominio.

Esta propuesta de proyecto es la cuarta etapa del desarrollo de una herramienta CASE. La primera etapa consistió en la creación de una propuesta para el modelado de aplicaciones utilizando grafos y teoría de conjuntos. Su objetivo era establecer una notación formal que permitiera a los clientes del proyecto visualizar claramente el proceso de desarrollo, involucrándose directamente de manera amigable.

La segunda fase del proyecto involucró el desarrollo de una herramienta llamada SODRA, que se basa en los artefactos propuestos y permite la creación de un entorno de desarrollo web amigable e intuitivo para los diagramas de modelo del cliente y modelo navegacional. Esta herramienta genera los diagramas mencionados anteriormente y proporciona una interfaz tecnológica en notación XMI para su representación, así como una imagen correspondiente en formato.

En la tercera etapa del proyecto, se procedió a generar el esqueleto para el desarrollo de aplicaciones en lenguaje PHP, utilizando como base el modelo XMI obtenido en la segunda etapa. Este modelo, que incluye la representación de las clases y sus relaciones que conforman el modelo del cliente, así como las rutas de navegación correspondientes al Modelo Navegacional, se utiliza para crear la estructura inicial del software.

El último módulo de la herramienta SODRA se creó en base al trabajo previo de Gámez [10] y Carreón [1], con el propósito de generar el metamodelo (MOF) a partir de los diagramas Cliente/navegacional, utilizando el modelo XMI para permitir la generación automática del metamodelo esperado. Este módulo es esencial para hacer de SODRA una herramienta CASE funcional y completa, ya que permite la creación de modelos de cliente y navegacional, la generación de código y metamodelos, y ahorra tiempo en la fase de análisis y diseño, permitiendo enfocarse en la parte robusta de las aplicaciones.

#### **1.4 Objetivo general y específicos**

A continuación, se presenta el objetivo general y los objetivos específicos del presente proyecto de tesis.

### **1.4.1 Objetivo general**

Desarrollar un módulo para la herramienta SODRA, que con base en los resultados obtenidos (Modelo XMI), genere el Metamodelo (MOF), a partir de los diagramas Cliente/navegacional, utilizando técnicas de generación de código que permitan la transición entre la generación de los diagramas y su representación en un lenguaje de Meta-modelado.

### **1.4.2 Objetivos específicos**

- Estudiar los artefactos propuestos para el modelado de aplicaciones alternativo.
- Estudiar los modelos XMI generados por la herramienta SODRA.
- Identificar y proponer tecnologías para el desarrollo de la generación de código para el meta-modelado.
- Proponer el procedimiento metodológico para el desarrollo de la herramienta.
- Estudiar ANTLR como analizador y generador para leer, procesar, ejecutar o traducir texto estructurado.
- Generar, a partir de una gramática, un analizador que pueda construir y recorrer árboles de análisis de los modelos XMI de cliente y navegacional.
- Generar el metamodelo MOF del modelo del cliente y el diagrama navegacional.
- Desarrollar casos de estudio mediante el empleo de la herramienta SODRA para validar y probar la funcionalidad del producto obtenido.

### **1.5 Justificación**

Debido a las necesidades presentadas es necesario mostrar en este trabajo un enfoque a partir del modelo de cliente y modelo navegacional que el sistema SODRA ofrece para la generación del metamodelo MOF, de esta manera el tiempo de desarrollo será más corto, ya que este tipo de metamodelo es implantado en Eclipse IDE sin necesidad de generar código de programación, y en cuanto al costo de igual forma será menor por los cambios que puedan ocurrir, al ser un desarrollo

basado en modelos y con el nuevo módulo desarrollado. Los contribuidores principales son los usuarios de SODRA, los cuales además de generar modelo del cliente, modelo navegacional y código PHP, podrán obtener un metamodelo MOF que permitirá la generación de código en Eclipse IDE.

## **Capítulo 2. Estado de la práctica**

En las próximas secciones se presentan algunos estudios previos relacionados con el tema de esta tesis, y se realiza una comparación de los mismos, destacando la información más importante.

### **2.1 Trabajos relacionados**

De acuerdo con Mengerink et al. [20] la ingeniería dirigida por modelos (MDE) se utiliza en la industria para impulsar el aumento de la productividad, algunos de estos controladores es el uso de lenguajes específicos de dominio (DSL) para permitir a los ingenieros especificar sistemas en términos relevantes para su dominio, en lugar de codificarlos en conceptos de propósito general como los de UML. El uso de DSL se respalda por metamodelos donde se expresan los conceptos y la estructura de los posibles modelos (es decir, sintaxis abstracta). Sin embargo, a medida que las DSL crecen en complejidad, la expresividad de los metamodelos solos no suele ser suficiente para especificar con precisión el dominio. Para abordar este problema, [20] propusieron mecanismos más complejos, como el lenguaje de restricción de objetos (OCL).

En [20] se integraron 9173 expresiones OCL derivadas de 504 metamodelos de código abierto, el conjunto de datos con transformaciones de modelo a texto y presentaron varios experimentos que utilizan las 94089 expresiones derivadas de 2634 transformaciones de modelo a texto de código abierto disponibles en GitHub. En esta investigación empírica, el proceso de recopilación de datos está sujeto a varias amenazas a la validez.

Según los hallazgos de Carreón [1], el desarrollo de aplicaciones web es un proceso que requiere una cantidad significativa de tiempo y esfuerzo para ser construido adecuadamente. Para comprender mejor el problema y ofrecer una solución eficiente que satisfaga las necesidades del cliente, se analizan los requisitos y se obtiene retroalimentación. En este contexto, se planteó la creación de una arquitectura y un prototipo de la herramienta SODRA, que sirvió como base para desarrollar una aplicación web y producir los diagramas del modelo del cliente y del modelo navegacional propuestos.

El manejo de los diagramas propuestos se realizó mediante la interacción y el control de nodos, donde la creación del diagrama del cliente fue fácil de entender ya que se asemeja a un mapa conceptual. En contraste, la curva de aprendizaje del diagrama navegacional resultó ser un desafío para los clientes, por lo que se necesita una alternativa que facilite su comprensión. En cuanto a las diferentes herramientas que generan diagramas, como Visual Paradigm, estas gestionan cada etapa del proceso de desarrollo de software, mientras que los editores de diagramas entidad-relación ofrecen una vista gráfica de la estructura de la base de datos. Finalmente, se presenta un avance de la herramienta tanto para el diagrama del cliente como para el navegacional.

El concepto de MDE desempeña un papel importante en el diseño de sistemas en diversos dominios de aplicación, como la automoción, la aviónica o las telecomunicaciones. La problemática que se describió en [21] es la falta de herramientas MDE que tienen como objetivo mejorar simultáneamente la calidad y disminuir los costos mediante una validación temprana al resaltar las fallas del diseño conceptual mucho antes de las fases de prueba tradicionales de acuerdo con el principio de correcto por construcción. Además, mejora la productividad de los ingenieros al sintetizar automáticamente diferentes artefactos de diseño (código fuente, tablas de configuración, casos de prueba, árboles de fallas, entre otros) necesarios para los estándares de certificación.

El caso práctico de [21] presentó un desafío complejo que incluye validación continua de restricciones de buena formación, un escenario de sincronización del modelo, es decir, una transformación de Modelo a Modelo (M2M) del modelo sistemas ciber-físicos inteligentes (CPS) a un modelo de implementación y un escenario de generación de código del modelo de implementación al código Java. Para ello se realizó en tres partes:

- En la versión de VIATRA1 se desarrolló un marco de transformación basado en Prolog, algunas características son la exportación / importación de modelos basados en XMI, transformaciones de modelos mediante

transformación de gráficos, las transformaciones UML como sintaxis visual, código de transformación y el motor de transformación Prolog.

- En cuanto a VIATRA2 constó en un marco de transformación de modelos en Eclipse donde se integró la gestión del modelo, lenguaje de transformación, consulta gráfica y motor de transformación, así como *plugins* de transformación y la transformación de Viatra2 comenzó a servir como núcleo para funciones y complementos de alto nivel.
- Finalmente, en VIATRA3 se desarrolló una plataforma de transformación reactiva basada en consultas incrementales, el uso de lenguaje de consulta de gráficos declarativos de alto nivel, motor de consulta incremental IncQuery que ofrece un motor altamente eficiente para evaluar consultas sobre modelos, y la Integración con herramientas del EMF.

En los últimos 16 años, las tres generaciones del marco de transformación del modelo Viatra han servido continuamente como un medio general para diseñar asignaciones complejas dentro y entre DSL.

El uso de metamodelos también se implementa en ingeniería inversa. Las herramientas de ingeniería inversa a menudo definen sus propios metamodelos de acuerdo con sus propósitos y características previstas. La representación del código (es decir, el metamodelo) depende del problema real de ingeniería inversa y de la técnica de análisis del programa aspirante. Cada herramienta de ingeniería inversa debe elegir el nivel de abstracción apropiado del metamodelo. Para muchas actividades de ingeniería inversa, sólo es necesaria una visión general del sistema. En consecuencia, la cantidad de datos extraídos por analizadores de lenguaje algunas veces es demasiado grande para comprender o analizar en un período de tiempo razonable. El objetivo de [22] es proporcionar una taxonomía integral y utilizar esta taxonomía para clasificar algunos metamodelos populares. La taxonomía, denominada *Program Metamodel Taxonomy* (ProMeTA), y los resultados de la clasificación apoyan la clasificación, comparación, reutilización y extensión de metamodelos del programa y herramientas de ingeniería inversa en varios escenarios de uso.

En el Desarrollo de Software Dirigido por Modelos, los modelos se procesan automáticamente para apoyar la creación, construcción y ejecución de sistemas. Existe una gran variedad de lenguajes de transformación de modelos dedicados, con la promesa de realizar de manera eficiente el tratamiento automatizado de los modelos. Para investigar el beneficio real de la utilización de este tipo de lenguajes de especialidad, se realizó un experimento en [23] a gran escala en la que más de 78 sujetos resolvieron 231 tareas individuales utilizando tres lenguajes. El experimento reveló las similitudes y diferencias entre los lenguajes de transformación de modelos (ATL y QVT-O) y los beneficios de su uso en tareas de desarrollo comunes (la comprensión, el cambio y la creación) en contra de un lenguaje de propósito general moderno (Xtend). El resultado mostró ningún beneficio estadísticamente significativo del uso de un lenguaje de transformación dedicada comparado con un lenguaje de propósito general moderno. Sin embargo, se identificaron que varios aspectos de la programación de transformación en lenguajes de transformación de dominio específico no parecen ayudar, incluyendo objetos de copia de identificación, contexto y acondicionado del cálculo de los tipos. Los idiomas estudiados, declarativos, imperativos, específicos de dominio o de propósito general en [23], ayudaron a los usuarios en tareas complejas, como la copia de partes del modelo. Los lenguajes específicos del dominio apoyaron a los sujetos porque se identificó mejor el punto de partida y el contexto de los cambios en comparación con Xtend (Lenguaje de Propósito General, GPL). Aun así, se identificó un potencial de mejora para todos los lenguajes estudiados. La mayoría de los sujetos tuvieron problemas para crear y cambiar correctamente las condiciones de valor, para inicializar propiedades de valores múltiples y para usar la recursividad. La futura investigación del lenguaje y el diseño de herramientas puede abordar algunos de estos problemas. Es importante destacar que no pudieron confirmar estadísticamente una ventaja anticipada de las DSL de transformación sobre Xtend, o de ATL sobre los idiomas imperativos. Este es un hallazgo preocupante: significa que la migración de un GPL a un lenguaje de transformación dedicado podría no traer beneficios sustanciales, especialmente si el tamaño y el número de transformaciones son pequeños. Al mismo tiempo, los



usuarios expertos de GPL son mucho más fáciles de contratar que los expertos en lenguaje de transformación, por lo que la productividad con un GPL moderno es mayor. Los investigadores de transformación de modelos deben considerar estos resultados, como una indicación de que se necesitan mejoras adicionales en esta tecnología para garantizar fuertes beneficios.

La aplicación de [24] se mencionó que la ingeniería de sistemas basados en modelos (MBSE) se utiliza ampliamente en el diseño de sistemas ciber-físicos complejos (CPS) mediante el uso de diversas herramientas de modelado. Las herramientas de MBSE ayudan a detectar fallas en el diseño temprano, lo que ahorra costos significativos. Además, también permiten la síntesis automatizada y la coevolución coherente de diferentes artefactos de diseño, haciendo que el proceso de desarrollo sea más productivo.

La aplicación de [24] se ha aplicado con éxito en múltiples proyectos de investigación con colaboradores industriales:

- En el proyecto Trans-IMA, el Solucionador VIATRA mostró consistencia, no ambigüedad e integridad del fragmento de DSL al detectar defectos de diseño en la especificación.
- En el proyecto R3-COP ARTEMIS, VIATRA Solver apoyó la prueba de sistemas de robots autónomos y cooperativos con generación de contexto de prueba iterativa.
- VIATRA Solver se utilizó como solucionador de fondo para la técnica de propagación de cambios hacia atrás en la cadena de herramientas de un estudio de caso de atención médica remota en el proyecto Concerto, que desarrolló un entorno para la medición del pulso y la presión arterial controlado por un teléfono inteligente.

VIATRA Solver ofrece garantías únicas de incrementación, integridad y diversidad, También, enumera todos los modelos diferentes con respecto a una clase de equivalencia personalizable y si no se derivan modelos hasta cierto tamaño, la búsqueda se continua desde estas soluciones parciales. Finalmente, el conjunto de modelos derivados debe ser más diverso que otros enfoques que usan

solucionadores lógicos en el *back-end*, por lo tanto, el solucionador es más apropiado para ser utilizado para escenarios de prueba de mutación.

En [25] , se presentó la mejora de la arquitectura subyacente de los sistemas en la nube, se propuso *UML2Cloud*, un marco dirigido a modelar y verificar sistemas en la nube. El núcleo principal de *UML2Cloud* utiliza perfiles UML para capturar los elementos principales de un sistema en la nube, incluidos, entre otros elementos, su arquitectura subyacente y la interacción con los clientes. Además, *UML2Cloud* utiliza técnicas de transformación de modelo a texto para generar automáticamente documentos de configuración que representan escenarios complejos en la nube. En este trabajo, usaron documentos como entrada para una herramienta de simulación en la nube, llamada *Simcan2Cloud*, para simular el comportamiento de diferentes sistemas.

Además, en [25] se mostró cómo una parte de un modelo de nube se transforma en varias líneas de texto que pertenecen a diferentes partes del archivo de configuración del *scenario.ned*. Los valores de propiedad de componentes y estereotipos se extraen a través de las plantillas *Acceleo* implementadas, y luego se escriben en el archivo *scenario.ned*. Para comenzar, la plantilla principal *generateSimcan2CloudFiles* extrae la información del componente principal, que es *CloudInfrastructure*, y llama a las plantillas *generateNEDFile* y *generateINIFile*. La plantilla *generateNEDFile* crea el archivo *scenario.ned*. Esta plantilla extrae los centros de datos a los que hace referencia la infraestructura de la nube y llama a las plantillas *generateScenario* y *generateDataCenter* para completar el archivo.

Por lo tanto, el análisis de los resultados de rendimiento obtenidos de las simulaciones permite obtener conclusiones sobre la mejora de la eficiencia de las aplicaciones en la nube estudiadas mediante el ajuste de la configuración de recursos de hardware.

En [26] se realizó un análisis centrado en el estado de la investigación, el estado de la práctica y el estado del arte en la ingeniería basada en modelos (MDE). El artículo menciona que se reunieron a expertos de la industria, la academia y la comunidad de código abierto para evaluar lo que ha cambiado en la investigación en MDE en

los últimos 10 años, qué desafíos quedan y qué nuevos desafíos han surgido. Este artículo informa sobre los resultados de esas reuniones y presenta un conjunto de grandes desafíos que surgieron de las discusiones y síntesis. Estos desafíos podrían conducir a iniciativas de investigación para la comunidad en el futuro.

La estandarización condujo a la producción de *Meta-Object Facility* (MOF), *Model-Driven Architecture* (MDA), *Object Constraint Language* (OCL), y *Query-View-Transformation* (QVT). Los investigadores en modelado participaron de manera significativa con los esfuerzos de estandarización relevantes, con diversos grados de éxito. En esencia, este período sentó las bases para una investigación más reciente, proporcionando las bases necesarias para una investigación más avanzada sobre herramientas de modelado y gestión de modelos.

Una sugerencia de [26] implica pasar a un MDE específico del dominio. En lugar de considerar el MDE como si fuera un enfoque de propósito general para la ingeniería de sistemas y software, se habló sobre “MDE para la banca”, “MDE para el seguro”, “MDE para la salud”, entre otros. Cada dominio requiere diferentes soluciones, cada uno en un enfoque actual de proponer diferentes lenguajes específicos de dominio para cada sector. El MDE específico del dominio podría involucrar, por ejemplo, herramientas diseñadas específicamente para diferentes partes interesadas en términos que entiendan (lo que puede tener una mayor probabilidad de ser adoptado).

## 2.2 Análisis comparativo

En la Tabla 2.1 se presenta un estudio comparativo de ocho artículos relacionados con el tema abordado en esta investigación. Se evaluaron aspectos clave como el problema abordado, la contribución realizada, las tecnologías utilizadas, los resultados obtenidos y el estado actual de la investigación en cada caso.

**Tabla 2.2 Análisis comparativo de los artículos relacionados.**

Artículo	Problema	Contribución	Tecnologías	Resultado	Estado
Mengerink et al. [20]	Deficiencia de expresividad entre lenguajes específicos de dominio en metadatos a falta de lenguajes de restricción de dominios y conceptos.	Conjunto de datos disponible públicamente de expresiones OCL derivadas de GitHub. El conjunto de datos se compone de dos colecciones de expresiones OCL. El conjunto de datos incluye los archivos originales .ocl, .ecore y .mtl, así como los	QVT, GitHub	Conjunto de datos que permite una variedad de estudios empíricos de la OCL, incluidos estudios de uso y evaluaciones prácticas de las técnicas propuestas.	En mejoras

Artículo	Problema	Contribución	Tecnologías	Resultado	Estado
		archivos AST generados para el conjunto de datos OCL / Ecore. Los archivos AST se almacenan en formato XMI conforme al OCL Pivot Meta Model.			
Carreón-Díaz [1]	Construir aplicaciones web es un proceso que demanda una cantidad significativa de tiempo y esfuerzo para ser realizado de manera adecuada.	Se propuso la creación de una arquitectura y un modelo de muestra para la herramienta SODRA, los cuales sirvieron como fundamento para el desarrollo de una aplicación Web y posibilitaron la	SODRA, Java y jQuery	Se presentó un avance de la herramienta tanto para el diagrama del cliente como el navegacional.	En mejoras

Artículo	Problema	Contribución	Tecnologías	Resultado	Estado
		generación de los diagramas correspondientes al modelo del cliente y navegacional.			
Varró et al. [21]	<p>Se encuentran los siguientes problemas:</p> <ul style="list-style-type: none"> <li>• Falta de herramientas MDA</li> <li>• Mejorar simultáneamente la calidad</li> <li>• Disminuir los costos mediante validación temprana</li> <li>• Resaltar las fallas del diseño conceptual</li> <li>• Mejorar la productividad de los ingenieros.</li> </ul>	<p>Desarrollo de una plataforma de transformación de modelos reactiva, impulsada por eventos construida sobre consultas gráficas incrementales donde las transformaciones se ejecutan continuamente como reacciones a cambios en el modelo subyacente.</p>	<p>Rational Rose, Prolog, UML2Prolog, native Prolog, Ad hoc, Eclipse, GT + ASM, Relational BD, EJB, Xtext, Java, Xtend + IncQuery, complex text suite.</p>	<p>Tres generaciones del marco de transformación del modelo Viatra.</p>	<p>En mejoras</p>

<b>Artículo</b>	<b>Problema</b>	<b>Contribución</b>	<b>Tecnologías</b>	<b>Resultado</b>	<b>Estado</b>
Washizaki et al. [22]	Los profesionales e investigadores deben clasificar, comparar, reutilizar y ampliar los metamodelos del programa y las herramientas de ingeniería inversa correspondientes de acuerdo con sus objetivos, algunos trabajos deben ser evaluados y comparados los metamodelos y herramientas.	Desarrollo de un marco conceptual con definiciones de metamodelos del programa y conceptos relacionados, así como también una taxonomía integral ProMeTA basada en este marco.	Java, C++, árbol de sintaxis abstracta (AST), árbol de sintaxis concreta (CST)	Validación de la taxonomía en términos de su ortogonalidad y utilidad a través de la clasificación de cinco metamodelos populares de la encuesta. ProMeTA está a disposición de la comunidad de ingeniería inversa, incluidos profesionales e investigadores, a través del sitio web oficial.	Terminado
Hebig et al. [23]	En el desarrollo de software basado en modelos (MDS), los modelos se	Experimento a gran escala en la que más de 78 sujetos	ATL, QVT-0, Xtend, C#, java	Es importante destacar que no se confirmó	Terminado

Artículo	Problema	Contribución	Tecnologías	Resultado	Estado
	<p>procesan automáticamente para admitir la creación, construcción y ejecución de sistemas. Las transformaciones se utilizan, entre otras razones, para calcular vistas en modelos, para validar modelos, para refactorizar modelos, para interpretar o ejecutar modelos, para implementar depuradores de modelos, para transformar modelos a modelos de nivel inferior, e implementar operaciones de gestión de modelos como el versionado.</p>	<p>resuelven 231 tareas individuales utilizando tres lenguajes: ATL, QVT-O y Xtend.</p>		<p>estadísticamente una ventaja anticipada de las DSL de transformación sobre Xtend, o de ATL sobre los idiomas imperativos. Este es un hallazgo preocupante: significa que la migración de una GPL a un lenguaje de transformación dedicado podría no traer beneficios sustanciales, especialmente si el tamaño y el número de transformaciones son pequeños (los</p>	



Artículo	Problema	Contribución	Tecnologías	Resultado	Estado
				beneficios observados en el experimento no son significativos, con tamaños de efectos pequeños).	
Semeráth et al. [24]	La falta de técnicas de prueba sistemáticas entre dominios bien fundamentados y escalables para herramientas de modelado. Los casos de prueba de las herramientas de modelado de CPS están altamente basados en datos. Toman la forma de modelos de gráficos complejos que necesitan satisfacer varias restricciones estructurales.	VIATRA Solver es un complemento integrado en el IDE de Eclipse o como una herramienta de línea de comandos independiente.	Xtext. IDE Eclipse	VIATRA Solver se ha aplicado con éxito en múltiples proyectos de investigación con colaboradores industriales. <ul style="list-style-type: none"> <li>• Trans-IMA</li> <li>• R3-COP ARTEMIS,</li> <li>• Concerto</li> </ul>	Terminado

Artículo	Problema	Contribución	Tecnologías	Resultado	Estado
	Los modelos de prueba contruidos manualmente por ingenieros son ineficaces para encontrar errores en esas herramientas.				
Bernal et al. [25]	El objetivo es mejorar la arquitectura subyacente de los sistemas en la nube y capturar los elementos principales de un sistema en la nube, incluidos, entre otros elementos, su arquitectura subyacente y la interacción con los clientes.	Plataforma que utiliza técnicas de transformación de modelo a texto para generar automáticamente documentos de configuración que representan escenarios complejos en la nube.	UML2Cloud, UML, Acceleo	El análisis de los resultados de rendimiento obtenidos de las simulaciones permitió sacar conclusiones para mejorar la eficiencia de las aplicaciones en la nube.	
Bucchiarone et al. [26]	Los problemas clave que se identificaron son los siguientes:	Presentación de los grandes desafíos en el campo de la	ATL, QVT, ETL, Henshin, VIATRA y	Clasificación de diferentes categorías tratando también de	Terminado

Artículo	Problema	Contribución	Tecnologías	Resultado	Estado
	<ul style="list-style-type: none"> <li>• Principios, prácticas y patrones de ingeniería del lenguaje</li> <li>• Bancos de trabajo de idiomas</li> <li>• Gestión de modelos</li> <li>• Análisis de modelos</li> <li>• Modelos en tiempo de ejecución, el uso de modelos para administrar y comprender los sistemas</li> <li>• Repositorios de modelado</li> <li>• La escalabilidad en diferentes dimensiones</li> </ul>	<p>ingeniería basada en modelos de acuerdo con los participantes expertos en los dos eventos que organizaron para discutir el futuro de MDE.</p>	<p>Stratego</p>	<p>ordenarlos con respecto a su relevancia cronológica además de la contribución a estimular a investigadores, profesionales y desarrolladores de herramientas para abordar y explorar algunos de ellos. Al mismo tiempo, proporciona un contexto útil para futuros proyectos de investigación, propuestas de becas de investigación y nuevas direcciones</p>	

<b>Artículo</b>	<b>Problema</b>	<b>Contribución</b>	<b>Tecnologías</b>	<b>Resultado</b>	<b>Estado</b>
				de investigación.	

La implementación de sistemas basados en modelos tiene como objetivo principal mejorar la eficiencia y reducir los errores en los procesos de desarrollo de software, incluyendo las pruebas unitarias e integración. Los trabajos previos analizados describen cómo se enfrentaron a esta problemática, las soluciones propuestas y los resultados esperados. En general, estos trabajos actúan como intermediarios entre diferentes tecnologías, traduciendo la información de una tecnología a otra con mayor precisión y utilidad para facilitar la tarea del programador y ahorrar tiempo.

El trabajo actual se basa en la herramienta SODRA, que tiene la capacidad de crear diagramas de navegación y cliente y generar un archivo XMI. Este archivo es utilizado en la fase actual del proyecto para interpretar los diagramas y crear diagramas *Meta-Object Facility*, que es un estándar en ingeniería dirigida por modelos establecido por el *Object Management Group*. La implementación de esta nueva fase permitirá mejorar significativamente la capacidad de SODRA, haciéndola más completa y robusta para su uso en diferentes tareas.

### 2.3 Propuesta de solución

Con el objetivo de encontrar el enfoque de trabajo más adecuado para la solución propuesta en el proyecto de tesis, se realizó una investigación y análisis exhaustivo de las herramientas de reconocimiento de lenguajes, IDEs y metodologías de desarrollo de software. Tras considerar las características, documentación, plataformas, costos de licencia, ventajas y desventajas de las diferentes Tecnologías de Información investigadas, se presentó en la Tabla 2.2 la alternativa de solución seleccionada para el desarrollo del proyecto de tesis actual.

**Tabla 2.3 Alternativa de solución.**

<b>Aspecto</b>	<b>Propuesta</b>
Herramienta de reconocimiento de lenguaje	ANTLR4
IDE	NetBeans
Metodología de desarrollo	Modelo espiral

Es importante considerar la propuesta de la Tabla 2.2 la herramienta de reconocimiento de lenguaje utilizada es ANTRL4. Se trata de una herramienta para

crear aplicaciones de lenguajes, su versión actual es la 4.9, Los archivos de entrada archivos de texto (extensión .g4) y devuelve, dependiendo de los parámetros, código para analizadores léxicos y gramaticales, armado de ASTs así como el recorrido de éstos (Código de salida en Java por omisión). A diferencia de otras herramientas, usa el mismo tipo de sintaxis para analizadores léxicos y gramaticales. ANTLR4 separa la gramática del lenguaje fuente de las acciones de recorrido de AST y validaciones semánticas. La herramienta en cuestión cuenta con el algoritmo *ALL(\*)*, el cual utiliza gramáticas libres de contexto para reconocer *tokens*, lo que le permite reconocer elementos sin contexto como los comentarios anidados. Además, el algoritmo tiene la capacidad de introducir o extraer *tokens* según el contexto semántico, lo que lo hace útil para el análisis léxico en casos donde es necesario un análisis más detallado, como se menciona en el artículo [29]. Cabe destacar que esta herramienta también puede utilizarse para el análisis léxico sin la necesidad de un escáner externo debido a su capacidad de reconocimiento.

ANTLR4 es una herramienta desarrollada en el lenguaje de programación Java, la cual se puede utilizar mediante plugins para distintos entornos de desarrollo integrados (IDEs). A pesar de que su integración con NetBeans presenta algunos errores, es posible ejecutarla mediante la línea de comandos y posteriormente incorporar el código generado en lenguaje Java al proyecto en NetBeans. Cabe destacar que NetBeans es un IDE oficial para el desarrollo en Java 8 y su uso es gratuito, ya que cuenta con las licencias CDDL y GNU. A pesar de que el IDE Eclipse es gratuito gracias a su licencia de código abierto EPL utilizada por la Fundación Eclipse, no incluye complementos y módulos integrados en su instalación, lo que implica que el desarrollador debe configurar el ambiente de desarrollo. Aunque el uso del metamodelo MOF es compatible con EMF, no es una exigencia para el proyecto ya que el objetivo es generar el metamodelo, por lo que NetBeans, que ya incluye el JRE, es una opción más conveniente.

La estrategia de desarrollo del proyecto se basa en el modelo en espiral, que combina elementos del modelo lineal y el modelo iterativo. Este modelo es muy solicitado en proyectos donde los riesgos de fallas son altos, ya que su principal contribución es la gestión de dichos riesgos. El modelo en espiral se compone de

ciclos de crecimiento, cada uno de los cuales sigue cuatro fases en forma de espiral. Cada ciclo sigue estas fases definidas y permite una mayor complejidad con cada iteración. Las ventajas de este modelo incluyen su capacidad de evolucionar a medida que el software se desarrolla, la posibilidad de que el desarrollador reaccione a los riesgos en cada fase del proceso y la posibilidad de utilizar un enfoque de construcción de prototipos en cualquier momento del proceso evolutivo.

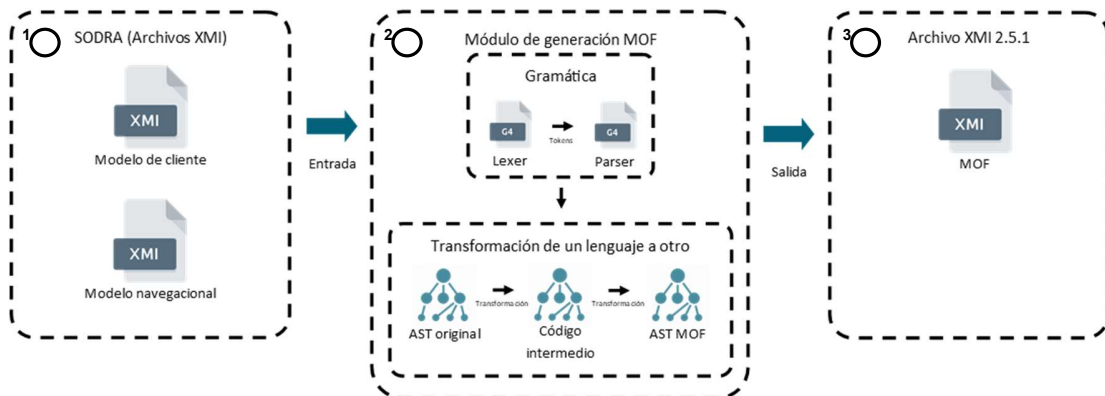
### Capítulo 3. Aplicación de la metodología

En este capítulo se describe la solución propuesta y se detalla el seguimiento de la metodología de desarrollo elegida para crear la aplicación. Se definen los requisitos del usuario y del sistema, así como las limitaciones del proyecto. También se especifica la arquitectura de las aplicaciones que se generarán en cada una de las combinaciones de tecnologías soportadas, y se destacan las distintas opciones de distribución de contenido disponibles para el usuario. Además, se proporciona una explicación detallada de todos los componentes que conforman la arquitectura del módulo generador.

#### 3.1 Descripción de la solución

SODRA proporciona la generación de proyectos basados en modelos, donde se comienza con la esquematización de los modelos de cliente y navegacional, brindando herramientas y elementos definidos para su interpretación. Partiendo del concepto y análisis, SODRA permite la exportación de ese modelo en una especificación para el intercambio de diagramas XMI.

En la Figura 3.1 se muestra el esquema de solución establecido en este proyecto de tesis, y se proporciona una explicación breve de cada uno de sus componentes.



**Figura 3.1 Esquema de la solución propuesta**

1. Como elementos de entrada, la herramienta espera modelos de cliente y navegacional (sólo diagramas exportados en SODRA) que representan el



metamodelo fuente simplificado basado en paquetes. Estas clases contienen propiedades tipificadas y se caracterizan por multiplicidades (superior e inferior). Los diagramas contienen operaciones con parámetros escritos.

2. La solución emplea una herramienta para el reconocimiento de lenguajes y gramáticas predefinidas para realizar la generación de modelo MOF que sea ejecutado las diferentes herramientas de generación de aplicaciones o modelos, correspondiente a un metamodelo con la conceptualización realizada por el usuario y que además cumpla con las especificaciones representadas en los modelos recibidos como entrada y con las directrices indicadas para la salida.
3. El objetivo es un metamodelo dimensional simplificado basado en paquetes que contienen hechos y dimensiones.

## **3.2 Metodología de desarrollo en Espiral**

Con el objetivo de abordar el problema que se plantea en este trabajo de tesis y alcanzar los objetivos establecidos de manera efectiva, se describen los ciclos que se llevaron a cabo para construir el módulo de generación de metamodelo MOF. Estos ciclos se basaron en los elementos del modelo de desarrollo en espiral que se definen en la fuente citada [30].

### **3.2.1 Primera iteración**

Durante el primer ciclo, se llevó a cabo la etapa inicial del desarrollo, que involucra la creación y análisis semántico de los archivos de entrada. Este proceso se basó en las directrices establecidas en la primera fase del modelo en espiral, la cual incluye la definición de los requisitos tanto del usuario como del sistema, junto con las limitaciones que afectan al proyecto.

#### **3.2.1.1 Análisis de requerimientos**

El proceso de obtener y analizar los requisitos implica una serie de actividades iterativas que se retroalimentan entre sí. Durante este proceso se identificaron dos grupos principales: uno enfocado en la descripción de los servicios y restricciones, y otro en los requerimientos del sistema, que detallan sus funciones, servicios y restricciones operativas. El primer grupo corresponde a los requisitos del usuario,

donde se establecen las declaraciones sobre lo que el sistema debe proporcionar y las limitaciones bajo las cuales debe operar. En la Tabla 3.1 se encuentran enlistados todos los requerimientos del usuario.

**Tabla 3.1 Requerimiento de usuario**

<b>Número</b>	<b>Requerimiento del usuario</b>
1	El módulo debe permitir al usuario la selección de la importación del modelo cliente y modelo navegacional de la herramienta SODRA en modelo MOF a generar.
2	El módulo debe permitir la selección de una combinación entre los modelos cliente y navegacional para el modelo MOF a generar.
3	La arquitectura del metamodelo MOF a generar debe estar basada en el estándar XMI 2.5.1.
4	El módulo debe generar un modelo MOF basado en estándar XMI de acuerdo con las selecciones realizadas en los puntos anteriores.

Las Tablas 3.2 a 3.5 detallan los requerimientos del sistema en relación a cada especificación del usuario.

**Tabla 3.2 Requerimientos del sistema relativos al requerimiento de usuario 1**

<b>Requerimiento del usuario</b>	
1	El sistema debe permitir al usuario la selección de la importación del modelo cliente y modelo navegacional de la herramienta SODRA en modelo MOF a generar.
<b>Requerimientos del sistema</b>	
1.1	La herramienta SODRA debe contar con una opción para que se seleccionen la importación del modelo MOF.
1.2	La aplicación debe incluir una validación del contenido del archivo XMI de los modelos de cliente y navegacional.

**Tabla 3.3 Requerimientos del sistema relativos al requerimiento de usuario 2**

<b>Requerimiento del usuario</b>	
2	El módulo debe permitir la selección de una combinación entre los modelos cliente y navegacional para el modelo MOF a generar.
<b>Requerimientos del sistema</b>	
2.1	La herramienta SODRA debe contar con una interfaz para que el usuario seleccione la combinación de los modelos cliente y navegacional.
2.2	El módulo debe incluir una validación del contenido del archivo XMI y que hayan sido creados en el proyecto los modelos de cliente y navegacional.

**Tabla 3.4 Requerimientos del sistema relativos al requerimiento de usuario 4**

<b>Requerimiento del usuario</b>	
3	La arquitectura del metamodelo MOF a generar debe estar basada en el estándar XMI 2.5.1.
<b>Requerimientos del sistema</b>	
3.1	El módulo, al momento de la generación modelo MOF, debe seguir la estructura definida por la OMG en el estándar XMI 2.5.1 para compatibilidad en otras herramientas.

**Tabla 3.5 Requerimientos del sistema relativos al requerimiento de usuario 4**

<b>Requerimiento del usuario</b>	
4	El módulo debe generar un modelo MOF basado en estándar XMI de acuerdo con las selecciones realizadas en los puntos anteriores.
<b>Requerimientos del sistema</b>	
4.1	El módulo debe llevar a cabo un proceso integral que incluya la lectura, procesamiento y validación de los archivos de entrada.
4.2	Es necesario que el módulo genere una representación intermedia utilizando los resultados obtenidos del proceso de análisis.
4.3	El módulo debe crear el modelo final requerido a partir de la representación intermedia, siguiendo las directrices establecidas por el usuario.

4.4	Si el proceso de generación del modelo MOF experimenta algún problema, el módulo debe notificar al usuario de inmediato.
-----	--------------------------------------------------------------------------------------------------------------------------

### 3.2.1.2 Restricciones

Las restricciones establecidas para el desarrollo del módulo que afectan su desarrollo son:

- A. La representación XMI del modelo de cliente debe ser coherente con el modelo navegacional indicado en el archivo XMI.
- B. La representación XMI de los modelos de cliente y navegacional debe ser específica de la herramienta SODRA.
- C. Debido a la falta de especificación en los modelos fuente, se asignarán valores predeterminados al modelo generado en cuanto a tipo de dato, accesibilidad, cardinalidad y relaciones. Ejemplo:
  - Tipo de dato: *String* y *dataType* según corresponda.
  - Accesibilidad: *private* en los atributos y *public* en los métodos.
  - Cardinalidad: 1:1.
  - Relaciones: asociación, agregación y composición.

Es necesario que los modelos estén equipados con datos predefinidos para permitir la visualización de los elementos generados en otras herramientas compatibles.

- D. Aunque actualmente sólo se cuenta con la combinación de los modelos de cliente y navegacional, la aplicación debe estar diseñada de manera que pueda agregar módulos en el futuro para generar otras combinaciones de modelos destino.
- E. La herramienta de generación debe estar diseñada en módulos que permitan la generación de características particulares, como la distribución de elementos dentro de la distribución general.
- F. El modelo MOF destino utilizado debe ser la versión 2.0 bajo el estándar XMI 2.5.1.

G. La versión requerida de los modelos tomados como archivos fuente es la versión del exportador de archivos XMI de SODRA como archivo XMI 2.1 con notación propia de SODRA.

**Tabla 3.4 Análisis de los principales riesgos**

<b>Número</b>	<b>Riesgo</b>	<b>% de aparición</b>	<b>Impacto</b>
<b>1</b>	La herramienta SODRA deja de proporcionar la representación XMI del modelo de cliente y navegacional.	Muy bajo	Catastrófico
<b>2</b>	La herramienta SODRA modifica la representación XMI del modelo de cliente y navegacional.	Muy bajo	Catastrófico
<b>3</b>	Complejidad en el uso de ANTLR.	Tolerable	Serio
<b>4</b>	Cambios en la estructura del estándar XMI 2.5.1.	Bajo	Serio

#### 3.2.1.2.1 Evaluación de riesgos

Prevenir eventos adversos es uno de los objetivos de la evaluación de riesgos, y es fundamental para garantizar la culminación exitosa de un proyecto. Por esta razón, la identificación de riesgos y la implementación de acciones para minimizar sus efectos son partes importantes del proceso de desarrollo en espiral [30]. En el siguiente análisis, se han establecido métricas para evaluar la probabilidad de aparición de un riesgo, las cuales son:

- Muy bajo: Porcentaje de aparición menor del 10%.
- Bajo: Porcentaje de aparición desde el 10% hasta no mayor al 25 %.
- Moderado: Porcentaje de aparición entre el 25% y 50%.
- Alto: Porcentaje de aparición mayor del 50% y no superior al 75%.
- Muy alto: Porcentaje de aparición mayor al 75%.

Del mismo modo, durante la evaluación de riesgos se definen los criterios para medir el impacto que cada riesgo puede tener, los cuales pueden ser catalogados como catastróficos, serios, tolerables o insignificantes [30]. La Tabla 3.6 muestra los principales riesgos identificados y su probabilidad de ocurrencia e impacto.

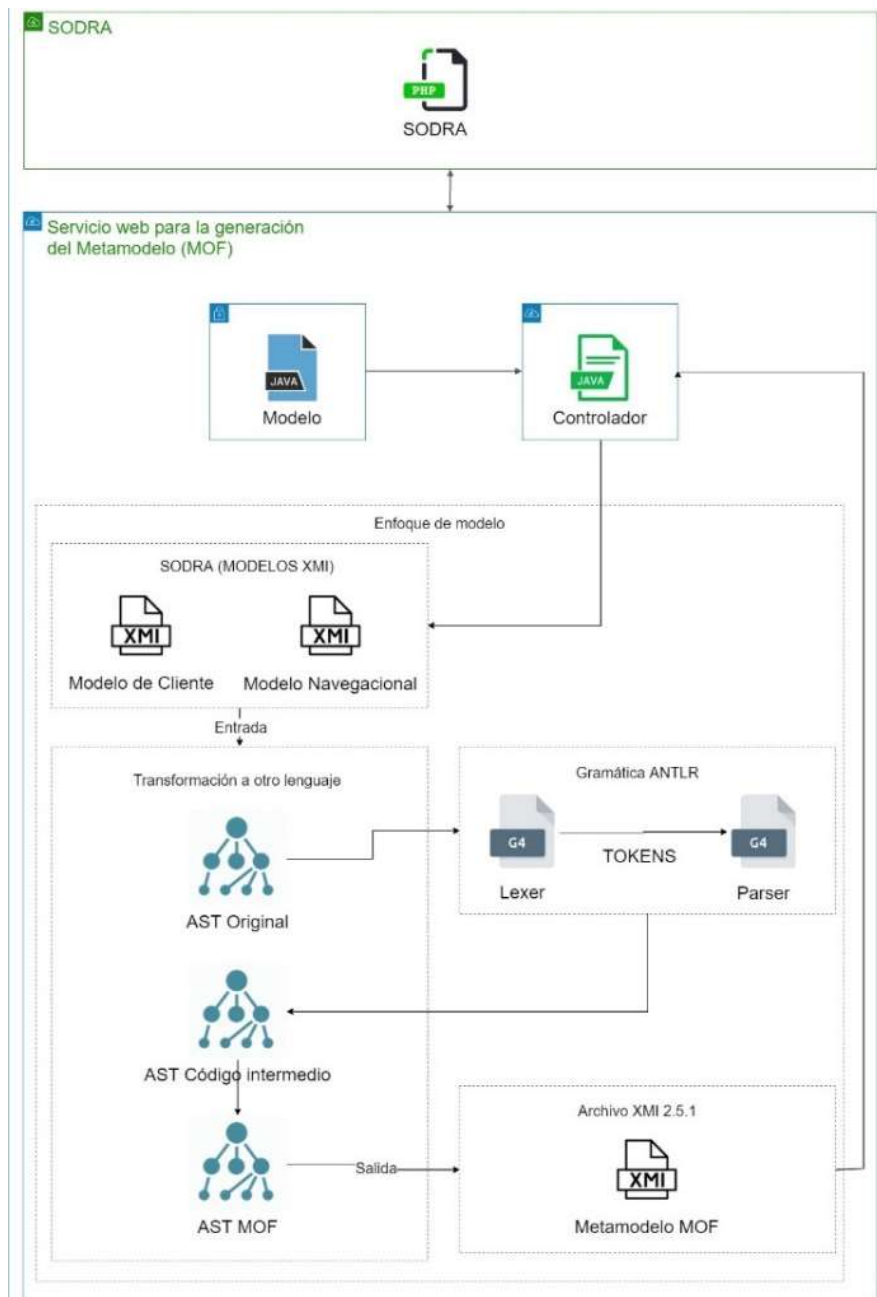
Para abordar la posible aparición de los riesgos considerados, se estableció una estrategia de gestión de riesgos que se detalla en la Tabla 3.7.

**Tabla 3.7 Estrategia para la gestión del riesgo**

Numero de riesgo	Estrategia para la gestión del riesgo
1	Para encontrar el problema en la representación XMI del modelo de cliente y navegacional, es necesario examinar el código fuente detalladamente.
2	Modificar la gramática para el reconocimiento del XMI, que representa el modelo de cliente y navegacional. Estos cambios, en la representación de los modelos de la herramienta SODRA, no la dejan inutilizable ya que son relevantes para la generación del modelo y serán integrados.
3	Documentación disponible para uso y manejo de ANTLR 4 [17].
4	La versión está sujeta a cambios o será reemplazada por un inventario más reciente, pero su uso local continuará.

### 3.2.1.3 Desarrollo y validación

Se describe cómo se realizó el desarrollo de la primera etapa del módulo de generador de metamodelo MOF bajo el estándar XMI 2.5.1.



**Figura 3.8 Arquitectura del sistema**

#### 3.2.1.4 Arquitectura del generador

El desarrollo de este módulo considera el uso de compiladores con sus principios, técnicas y herramientas pero no se trata de un traductor que siga el proceso tradicional que V. Aho propone. También el enfoque es orientado según Parr que lo define como una aplicación de lenguaje. En este contexto, se hace referencia al uso selectivo de ciertas partes del proceso de traducción, sin que se trate de la

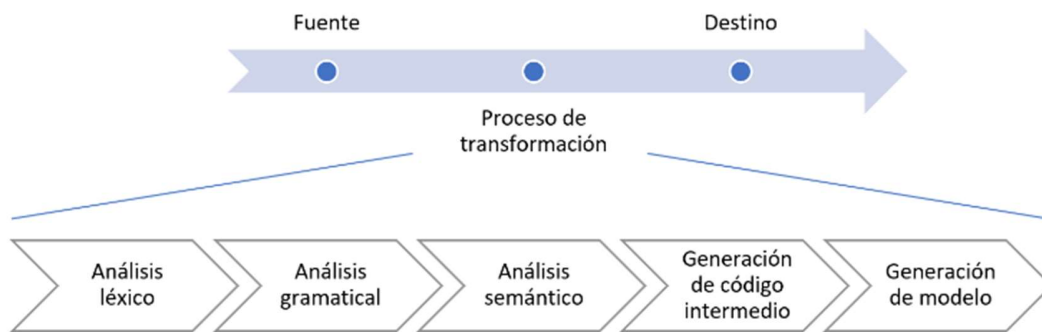
creación de un compilador para un lenguaje de alto nivel. Además, se menciona que se empleará la herramienta ANTLR4 para la construcción de dicho sistema . En la Figura 3.2 se observan los componentes existentes en el proyecto y los que se agregan a SODRA.

En Servicio para la generación de Metamodelo (MOF) se encuentra:

- El Archivo XMI es el modelo fuente, un modelo Navegacional y otro modelo de cliente simplificado basado en paquetes. Estas clases contienen propiedades tipificadas y se caracterizan por multiplicidades (superior e inferior) [27].
- La conversión a otro lenguaje implica la utilización del AST, que es la estructura primaria de los modelos XMI originales. Para llevar a cabo este proceso, se utiliza la herramienta ANTLR, que es capaz de leer, procesar, ejecutar y traducir archivos de texto estructurados o binarios [17]. Se definió la gramática (Lexer.g4) con los tokens identificados en el XMI de SODRA y el léxico (Parser.g4) donde se determinó la estructura y reglas que debe cumplir el modelo. A partir de estos archivos se obtienen clases Java que apoyan en la integración a aplicaciones con código autogenerado con métodos ya definidos para la transformación a AST de código intermedio.
- El *Metamodelo MOF2.0* es un metamodelo dimensional simplificado basado en paquetes que contienen hechos y dimensiones [13].
- El *Servicio WEB* implementado en el transformador de modelos y el sistema SODRA posibilita el intercambio de datos a través de la red, lo que permite la interoperabilidad entre ellos. Para lograr esto, la información a transmitir se codifica y se establece un proceso que se basa en la solicitud de servicio por parte del sistema SODRA y la respuesta correspondiente por parte del proveedor del servicio, es decir, el transformador de modelos.

Con respecto a SODRA se incluyeron el enlace de comunicación con el servicio WEB y los artefactos que genera SODRA (XMI), y se muestra el resultado de la transformación cuando este esté disponible por parte del servicio de generación de metamodelo (MOF).





**Figura 3.9 Proceso de Compilación/Traducción**

### 3.2.1.5 Proceso de transformación

La figura 3.3 muestra la explicación del proceso del compilador/traductor para la transformación de modelo. Este proceso se descompone en varias partes, incluyendo un programa fuente que consta de piezas componentes como el análisis léxico, el análisis gramatical y el análisis semántico. Estas piezas componentes trabajan juntas para producir una representación interna conocida como código intermedio. Posteriormente, en la fase de síntesis, el código intermedio es traducido al modelo destino [31].

```

xmi:xmi ..
  @xmlns:sodra: https://sodra.gov.uk/metadata
  @xmlns:xmi: http://schema.omg.org/spec/XMI/2.1
  @xmlns:xsd: http://www.w3c.org/2001/XMLSchema
  @xmi:version: 2.1
  xmi:properties ..
  sodra:model ..
  sodra:diagram ..

```

**Figura 3.10 Estructura general del modelo XMI**

### 3.2.1.6 Fuente

Se realizó una evaluación detallada de los dos archivos fuente requeridos por el generador de metamodelo MOF, los cuales son generados por la herramienta SODRA en formato XMI y contienen información sobre el modelo de cliente y el modelo navegacional del proyecto a generar. El propósito de este análisis fue identificar de forma exhaustiva los elementos de entrada pertinentes para la generación del metamodelo MOF.

En ambos archivos XMI, se emplea la versión de XML 1.0 con codificación UTF-8 y la versión 2.1 de XMI. Estos archivos cuentan con dos espacios de trabajo distintos: el espacio de trabajo *xmlns:sodra*, que se utiliza para la definición del meta-modelo de los nodos de los diagramas del cliente y navegacionales, así como sus relaciones correspondientes; y el espacio de trabajo *xmlns:xmi*, que pertenece al estándar del W3C y se utiliza para definir los tipos de datos, atributos, elementos, elecciones y la definición de tipos simples y complejos [32].

En la figura 3.4 se presenta el código XMI que contiene la definición de los espacios de trabajo y las versiones utilizadas. Se pueden identificar tres grupos principales en este código: *xmi:properties*, *sodra:model* y *sodra:diagram*.

```
<xmi:Properties exporter="Sodra">
  <xmi:Extension extender="Sodra">
    <xsd:element name="lastEdit"
      type="xsd:integer"/>
    <xsd:element name="grid" type="xsd:boolean"/>
    <xsd:element name="autosave"
      type="xsd:boolean"/>
    <xsd:element name="rule" type="xsd:boolean"/>
    <xsd:element name="fileopen"
      type="xsd:boolean"/>
  </xmi:Extension>
</xmi:Properties>
```

### **Figura 3.11 Estructura del nodo *xmi:properties* del modelo XMI**

Se identificaron algunas características comunes en la definición del *xmi:properties*, tales como las propiedades del modelo, como la última modificación, la aplicación de cuadrículas, el guardado automático, las reglas y el archivo abierto. Estos elementos se pueden observar en la figura 3.5. La funcionalidades de cada tipo simple dentro del tipo complejo de *xmi:properties* son:

- **lastEdit:** Indica la última modificación del archivo.
- **Grid:** Muestra la cuadrícula en la herramienta.
- **Autosave:** Activa el autoguardado del diagrama del cliente o navegacional en la herramienta.
- **Rule:** Gestiona la visibilidad de las reglas vertical y horizontal de la herramienta.

- **Fileopen:** Determina si el archivo se encuentra abierto con el propósito de evitar que se modifique el diagrama en dos pestañas de un mismo visualizador o de diferentes visualizadores.



**Figura 3.12 Estructura del nodo sodra:model del modelo XMI**

Para la definición del *sodra:model*, existen diferencias entre el diagrama del cliente y navegacional, al analizar el código XMI, los cuales se observan en la Figura 3.6, se encontraron las funcionalidades de once tipos simples dentro del tipo complejo en función de los nombres:

- **idobj:** Define un identificador único del nodo.
- **x:** Establece la coordenada X del nodo.
- **y:** Muestra la coordenada Y del nodo.

- **isnodo**: Determina si se trata de un nodo del cliente o navegacional.
- **extender**: Define al nodo como sucesor de otro.
- **action**: Establece el nombre del nodo, en el diagrama del cliente considera tres elementos importantes y le antecede el prefijo *cte\_nodo\_*:
  - *cte\_nodo\_concepto*: Representa un nodo de concepto que se muestra en la figura 3.7



**Figura 3.13** *Nodo de concepto*

- *cte\_nodo\_relacion*: Representa un nodo de relación que se muestra en la Figura 3.8



**Figura 3.14** *Nodo de relación*

- *cte\_nodo\_nota*: Representa un nodo de nota que se muestra en la Figura 3.9



**Figura 3.15** *Nodo de nota*

El diagrama navegacional considera a ocho elementos y le antecede el prefijo *nav\_nodo*:

- *nav\_nodo\_inicio*: Representa un nodo de inicio que se muestra en la Figura 3.10



**Figura 3.16 Nodo de lista**

- nav\_nodo\_acceso: Representa un nodo de acceso que se muestra en la Figura 3.11



**Figura 3.17 Nodo de proceso**

- nav\_nodo\_navegacional: Representa un nodo navegacional que se muestra en la Figura 3.12



**Figura 3.18 Nodo navegacional**

- nav\_nodo\_menu: Representa un nodo de menú que se muestra en la Figura 3.13



**Figura 3.19 Nodo de menú**

- nav\_nodo\_consulta: Representa un nodo de consulta que se muestra en la Figura 3.14



**Figura 3.20 Nodo de consulta**

- `nav_nodo_lista`: Representa un nodo de lista que se muestra en la Figura 3.15



**Figura 3.21 Nodo de lista**

- `nav_nodo_proceso`: Representa un nodo de proceso que se muestra en la Figura 3.16



**Figura 3.22 Nodo de proceso**

- `nav_nodo_nota`: Representa un nodo de nota que se muestra en la Figura 3.17



**Figura 3.23 Nodo de nota**

- **form**: Muestra la forma gráfica del nodo (square/circle).
- **in**: Determina el identificador de los nodos del cual es destino.
- **out**: Define el identificador de los nodos del cual es origen.

- **name:** Establece el nombre del nodo.
- **locked:** Determina si un nodo se permite editar o mover

```

<sodraModel name="nodo_cliente/nodo_navegacional"
id="nodo_cliente/nodo_navegacional">
  <xsd:complexType name="nodo_cliente/nodo_navegacional">
    <xsd:choice minOccurs="1" maxOccurs="1">
      <xsd:element name="idobj" type="xsd:integer" />
      <xsd:element name="x" type="xsd:integer" />
      <xsd:element name="y" type="xsd:integer" />
      <xsd:element name="isnodo" type="xsd:integer" />
      <xsd:element name="extender" type="xsd:integer" />
      <xsd:element name="action" type="xsd:string" />
      <xsd:element name="form" type="xsd:string" />
      <xsd:element name="in" type="xsd:string" />
      <xsd:element name="out" type="xsd:string" />
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="locked" type="xsd:boolean" />
    </xsd:choice>
    <xsd:attribute name="idobj" type="xsd:integer" />
    <xsd:attribute name="x" type="xsd:integer" />
    <xsd:attribute name="y" type="xsd:integer" />
    <xsd:attribute name="isnodo" type="xsd:integer" />
    <xsd:attribute name="extender" type="xsd:integer" />
    <xsd:attribute name="action" type="xsd:string" />
    <xsd:attribute name="form" type="xsd:string" />
    <xsd:attribute name="in" type="xsd:string" />
    <xsd:attribute name="out" type="xsd:string" />
    <xsd:attribute name="name" type="xsd:string" />
    <xsd:attribute name="locked" type="xsd:boolean" />
  </xsd:complexType>
  <xsd:element name="nodo_cliente/nodo_navegacional"
type="cliente/navegacional:nodo_cliente/nodo_navegacional" />
</sodraModel>

```

**Figura 3.24 Estructura detallada del nodo sodra:model del modelo XMI**

La parte llamada "sodra:model", tal como se explicó antes, incluye las definiciones de los nodos. Este concepto se muestra con más detalles en la Figura 3.18, donde se puede ver que para cada tipo de entidad compleja, hay una combinación de tipos de entidad simples, cada uno con su propio nombre y tipo de datos correspondiente.



**Figura 3.25 Estructura del nodo sodra:diagram del modelo XMI navegacional a la derecha y modelo XMI de cliente a la izquierda**

Respecto a la sección "sodra:diagram", aquí se incluyen los nodos relevantes para el diagramado, que permiten saber qué se debe obtener como resultado. La Figura 3.19 muestra que, para cada tipo complejo "sodra:diagramelement", se proporciona información del elemento como su tipo, asunto y xmi:id, así como también se



muestran los tipos simples con información relevante para la generación del metamodelo.

Entre los tipos simples mencionados, se encuentran:

- "Idobj": Identificador único de cada nodo, que sirve para hacer referencia a él en relación con otros nodos.
- "Action": Indica el tipo de nodo.
- "In": Indica desde dónde proviene el nodo actual a través del idobj del objeto padre. Es decir, muestra de qué nodo depende. Si "In" está vacío, significa que el elemento es el primer nodo en el diagrama.
- "Out": Indica los nodos hijos del nodo actual a través del idobj del objeto hijo. Puede ser uno o más nodos a los que se debe dirigir. Un nodo puede apuntar a varios nodos, lo cual se indica en la Figura 3.18. Si "Out" está vacío, significa que es el último nodo del diagrama y ahí finaliza.
- "Name": Nombre del objeto.
- "Locked": Indica si el objeto está bloqueado o no.

#### 3.2.1.7 Análisis léxico

El proceso del analizador léxico consiste en leer la secuencia de caracteres que conforma el programa fuente y agruparlos en secuencias significativas [31]. Esto se logra clasificando la cadena de acuerdo con las palabras válidas en el lenguaje, lo cual se realiza en forma de lexemas o tokens. La Figura 3.20 [31] ilustra cómo se aplican las reglas léxicas definidas para obtener los tokens de una etiqueta de un archivo XML que describe un atributo del modelo de dominio.

El analizador léxico es una parte importante del proceso de compilación, ya que es el encargado de transformar el código fuente en una secuencia de tokens que pueden ser procesados por el compilador para generar el código ejecutable.



**Figura 3.26 Ejemplo de la obtención de tokens de una etiqueta XML**

En el caso de la herramienta ANTLR se utilizan otras aproximaciones más cercanas al análisis gramatical, en el listado 3.1 se exponen los *tokens* que se identificaron donde las líneas del 2 al 21 son símbolos o palabras que se repiten dentro de XML que carecen de información relevante, pero son necesarios para identificar estructuras complejas que consecutivamente a otros tokens dan la información necesaria. En cuanto a las líneas 22 a 36 son palabras reservadas necesarias que tienen información relevante, es por ello el prefijo *PR\_* y se relaciona con el análisis anterior.

```

1  Lexer grammar SodraXMLLexer;
2  COMENTARIO: '<!--' .*? '-->' -> skip;
3  CDATA: '<![CDATA[' .*? ']]>' -> skip;
4  DTD: '<!' .*? '>' -> skip; /** reconoce !ENTITY, !ATTRIB y otros elementos en DTD */
5  ETIQ_INI: '<' -> pushMode(ETIQUETA);
6  XML_INI: '<?xml' WS_INTERNO -> pushMode(ETIQUETA);
7  ENTITY: '&' .*? ';' ;
8  REF_CHAR: '&#' DIGITO+ ';' / '&#x' HEXADEC+ ';' ;
9  WS_EXTERNO: (' / '\t' / '\r'? '\n') ;
10 TEXTO: ~[<&]+ ;
11 fragment
12 DIGITO: [0-9] ;
13 fragment
14 HEXADEC: [a-fA-F0-9] ;
15 mode ETIQUETA ;
16 ETIQ_FIN: '>' -> popMode ;
17 XML_PROC_INST_FIN: '?>' -> popMode ;
18 ETIQ_AUTOFIN: '/>' -> popMode ;
19 DIAG: '/' ;
20 IGUAL: '=' ;
21 CADENA: '"' ~[<" ] .*? '"' ;
22 PR_XMI_ETIQ: 'xmi:XMI' ;
23 PR_DIAGRAM: 'sodra:Diagram' ;
24 PR_DIAGRAM_ELEMENTS: 'sodra:Diagram.elements' ;
25 PR_DIAGRAM_ELEMENT: 'sodra:DiagramElement' ;
26 PR_DIAGRAM_TYPE: 'diagramType' ;
27 PR_NAME: 'name' ;
28 PR_SUBJECT: 'subject' ;
29 PR_XMI_ID: 'xmi:id' ;
30 PR_IDOBJ: 'idobj' ;
31 PR_ACTION: 'action' ;
32 PR_IN: 'in' ;
33 PR_OUT: 'out' ;
34 PR_LOCKED: 'locked' ;
35 PR_VALUE: 'value' ;
36 ID: [a-zA-Z]([a-zA-Z0-9] / '-' / '_' / '.') * ;
37 WS_INTERNO: [ \t\r\n ] -> skip ;
38

```

### Listado 3.3 Fragmento extraído de SodraXMLLexer.g4

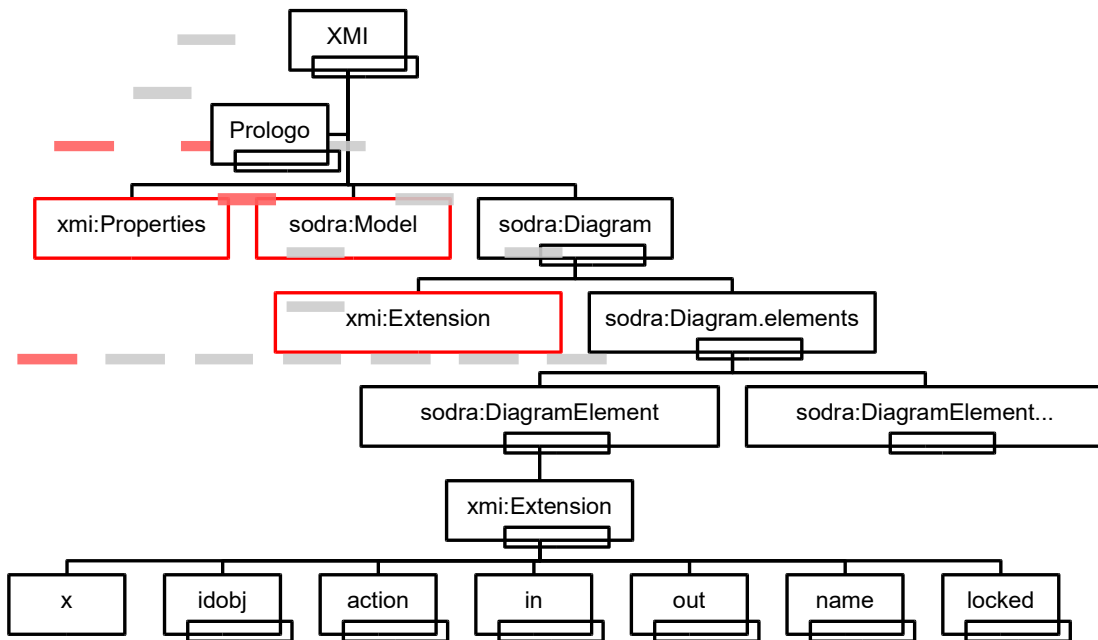
#### 3.2.1.8 Análisis gramatical

El análisis gramatical es la segunda fase del proceso de traducción y tiene como objetivo verificar que la construcción de la cadena de entrada sea correcta según las reglas del lenguaje [31]. En esta fase, el analizador utiliza los *tokens* producidos por el analizador léxico para crear una representación similar a un árbol, que simboliza la estructura gramatical de la secuencia de *tokens*.

Esta representación permite analizar la sintaxis del programa fuente y comprender su estructura gramatical de acuerdo con las reglas gramaticales del lenguaje en cuestión. Además, el análisis gramatical es el encargado de detectar errores

sintácticos en el código fuente, como el uso incorrecto de operadores, la falta de paréntesis o la omisión de comas.

El análisis gramatical es una fase crucial del proceso de compilación ya que garantiza que el código fuente cumpla con las reglas gramaticales del lenguaje y pueda ser traducido correctamente en un código ejecutable.



**Figura 3.27 ATS de modelo XMI**

Se utilizó un Árbol de Sintaxis Abstracta (AST) para representar el modelo XMI fuente en la figura 3.21. Este modelo es el mismo tanto para el modelo de cliente como para el modelo navegacional, y se usó para analizar los diferentes niveles, reglas y *tokens* que conforman la gramática del XMI. Después de revisar el contenido del archivo XMI fuente, se identificaron las etiquetas que lo conforman y se clasificaron según su nivel de profundidad y el nodo al que pertenecen, lo cual se puede apreciar en la figura 2.21.

El tipo de estrategia de análisis sintáctico en ANTLR4 es descendente, es decir, del axioma hacia la cadena, también se conocen como LL (*Left to right, Left most derivation*) [17]. Algunas reglas están definidas en el listado 3.2, utilizando el AST de la figura 3.21 para comparar la regla contra la fuente (lexemas). Para relacionar

el ATS de la figura 3.21 con el listado 3.2 se referirá a los elementos en subrayado al primero y en **negrita** los elementos del segundo. La línea 3 del listado, **docum** corresponde al **XMI** junto con el **prólogo** dando inicio al recorrido del AST. En la línea 5, los **elem\_otro** se refiere a xmi:Properties y sodra:Model que están en color rojo ya que se omitirá su análisis, pues no contienen información relevante y su recorrido costaría tiempo en el procesamiento, pero a pesar de eso se verifica que cumpla la estructura de XMI 2.1, sodra:Diagram es una etiqueta relevante para la generación de metamodelo, es por ello que mediante **diagram** se hace referencia a su estructura en la línea 7 que está formada por *tokens* vistos en el listado 3.1.

```

1 parser grammar SodraXMIParser;
2 options {tokenVocab=SodraXMLLexer;} /** No soporta el import debido a Los modos, se procesan por separado */
3 docum: prologo xmi miscelaneo*;
4 prologo: XML_INI atributo* XML_PROC_INST_FIN miscelaneo*;
5 xmi: datosTexto? ETIQ_INI PR_XMI_ETIQ atributo* ETIQ_FIN elem_otro elem_otro diagram datosTexto?
6     ETIQ_INI DIAG PR_XMI_ETIQ ETIQ_FIN;
7 diagram: datosTexto? ETIQ_INI PR_DIAGRAM atrib_diagramType atrib_name atributo* ETIQ_FIN elem_otro cont_diagram
8     + datosTexto? ETIQ_INI DIAG PR_DIAGRAM ETIQ_FIN;
9 cont_diagram: datosTexto? ETIQ_INI PR_DIAGRAM_ELEMENTS ETIQ_FIN (datosTexto? cont_elements datosTexto?)+ datosTexto?
10    ETIQ_INI DIAG PR_DIAGRAM_ELEMENTS ETIQ_FIN;
11 cont_elements: datosTexto? ETIQ_INI PR_DIAGRAM_ELEMENT atributo atrib_subject atrib_xmi_id ETIQ_FIN cont_element
12    + datosTexto? ETIQ_INI DIAG PR_DIAGRAM_ELEMENT ETIQ_FIN;
13 cont_element: datosTexto? ETIQ_INI ID atributo ETIQ_FIN (datosTexto? element datosTexto?)+ datosTexto? ETIQ_INI DIAG ID ETIQ_FIN;
14 element: etiq_idobj / etiq_action / etiq_in / etiq_out / etiq_name / etiq_locked / elem_otro;
15 etiq_idobj: datosTexto? ETIQ_INI PR_IDOBJ atrib_value ETIQ_AUTOFIN;
16 etiq_action: datosTexto? ETIQ_INI PR_ACTION atrib_value ETIQ_AUTOFIN;
17 etiq_in: datosTexto? ETIQ_INI PR_IN atrib_value ETIQ_AUTOFIN;
18 etiq_out: datosTexto? ETIQ_INI PR_OUT atrib_value ETIQ_AUTOFIN;
19 etiq_name: datosTexto? ETIQ_INI PR_NAME atrib_value ETIQ_AUTOFIN;
20 etiq_locked: datosTexto? ETIQ_INI PR_LOCKED atrib_value ETIQ_AUTOFIN;
21 elem_otro: datosTexto? ETIQ_INI ID atributo* ETIQ_AUTOFIN
22    / datosTexto? ETIQ_INI ID ETIQ_FIN (datosTexto? elem_otro datosTexto?)+ datosTexto? ETIQ_INI DIAG ID ETIQ_FIN
23    / datosTexto? ETIQ_INI ID atributo* ETIQ_FIN (datosTexto? elem_otro datosTexto?)+ datosTexto? ETIQ_INI DIAG ID ETIQ_FIN;
24 contenido: datosTexto? ((elem_otro/referencia/COMENTARIO) datosTexto?)*;
25 referencia: ENTITY / REF_CHAR;
26 atrib_diagramType: PR_DIAGRAM_TYPE IGUAL CADENA;
27 atrib_name: PR_NAME IGUAL CADENA;
28 atrib_subject: PR_SUBJECT IGUAL CADENA;
29 atrib_xmi_id: PR_XMI_ID IGUAL CADENA;
30 atrib_value: PR_VALUE IGUAL CADENA;
31 atributo: ID IGUAL CADENA / atrib_name / atrib_xmi_id / atrib_value;
32 datosTexto: TEXTO / WS_EXTERNO;
33 miscelaneo: COMENTARIO / WS_EXTERNO;
34

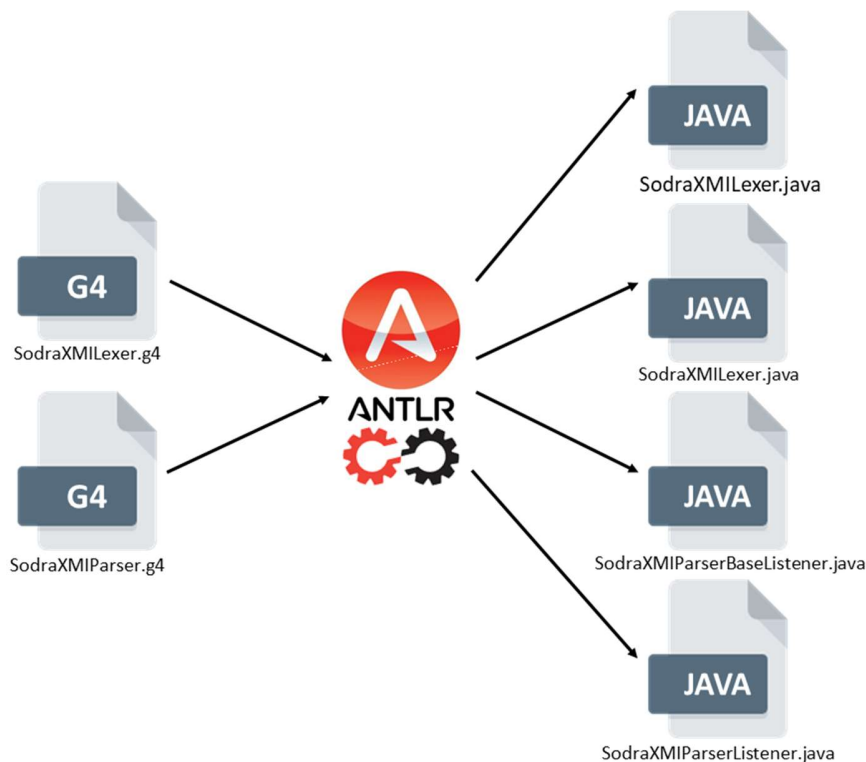
```

### Listado 3.4 Archivo completo SodraXMIParser.g4

En la línea 9, **cont\_diagram** permite acceder a los elementos de sodra:Diagram.elements y omite a xmi:Extension por su irrelevancia en información. Para las líneas 11, 13 y 14 abren los elementos sodra:Diagram.elements, sodra:DiagramElement y xmi:Extension respectivamente verificando las estructura correcta y orden. La línea 14 se encuentran las etiquetas: idobj, action, in, out, name y locked identificados con el prefijo **etiq\_xxxx** (donde las x corresponden a su nombre) que van de la línea 15 al 20, su estructura es similar variando en el *token* PR\_XXXX. De la línea 26 al 30 con el prefijo **atrib\_xxxx** (donde las x corresponden

a su nombre) son aquellos atributos que se encuentran en el documento y apoyaran al recorrido en la siguiente fase. El resto de líneas son genéricas que se adaptan a otros elementos no relevantes.

Después de construir los tokens, ANTLR4 (versión antlr-4.9-complete.jar) se utilizó para procesarlos y generar cuatro clases de Java, como se muestra en la Figura 3.22. Estas clases son esenciales para reconocer las sentencias del lenguaje que cada gramática describe.



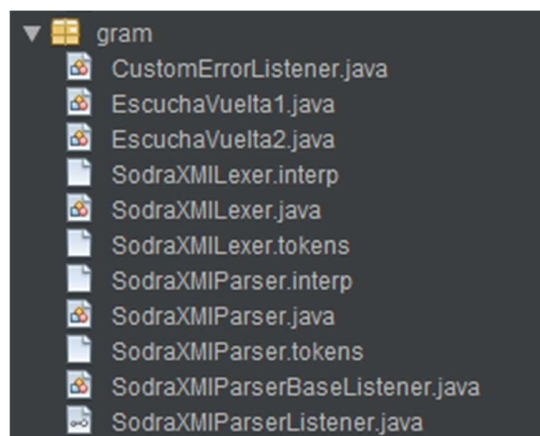
**Figura 3.28 Archivos Java generados por ANTLR4 para el reconocimiento y recorrido del del contenido de los archivos XMI**

El archivo `sodraXMILexer.java` es generado por ANTLR al analizar las reglas léxicas y los literales gramaticales definidos en `SodraXMILexer.g4`, y contiene la definición de la clase del analizador léxico. Por otro lado, `SodraXMIParser` contiene la definición de la clase del analizador específico de las gramáticas establecidas en `SodraXMIParser.g4`. El analizador de ANTLR, de manera predeterminada, construye un árbol a partir de la entrada y emite eventos a un objeto escucha proporcionado. `SodraXMIParserListener.java` es una interfaz que describe los

métodos a implementar para el recorrido del AST, mientras que `SodraXMIParserBaseListener.java` contiene implementaciones vacías por defecto de estos métodos definidos en la interfaz `SodraXMIParserListener.java`.

ANTLR4 ofrece dos enfoques para el recorrido del AST: patrones listener y patrones visitor. En este módulo en particular, se utiliza el enfoque de patrones listener. Los métodos de visita a los nodos hijos son llamados automáticamente por un objeto de tipo `ParseTreeWalker`, que permite recorrer el AST. Estos métodos no tienen un valor de retorno.

Las clases generadas por ANTLR4, como `sodraXMLexer.java` y `SodraXMIParser.java`, se incluyeron en el proyecto y se utilizan para validar el archivo XMI de entrada y extraer la información necesaria para la generación de código. En la Figura 3.23 se muestra el paquete que contiene estas clases generadas.

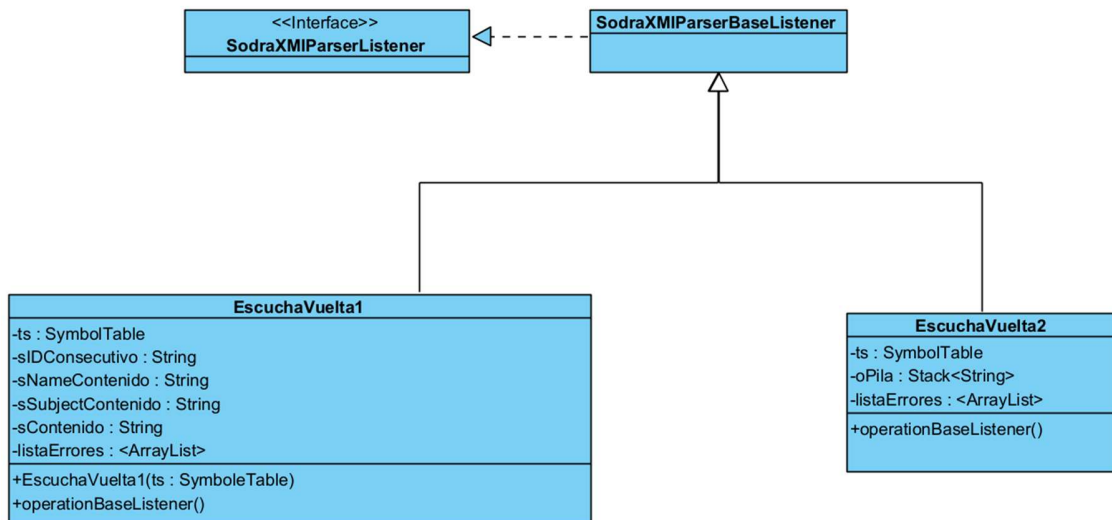


**Figura 3.29 Paquete gram del módulo del generador MOF**

La Figura 3.23 muestra las clases "EscuchaVuelta1" y "EscuchaVuelta2" podrían ser clases implementadas por el desarrollador del proyecto, utilizando el enfoque de patrones listener proporcionado por ANTLR4. Estas clases podrían tener métodos implementados para recuperar la información alojada en cada etiqueta reconocida en el archivo XMI durante el proceso de análisis gramatical. Por ejemplo, podrían tener métodos que sean llamados automáticamente cuando se visite cada nodo del AST correspondiente a las etiquetas del archivo XMI, y dentro de estos métodos se

realice la extracción de la información necesaria para la generación de código u otras acciones específicas del proyecto.

Es importante tener en cuenta que los nombres exactos y el funcionamiento de estas clases dependerán de la implementación específica del proyecto y las reglas gramaticales definidas en las gramáticas ANTLR4.

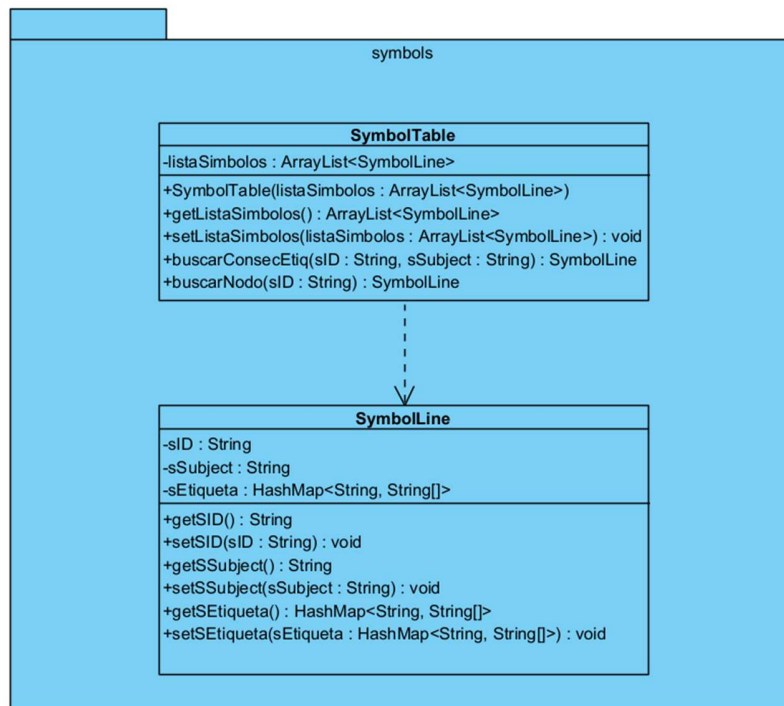


**Figura 3.30 Diagrama de clases del paquete `sodra.moduloMOF.gram`**

El diagrama de clases de la Figura 3.24 presenta dos clases importantes: `EscuchaVuelta1` y `EscuchaVuelta2`. Estas clases se encargan de recuperar la información almacenada en cada etiqueta reconocida en los archivos XMI, utilizando el patrón de escuchas para recorrer el AST.

Para facilitar la tarea de encontrar rápidamente cada etiqueta y sus características asociadas, se definió una tabla de símbolos que almacena los datos relevantes de cada una de ellas. En particular, se almacenan como entrada en la tabla de símbolos los atributos más relevantes de cada etiqueta, como su identificador, nombre, tipo, alcance y otros atributos definidos. De esta manera, el traductor puede acceder rápidamente a la información relevante de cada etiqueta para realizar la traducción de los archivos XMI de manera efectiva.





**Figura 3.31 Diagrama de clases del paquete sodra.moduloMOF.symbols**

### 3.2.1.9 Análisis semántico

Para asegurarse de que el programa de origen se ajuste a la definición del lenguaje y tenga una coherencia semántica, se utiliza el análisis semántico. Este proceso utiliza un árbol de sintaxis y la tabla de símbolos para verificar la información y realizar una verificación estática de la información. También recopila información de tipos para su uso posterior en la generación de código intermedio.

En los compiladores de alto nivel, una de las verificaciones más importantes del análisis semántico es la verificación de tipos, que comprueba que cada operador tenga operandos coincidentes, realiza una verificación de coerción en casos permitidos por la especificación del lenguaje, revisa los flujos de control, la unicidad y consistencia de nombres, entre otras cosas [31].

En el caso de las gramáticas especificadas en el módulo, se llevaron a cabo diversas comprobaciones semánticas, entre las que se destacan:

- Comprobación de la consistencia del XMI 2.1

Para la comprobación de este rubro se verifica que el identificador de inicio concuerde con el final, siguiendo las reglas del estándar XMI 2.1, en la figura listado 3.3 se aprecian estas reglas implementadas al guardar en la tabla de símbolos.

```
400      @Override
401      public void exitDiagram(SodraXMIParser.DiagramContext ctx) {
402          pila.pop();
403          if (!ctx.PR_DIAGRAM(0).getText().equals(ctx.PR_DIAGRAM(1).getText())) {
404              this.listaErrores.add("Línea "
405                  + ctx.PR_DIAGRAM(1).getSymbol().getLine() + ":"
406                  + ctx.PR_DIAGRAM(1).getSymbol().getCharPositionInLine()
407                  + " identificador no consistente "
408                  + ctx.PR_DIAGRAM(1).getText());
409          }
410      }
```

### **Listado 3.5 Comprobación de consistencia en EscuchaVuelta1.java**

- Revisión de la definición de identificadores y su uso.

Para que un identificador sea válido debe existir y estar definido en el modelo para su uso e interacción, pero sin duplicarse. Para su uso, se realiza la comprobación que esté relacionado a otros identificadores, a los que hace referencia a una etiqueta que se encuentre definida dentro del documento. En el listado 3.4 se muestra esta verificación.

```

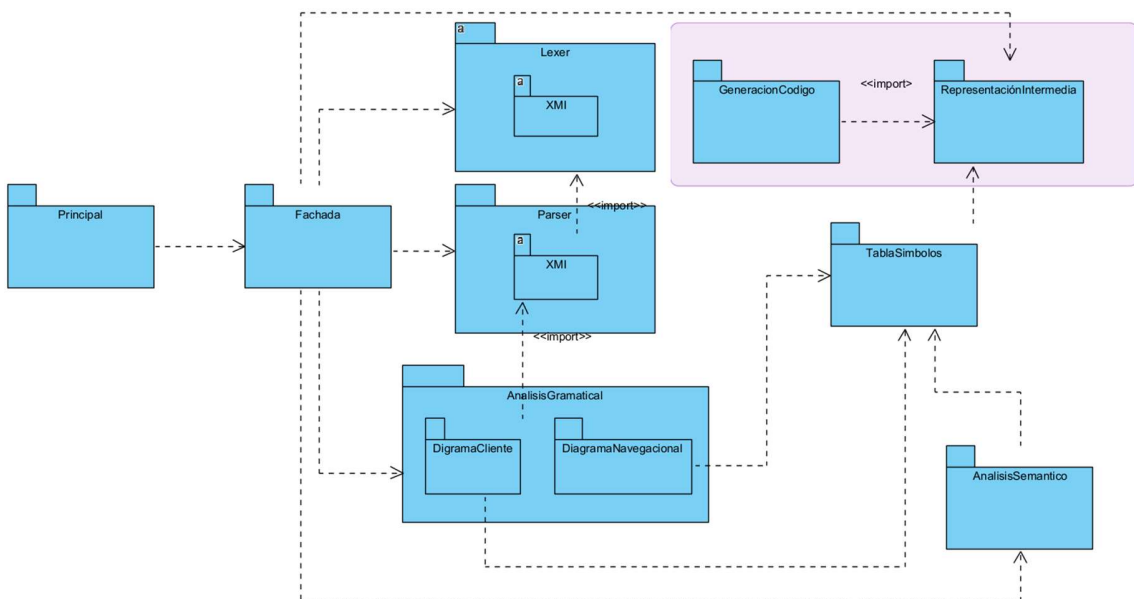
31  @Override
32  public void exitCont_elements(SodraXMLParser.Cont_elementsContext ctx) {
33      ArrayList<SymbolLine> lineasDef, lineasDefIDtoIn, lineasDefIDtoOut;
34      lineasDef = this.ts.buscarID(sIDConsecutivo);
35
36      if (lineasDef.size() != 1) {
37          this.listaErrores.add("Línea "
38              + ctx.PR_DIAGRAM_ELEMENT(0).getSymbol().getLine() + ":"
39              + ctx.PR_DIAGRAM_ELEMENT(0).getSymbol().getCharPositionInLine()
40              + " el identificador esta duplicado");
41      }
42
43      lineasDefIDtoIn = this.ts.buscarIDtoIn(sIDConsecutivo);
44      lineasDefIDtoOut = this.ts.buscarIDtoOut(sIDConsecutivo);
45      if (lineasDefIDtoIn.isEmpty() && lineasDefIDtoOut.isEmpty()) {
46          this.listaErrores.add("Línea "
47              + ctx.PR_DIAGRAM_ELEMENT(0).getSymbol().getLine() + ":"
48              + ctx.PR_DIAGRAM_ELEMENT(0).getSymbol().getCharPositionInLine()
49              + " el identificador no esta relacionado o no esta definido");
50      }
51      sIDConsecutivo = "";
52  }

```

**Listado 3.6 Comprobación de verificación en EscuchaVuelta2.java**

### 3.2.1.10 Planificación

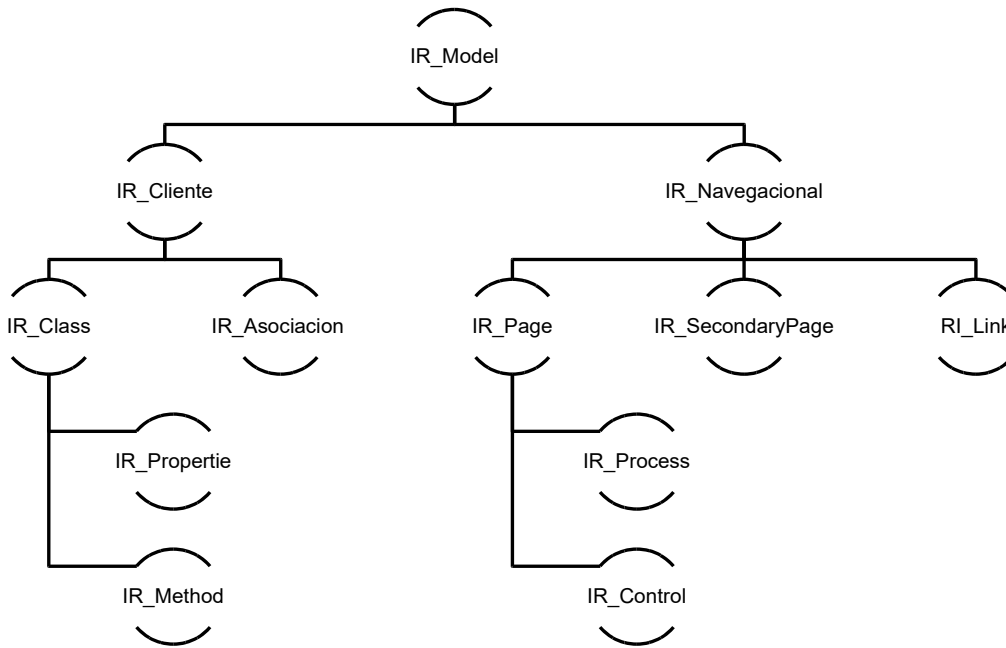
El ciclo finaliza con la estipulación de los archivos generados por ANTLR y sus derivados para el análisis semántico, donde la estructura entre estos elementos se visualiza en la figura 3.26 representando su interacción. El segundo ciclo estima los elementos de representación intermedia y generación de código que están dentro del rectángulo rosa.



**Figura 3.32 Estructura de la aplicación al finalizar el primer ciclo de la espiral**

**3.2.2 Segunda iteración**

La segunda etapa del módulo generador de metamodelo MOF en el estándar XMI 2.5.1, en la cual se lleva a cabo la especificación de la representación intermedia y la generación de modelo final.



**Figura 3.33 AST de la representación intermedia**

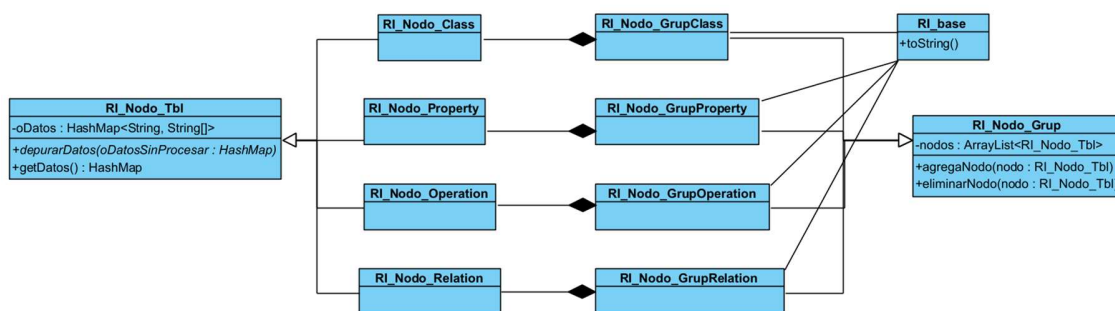
**3.2.2.1 Representación intermedia**

Durante el proceso de compilación de un programa fuente a código objetivo, es común que se utilicen una o varias representaciones intermedias, como los árboles de sintaxis, que son utilizados durante los análisis sintáctico y semántico. En la Figura 3.27 se puede observar la estructura de la representación intermedia utilizada en este proyecto, que consiste en un árbol de sintaxis abstracto (AST) con las especificaciones de los componentes encontrados en el modelo de cliente o navegacional de SODRA. El nodo raíz es IR\_MODEL, del que se derivan dos ramas principales: IR\_Cliente e IR\_Navegacional, que corresponden al modelo de cliente y modelo navegacional, respectivamente.

La rama IR\_Cliente muestra las propiedades de clases, atributos, métodos y sus parámetros, mientras que IR\_Navegacional incluye RI\_Pages como contenedores

y RI\_SecondaryPages como componentes de la vista, RI\_link como los flujos de navegación y sus reglas de validación. Por otro lado, la Figura 3.28 muestra un fragmento del diagrama de clases de la representación intermedia, donde cada tipo de nodo del árbol tiene una clase que hereda directamente de la clase IR\_Node\_tbl. Además, cada nodo se enfoca en las propiedades relevantes y representa el comportamiento de cualquier nodo de representación intermedia. También existe la clase abstracta IR\_Node\_Grup, que agrupa los nodos y permite su gestión para crear o eliminar nodos, y RI\_base, que funciona como fachada para acceder a los grupos de nodos.

Es importante destacar que, para verificar la coherencia semántica del programa de origen con la definición del lenguaje, se realiza un análisis semántico utilizando el árbol de sintaxis y la información de la tabla de símbolos. Durante este proceso, se llevan a cabo verificaciones importantes, como la verificación de tipos, la revisión de flujos de control, la unicidad y consistencia de nombres, entre otras. En particular, para las gramáticas especificadas en este proyecto se realizaron una serie de comprobaciones semánticas, entre las que se destaca la comprobación de la consistencia del XMI 2.1.



**Figura 3.34 Fragmento del diagrama de clases de la representación intermedia del cliente**

La representación intermedia generada es un metamodelo UML abstracto y no está ligada a ninguna tecnología específica. La información extraída del modelo de cliente y navegacional se transforma en nodos que representan elementos comunes en la modelación de clases UML. La tabla 3.8 presenta la correspondencia entre los

nodos de la representación intermedia y el diagrama de clases de acuerdo con el modelo de cliente y navegacional.

**Tabla 3.5 Equivalencia de representación intermedia en nodos y diagrama de clases según el modelo cliente y navegacional**

<b>AST representación intermedia</b>	<b>Diagrama de clases</b>
IR_Page	RI_Node_Class
IR_Process	RI_Node_Operation
IR_SecondaryPage	RI_Node_Class
IR_Control	RI_Node_Operation
RI_Link	RI_Node_Relation
IR_Class	RI_Node_Class
IR_Propertie	RI_Propertie
IR_Method	RI_Node_Operation
IR_Asociacion	RI_Node_Relation

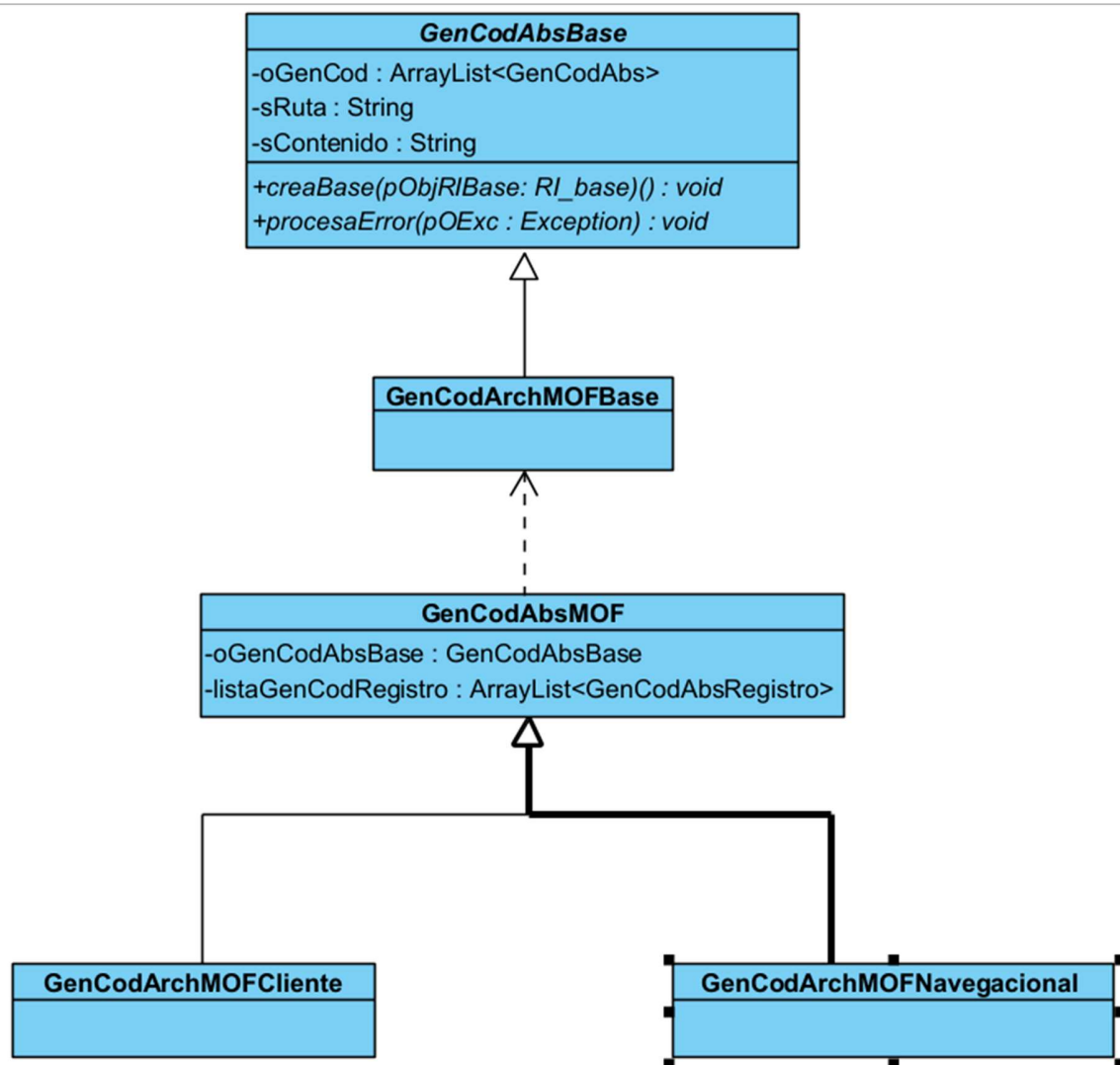
En algunas ocasiones, los elementos del modelo de navegación no tienen una correspondencia directa con los nodos de la representación intermedia. En particular, el RI\_SecondaryPage presentado por query\_node y list\_node en el modelo de navegación genera problemas cuando las páginas y las páginas secundarias se encuentran como elementos separados en el mismo nivel jerárquico. Esto no es apropiado para este proyecto, ya que las páginas secundarias se encuentran dentro de las páginas y no como elementos separados.

### 3.2.2.2 Generación de modelo

Durante la generación de código, se utiliza la información recopilada durante el análisis léxico, sintáctico y semántico para producir el código objeto final. Esto puede ser realizado mediante diferentes técnicas y algoritmos, como la generación de código directo o la generación de código en dos o más pasos.

En la generación de código directo, el compilador traduce cada instrucción del programa fuente en una secuencia de instrucciones en el lenguaje objetivo, de forma que la traducción es realizada de manera directa y sin la necesidad de construir una representación intermedia explícita.

Por otro lado, en la generación de código en dos o más pasos, se construye primero una representación intermedia, la cual es luego utilizada para generar el código objetivo final. Esto puede ser beneficioso en términos de modularidad y reutilización de código, ya que la representación intermedia puede ser utilizada para generar diferentes versiones del programa objetivo.



**Figura 3.35** Conjunto de clases abstractas para la generación de código

La generación de código se realiza a través de la implementación de las clases concretas GenCodArchMOFCliente y GenCodArchMOFNavegacional, las cuales tienen la capacidad de entregar los archivos correspondientes a clases, atributos, métodos y relaciones en archivos XML. Además, se ha considerado la creación de familias de objetos relacionados o dependientes, lo que permite encapsular la creación de objetos de manera separada y facilita la implementación de nuevas generaciones de modelos o código en el futuro. La figura 3.29 muestra el diagrama de clases de la generación de modelo con las firmas de los métodos a ser implementados por estas clases concretas.

```
38         this.sModelo = modelo;
39     try{
40         oGenRI = new EscuchaRI();
41         oGenRI.setTable(ts);
42         oBase = new RI_Base();
43         oGenRI.setBase(oBase);
44         walker = new ParseTreeWalker();
45         walker.walk(oGenRI, tree);
46         oBase = oGenRI.getBase();
47         if (this.sModelo.equals("Cliente"))
48             oGenera = new GenCodArchClienteBase();
49         else
50             oGenera = new GenCodArchNavegacionalBase();
51         oGenera.creaBase(oBase);
52         System.out.println(oGenera.getContenido());
53         this.oSalida.setText("Código generado");
54     }catch(Exception e){
55         processGeneratorError(e);
56     }
```

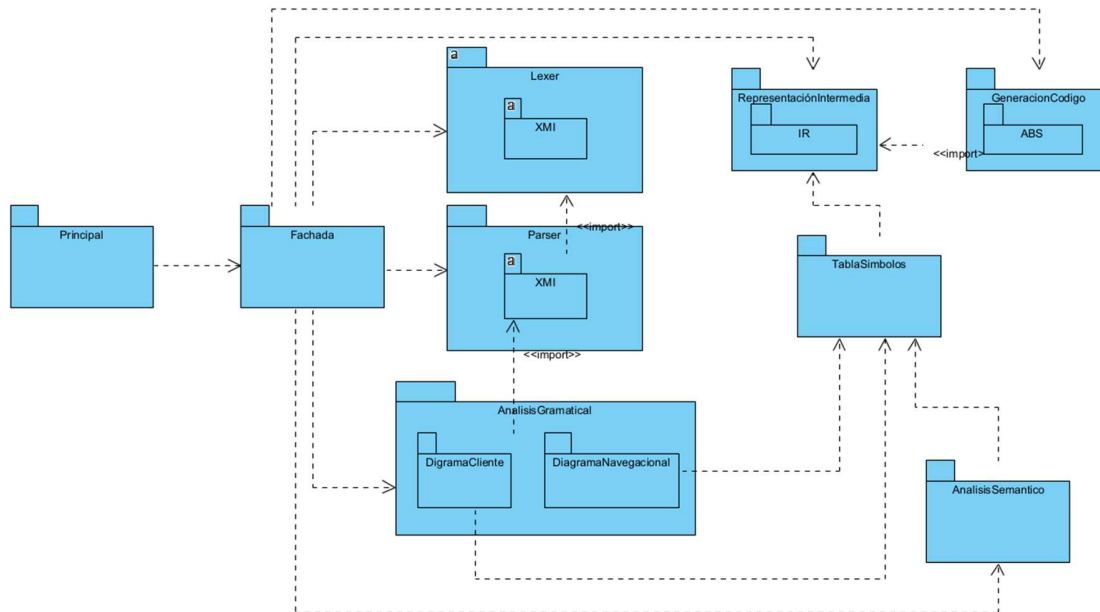
### ***Listado 3.7 Generación de modelo MOF***

En el proceso de generación de código, la clase encargada de realizar el recorrido por el árbol de la representación intermedia es GenCodArchMOFBase. Esta clase implementa el método creaBase(), que se muestra en el listado 3.5. En dicho método, se recibe como parámetro el tipo de modelo (línea 38) y, posteriormente, se crea el archivo específico correspondiente al modelo (líneas 48 y 50). Luego, se procede a recorrer el contenido del nodo que contiene la representación intermedia de la aplicación y se ejecutan los métodos necesarios para crear el modelo (línea 51).



### 3.2.2.3 Planificación

El proceso llega a su fin con la obtención de la estructura final de la aplicación, la cual se muestra en la figura 3.2. En dicha figura se puede apreciar la inclusión de los paquetes correspondientes a la representación intermedia, así como el paquete que contiene las clases abstractas de los elementos a generar.



**Figura 3.36 Estructura de la aplicación al finalizar el segundo ciclo de la espiral**

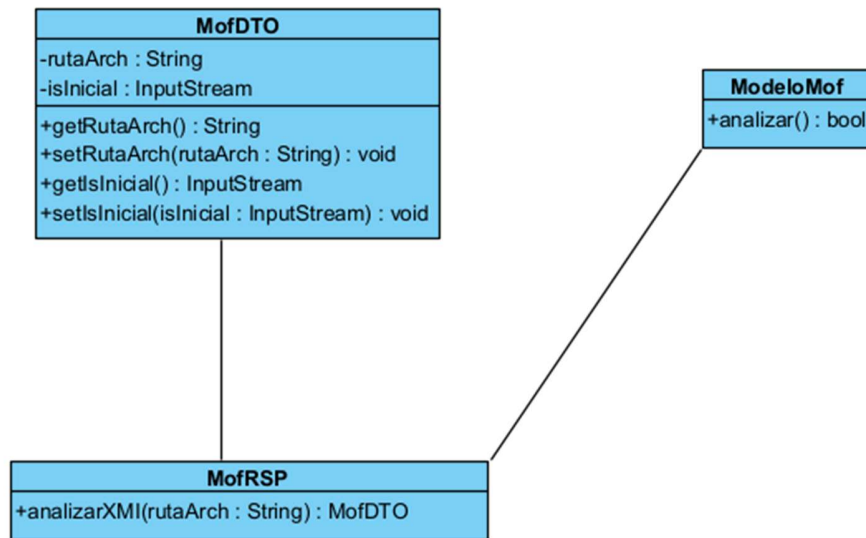
### 3.2.3 Tercera iteración

Corresponde a la tercera etapa del módulo generador de metamodelo MOF bajo el patrón REST junto con los paquetes de clases anteriormente descritas, la creación del servicio es por motivos que la herramienta SODRA está creada en el entorno de PHP y la transformación en este entorno no fue convincente como la propuesta por ANTLR4 y java.

#### 3.2.3.1 Creación del servicio web

Roy Fielding [33] describe como abreviatura REST un estilo arquitectónico de un sistema hipermedia distribuido, es decir, una arquitectura web vista como un sistema hipermedia distribuido a gran escala. Una característica clave de este estilo arquitectónico es la interfaz unificada entre los componentes del sistema.

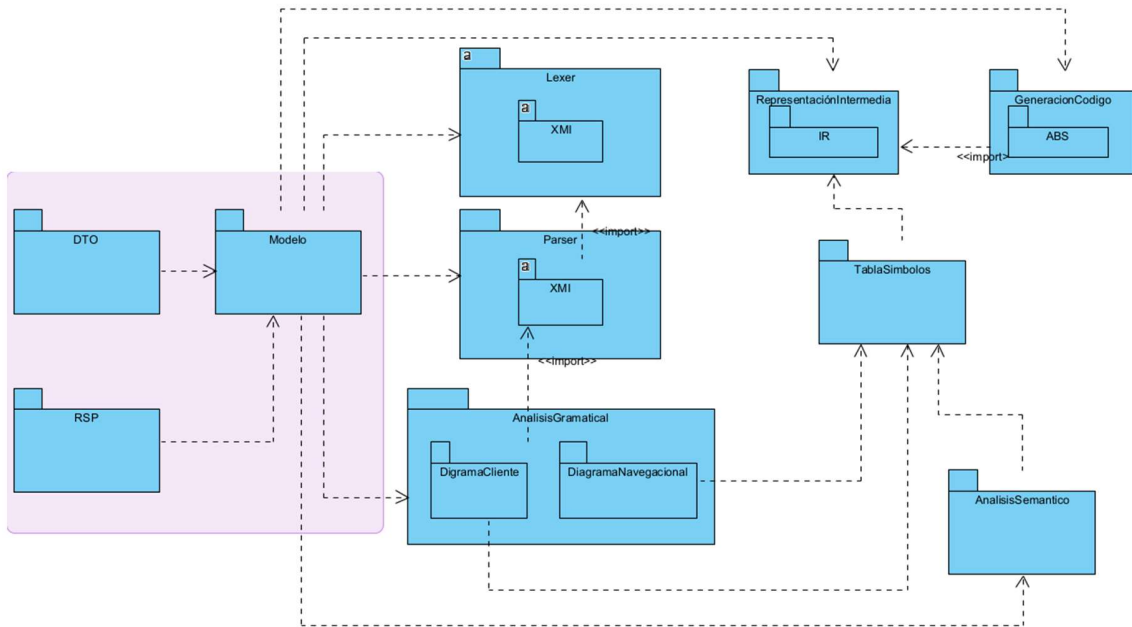
En la Figura 3.31 se describe el diagrama de clases del servicio web, el cual hace uso de dos atributos rutaArch y isInicial, el primero haciendo referencia a la ruta en el servidor local del archivo XMI 2.1 que crea SODRA y el segundo al archivo XMI 2.1 como tal. Esto apoya a la generación de metamodelo sirviendo como comunicador entre SODRA y el módulo de generación MOF colocando este en la misma ruta el nuevo XMI 2.5.1.



**Figura 3.37 Diagrama de clases para servicio REST**

### 3.2.3.2 Planificación

Al finalizar la iteración, el generador puede comunicarse y generar el metamodelo MOF de manera local. En este punto, solo queda la integración con SODRA. La estructura final de paquetes del proyecto se muestra en la Figura 3.32.



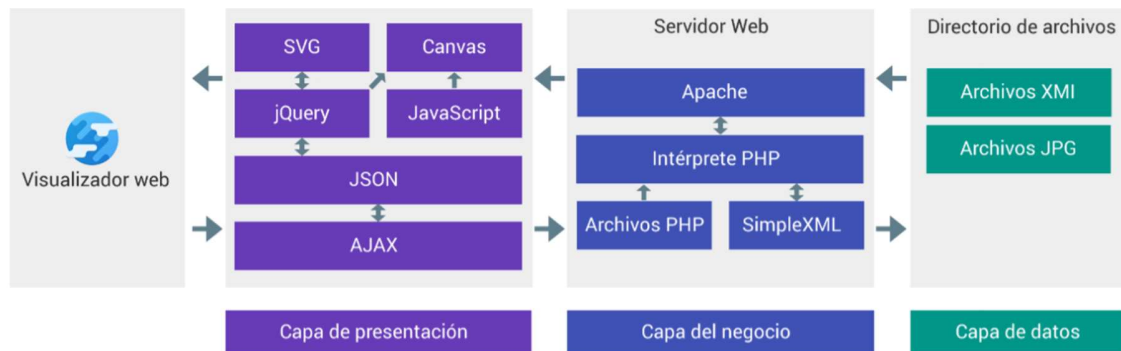
**Figura 3.38 Estructura de la aplicación al finalizar el tercer ciclo de la espiral**

### 3.2.4 Cuarta iteración

Conciene a la última etapa del proceso de desarrollo del módulo generador de metamodelo MOF, en la que se implementa la integración a la herramienta SODRA.

#### 3.2.4.1 Integración del módulo a SODRA

La herramienta SODRA fue desarrollada utilizando una arquitectura basada en tres capas, donde cada capa tiene una responsabilidad específica y ofrece un nivel de abstracción para separar las actividades en tareas más pequeñas y manejables. La figura 3.33 muestra el esquema de la arquitectura de la herramienta SODRA.



**Figura 3.39 Arquitectura del sistema SODRA**

Se explica cada una de las capas de la arquitectura de SODRA según [8]:

- Capa de presentación:** La capa de presentación brinda al usuario una interfaz gráfica para manejar las herramientas que generan los diagramas del modelo cliente y navegacional. Para funcionar, esta capa utiliza bibliotecas que se ejecutan en el lado del cliente, como jQuery, Canvas, SVG y JavaScript, las cuales se encargan de la gestión de los diagramas. Por otro lado, AJAX se encarga de la comunicación con la capa de negocio y devuelve un elemento que sirve como base para construir toda la interfaz de usuario. En la Figura 3.34 se puede observar el diagrama de clases, el cual muestra la clase diagrama.js, perteneciente al paquete js, que incluye el checklist para generar el metamodelo MOF, el cual está disponible tanto para el modelo navegacional como para el modelo cliente.

```

707 exportar : function(){
708     diagrama.save(function(){
709         var wx = w.open([600,180,'','undefined','']);
710         $('<div class="w-w-content#co'+wx>.html(
711             '<div class="w-w-c-title">Elija la forma de exportar su proyecto</div>'
712             + '<div class="w-w-c-tars"></div><div class="w-w-c-ctnx"><div class="w-w-c-o-opt"><label>'
713             + 'form.input(['<div class="w-w-c-o-opt"><label>'
714             + 'form.input(['<div class="w-w-c-o-opt"><label>'
715             + 'form.input(['<div class="w-w-c-m-button">'
716             + 'form.button(['</div></div>');
717     });
718 }

```

**Listado 3.8 Fragmento de código de diagrama.js**

En la figura 3.34, se muestra el nuevo menú de la venta *pop-up* con la nueva opción de exportación.



**Figura 3.40 Ventana pop-up con opciones para la exportación del modelo.**

- Capa del negocio: La capa del negocio se gestiona en un servidor Web y sirve como medio de entrada para acceder a los archivos PHP que se procesarán bajo un intérprete PHP en su versión 5.6. Los archivos PHP se comunican con la capa de datos para recuperar los archivos XMI y entregarlos en la capa de presentación en formato JSON. Además, se utilizará la extensión de PHP *SimpleXML* que, a través de un archivo XMI seleccionado convierta su contenido en un conjunto de arreglos de propósito general. En la figura 3.34 se observa el diagrama de clases donde en el paquete *php* la clase *proyecto.php* se agregó funcionalidad para la petición a la generación de metamodelo MOF, esta opción está disponible tanto en modelo navegacional como en el de cliente.

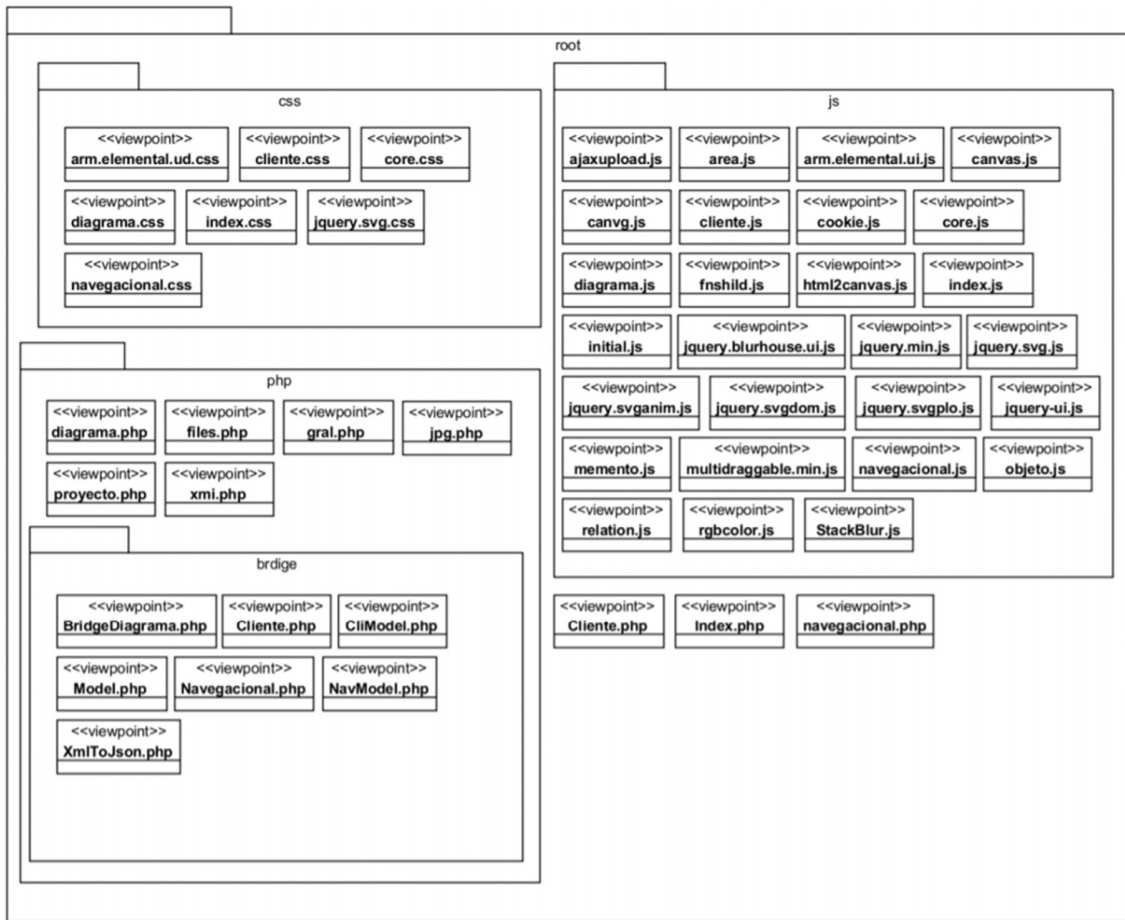
```

276 function exportaProyecto($kind, $spc, $pro, $jpgg, $xmip, $mofp){
277     $rute = "../wkrsp/" . $spc . "/" . $pro;
278     $zip = new ZipArchive();
279     $filename = $rute . "/" . $pro . ".zip";
280     @unlink($filename);
281     if ($zip->open($filename, ZipArchive::CREATE) !== true)
282         $this->arrde(array('0'));
283     $kind = ($kind != 'all') ? array($kind) : array("cliente", "navigacional");
284     for ($i = 0; $i < count($kind); $i++){
285         if ($this->fileExists($rute . "/" . $pro . "-" . $kind[$i] . ".xmi")){
286             if ($xmip == '1')
287                 $zip->addFile($rute . "/" . $pro . "-" . $kind[$i] . ".xmi", $pro . "-" . $kind[$i] . ".xmi");
288             if ($jpgg == '1')
289                 $zip->addFile($rute . "/" . $pro . "-" . $kind[$i] . ".jpg", $pro . "-" . $kind[$i] . ".jpg");
290             if ($mofp == '1'){
291                 try{
292                     //Crear cliente del servicio web
293                     $client = new SoapClient("http://localhost:36512/analizarXMI/"+, array('trace'=>1));
294                     $param["rute"] = $rute . "/" . $pro . "-" . $kind[$i] . ".xmi";
295                     $oResultado = $client->WindChillInFahrenheit($param);
296                 }
297                 if (!$oResultado){
298                     $sErr = "Error al invocar";
299                 }
300             }catch(Exception $e){
301                 error_log($e->getFile()." " . $e->getLine()." " . $e->getMessage(),0);
302                 $sErr = "Error al invocar";
303             }
304             $zip->addFile($rute . "/" . $pro . "-" . $kind[$i] . ".mof.xmi", $pro . "-" . $kind[$i] . ".mof.xmi");
305             if ($sErr != "") header("Location: index.php?sErr=".$sErr);
306         }
307     }
308 }
309 $zip->close();
310 sleep(3);
311 return true;

```

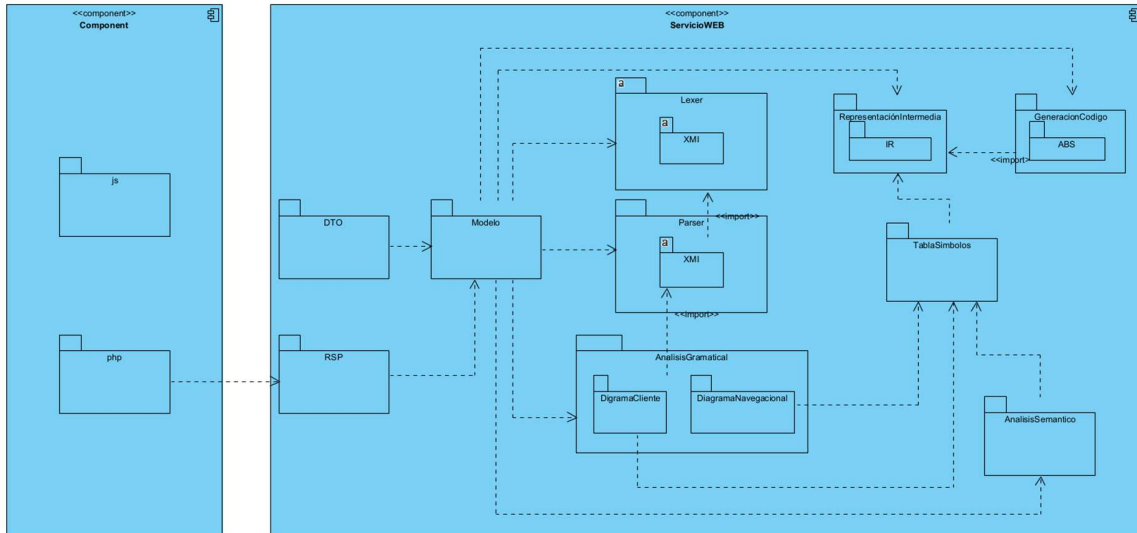
### **Listado 3.9 Fragmento de código de proyecto.php**

- Capa de datos:** La capa de datos almacena los directorios y archivos de los diagramas que se distribuyen según el nombre del proyecto. Los archivos XMI 2.5.1 contienen información sobre el meta-metamodelo, mientras que los archivos XMI 2.1 contienen información sobre el metamodelo y el modelo, los cuales son necesarios para su interpretación y gestión en las capas de negocio y presentación. Por último, los archivos JPG contienen una vista previa del diagrama y se utilizan como recurso de propósito general.



**Figura 3.41 Diagrama de paquetes de la herramienta SODRA**

La Figura 3.36 resalta los últimos elementos añadidos al generador de metamodelo MOF al concluir su proceso de desarrollo.



**Figura 3.42 Estructura de la aplicación al finalizar el cuarto ciclo de la espiral**



## **Capítulo 4. Resultados**

### **4.1 Planteamiento del caso de estudio**

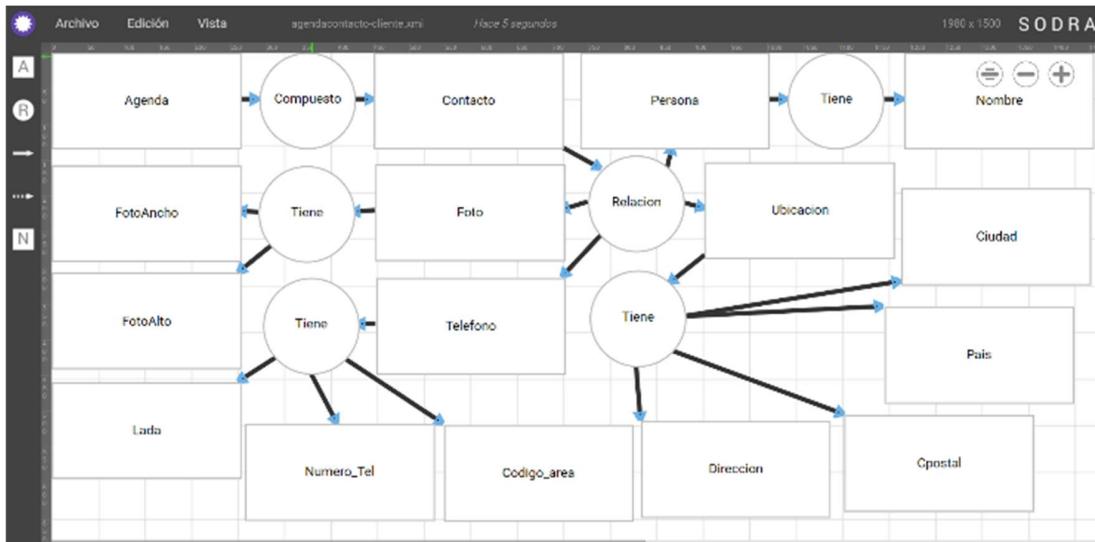
Este caso de estudio se basa en un trabajo de tesis anterior [8], y tiene como objetivo comparar dos trabajos. El primer trabajo de estudio implica el desarrollo de una aplicación de agenda de contactos que realiza cuatro tareas: buscar contactos, eliminar contactos, crear nuevos contactos y actualizar contactos existentes. Para lograr esto, se han identificado cinco clases: Agenda, Contacto, Dirección, Teléfono y Foto.

La clase Agenda es la encargada de manejar la lista de contactos, mientras que la clase Contacto contiene los detalles del contacto. La clase Dirección tiene atributos como la ciudad, país, dirección y código postal, mientras que la clase Teléfono tiene atributos como el código de área, lada y número. La clase Foto tiene atributos como alto y ancho.

Para llevar a cabo las tareas de la agenda, se han identificado cuatro acciones principales: buscar, eliminar, crear y actualizar.

En resumen, se ha presentado un caso de estudio que se basa en un trabajo de tesis anterior [8] y se compara con el trabajo actual. El objetivo del caso de estudio es desarrollar una aplicación de agenda de contactos que puede realizar cuatro tareas principales. Se han identificado cinco clases y cuatro acciones para llevar a cabo estas tareas.

#### 4.1.1 Generación de metamodelo MOF del modelo de cliente



**Figura 4.43** Modelo de cliente desarrollado en SODRA para caso de estudio de Agenda

Se ha seleccionado el caso de estudio "Agenda telefónica" para validar la especificación de la transformación en este trabajo. La Figura 4.1 muestra la estructura de la agenda, que está compuesta por la entidad "Contacto" y tiene relaciones con "Ubicación", "Teléfono", "Foto" y "Persona". Cada entidad tiene atributos específicos según lo solicitado.



En la Figura 4.3 se han determinado algunos de estos atributos, así como las relaciones entre la agenda y el contacto, así como entre el contacto y la dirección, el teléfono y la foto. La notación utilizada en este diagrama se puede obtener haciendo clic en el menú "vista" y seleccionando el elemento "ver notación", o accediendo a la configuración de teclas Ctrl + O.

En la Figura 4.18 se muestra una notación propuesta en [10] que simplifica las relaciones entre los nodos para una mejor comprensión del modelo.



Para exportar el diagrama, se puede acceder al menú "Archivo" y seleccionar la opción "Exportar", o utilizar la combinación de teclas Ctrl + E. En la Figura 4.3 se muestra la interfaz de usuario que permite seleccionar los formatos de archivo deseados para la exportación.

Para este caso de estudio en particular, se han seleccionado dos formatos para ejemplificar el proceso de generación del metamodelo MOF. La Figura 4.4 muestra los dos archivos que se han exportado desde SODRA.

Nombre	Tipo	Tamaño
 agendacontacto.zip	Archivo WinRAR ZIP	2 KB
 agendacontacto-clientemof.xml	Archivo XMI	6 KB

**Figura 4.46 Archivos resultantes de la exportación MOF del diagrama del cliente**

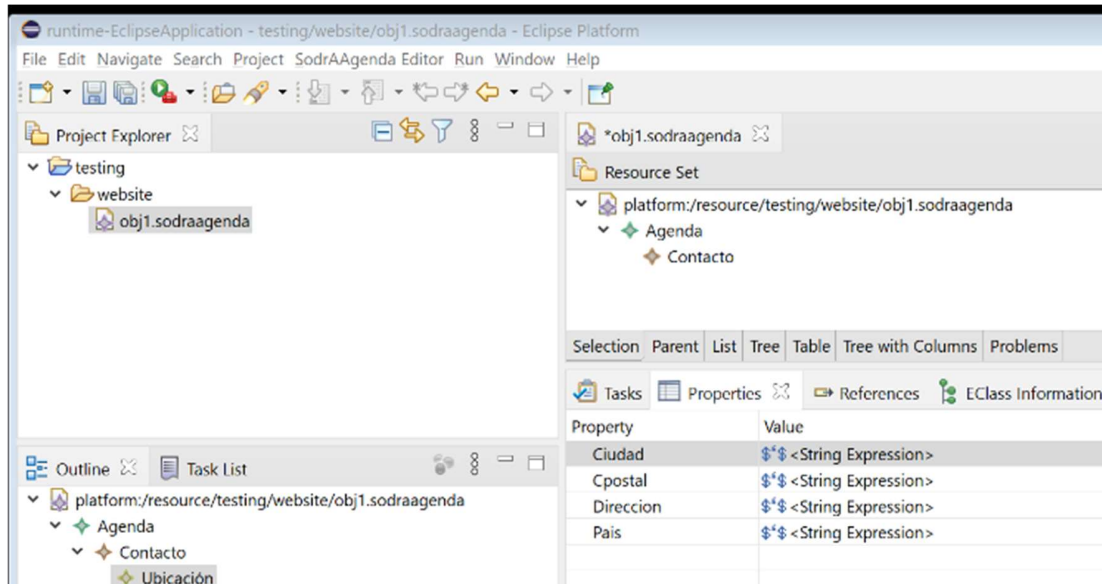
#### 4.1.2 Generación de código a partir del metamodelo MOF del modelo cliente

El objetivo es comprobar hasta qué punto ayuda la generación de metamodelos en el desarrollo de un proyecto. EMF es un conjunto de Eclipse *plugins* que se utiliza para modelado de datos y en el código generado u otra salida basada en este modo. EMF permite al desarrollador importar el metamodelo (*agendacontacto-clientemof.xml*) para definir el modelo de dominio de la aplicación y generar las clases de implementación de Java correspondientes y, a partir de este metamodelo, con el código generado se extiende manualmente de forma segura. De igual forma, en la Figura 4.5. según los archivos *genmodel*, se genera código Java que constará de lo siguiente:

- *SODRASodrAAgenda*: contiene interfaces y la fábrica para crear las clases de Java. En la Figura 11 en el panel izquierdo, se observa la generación de las clases y sus respectivos atributos.
- *SODRASodrAAgenda.impl*: Es la implementación concreta de las interfaces definidas en el modelo.
- *SODRASodrAAgenda.util*: Realiza el patrón de diseño *AdapterFactory*.



que el entorno de diseño en tiempo real se trate como una vista dedicada que se integra consistentemente en un entorno más general, apoyando así al resto del diseño del sistema [34].

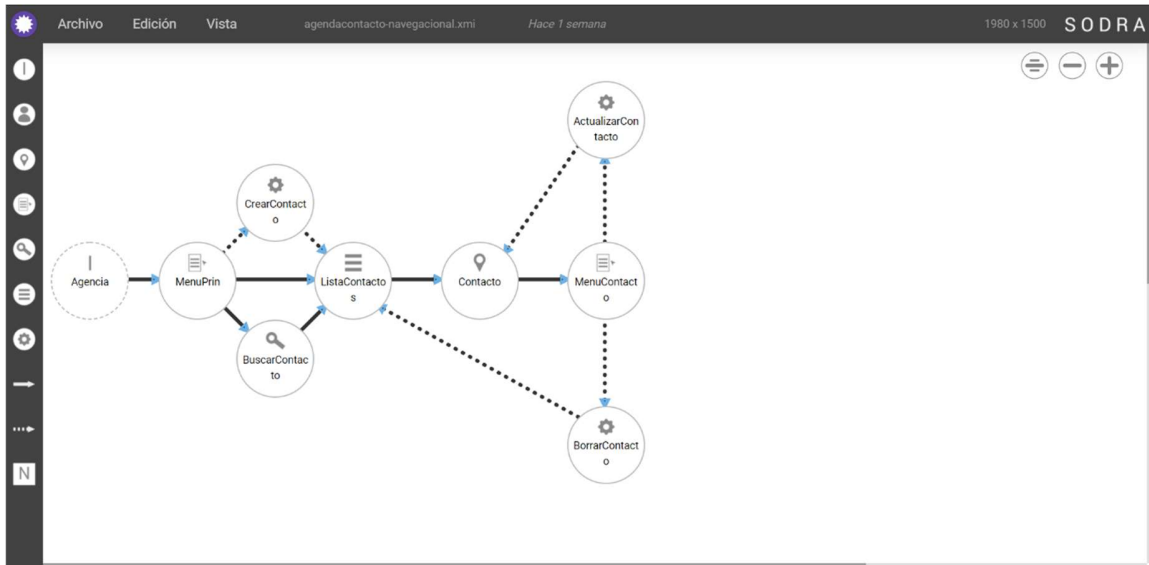


**Figura 4.48 Creación EMF Editor plug-ins**

### 4.1.3 Generación de metamodelo MOF del modelo navegacional

Para generar el diagrama navegacional, es necesario convertir cada clase en un nodo de navegación y establecer los enlaces navegacionales correspondientes, incluyendo las acciones previamente definidas. En la herramienta, se debe seleccionar el cuadro "Diagrama Navegacional" para iniciar el proceso de desarrollo.

Una vez iniciado el proceso, se abrirá la interfaz del lienzo correspondiente al diagrama navegacional. Para agregar un nodo al lienzo, se debe seleccionar el nodo correspondiente y hacer clic en el lugar de destino del lienzo, finalizando la edición del diagrama con su nombre correspondiente. Es importante destacar que, dado que se trata de un diagrama navegacional, se necesita un nodo menú que permita acceder a los recursos de la aplicación web.



**Figura 4.49 Modelo navegacional desarrollado en SODRA para caso de estudio de Agenda**

Después de haber creado el diagrama navegacional, es posible obtener su notación mediante la selección de la opción "Ver Notación" en el menú "Vista", o utilizando la combinación de teclas Ctrl + O. En la Figura 4.7 se puede apreciar la notación propuesta para este diagrama.

```

Notación del diagrama
( Agencia ){ NodoInicio } - [ EnlaceNavegacional ] -> ( MenuPrin ){ NodoMenu };
( MenuPrin ){ NodoMenu } - [ EnlaceProceso ] -> ( CrearContacto ){ NodoProceso };
( MenuPrin ){ NodoMenu } - [ EnlaceNavegacional ] -> ( BuscarContacto ){ NodoConsulta };
( MenuPrin ){ NodoMenu } - [ EnlaceNavegacional ] -> ( ListaContactos ){ NodoLista };
( CrearContacto ){ NodoProceso } - [ EnlaceProceso ] -> ( ListaContactos ){ NodoLista };
( BuscarContacto ){ NodoConsulta } - [ EnlaceNavegacional ] -> ( ListaContactos ){ NodoLista };
( ListaContactos ){ NodoLista } - [ EnlaceNavegacional ] -> ( Contacto ){ NodoNavegacional };
( Contacto ){ NodoNavegacional } - [ EnlaceNavegacional ] -> ( MenuContacto ){ NodoMenu };
( MenuContacto ){ NodoMenu } - [ EnlaceProceso ] -> ( ActualizarContacto ){ NodoProceso };
( MenuContacto ){ NodoMenu } - [ EnlaceProceso ] -> ( BorrarContacto ){ NodoProceso };
( ActualizarContacto ){ NodoProceso } - [ EnlaceProceso ] -> ( Contacto ){ NodoNavegacional };
( BorrarContacto ){ NodoProceso } - [ EnlaceProceso ] -> ( ListaContactos ){ NodoLista };

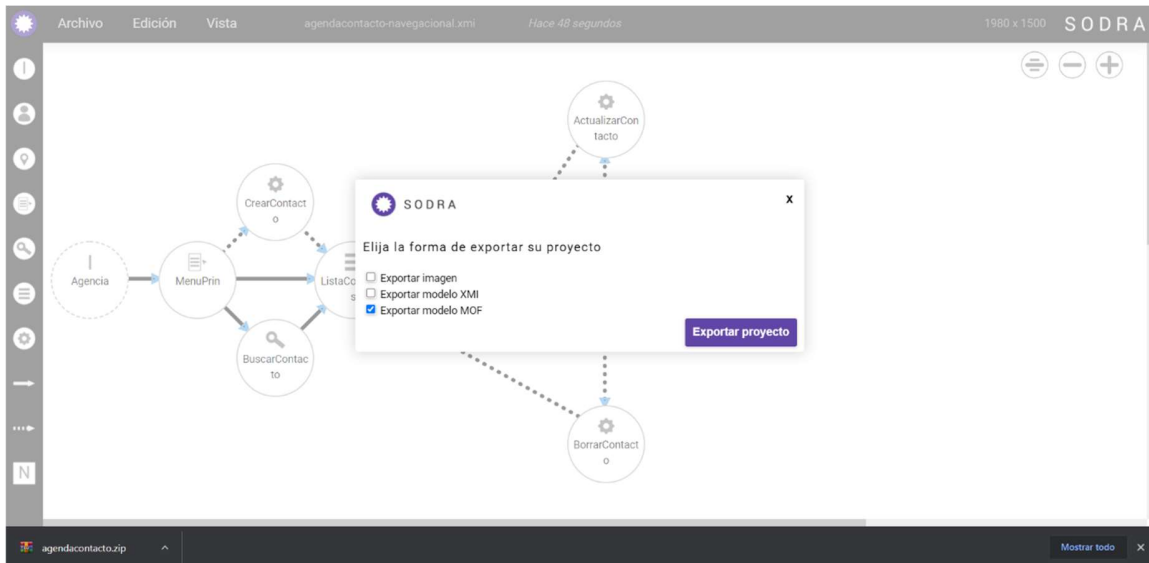
Comentarios de los nodos:

```

**Figura 4.50 Notación del diagrama navegacional del caso de estudio**

Para el modelo navegacional, se realiza la exportación de los formatos XMI 2.1, JPG y MOF 2.0 en estándar XMI 2.5.1 incluido en este trabajo. El XMI 2.5.1 permitirá

servir como punto de partida para la generación de aplicaciones Web, particularmente en este modelo front end. Para exportar el diagrama, se accede al menú Archivo seguido del botón exportar o se realiza la combinación de teclas Ctrl + E. En la Figura 4.9 se muestra la interfaz para decidir los formatos de archivos a exportar.



**Figura 4.51 Exportación de MOF en SODRA Modelo navegacional desarrollado en SODRA para caso de estudio de Agenda**

En la Figura 4.10 se muestran los archivos resultantes de la exportación del diagrama navegacional:

Nombre	Tipo	Tamaño
agendacontacto.zip	Archivo WinRAR ZIP	1 KB
agendacontacto-navegacionalmof.xmi	Archivo XMI	3 KB
agendacontacto-clientemof.xmi	Archivo XMI	6 KB

**Figura 4.52 Archivos resultantes de la exportación MOF del diagrama navegacional**



## **Capítulo 5. Conclusiones y recomendaciones**

En este capítulo se presentan las conclusiones y recomendaciones para este trabajo de tesis. Se destaca la importancia de los sistemas basados en modelos como intermediarios de transformación entre tecnologías y se explica cómo estos sistemas pueden automatizar y agilizar los procesos de desarrollo de software, evitando errores en el proceso y en las pruebas unitarias e integración.

### **5.1 Conclusiones**

Los sistemas basados en modelos sirven como intermediarios de transformación entre diferentes tecnologías, ya que pueden traducir elementos de una tecnología a otra para un manejo de información más preciso y útil. Estos sistemas ayudan a los programadores a optimizar el tiempo y a automatizar y agilizar los procesos de desarrollo de software para evitar errores durante las pruebas unitarias e integración. En esta tesis se desarrolló el complemento de mapeo de modelos a metamodelos para llevar a cabo el proceso de MDA de manera correcta y reducir tiempos y costos en la fase de análisis. Aunque falta el modelo navegacional, se comprobó el uso del modelo de cliente en metamodelo MOF en otras plataformas, lo que hace de SODRA una herramienta sólida para el diseño y generación de proyectos dirigida a modelos.

Es importante tener en cuenta que el nivel de abstracción y detalle del metamodelo depende del diseñador, y si se requiere un nivel de detalle más específico, se deben complementar los atributos de cada clase con los tipos de datos que UML permite y las multiplicidades en cada relación, lo que se puede realizar en EMF o alguna herramienta de diseño que soporte XMI 2.5.1.

En la etapa cuatro de esta tesis, se realizaron nuevas adiciones para los diagramas del cliente y navegacional en la herramienta SODRA, construyendo así una herramienta más completa y eficiente para el desarrollo de software.

En resumen, la tesis trata sobre el desarrollo de un complemento de mapeo de modelos a metamodelos para facilitar el proceso de MDA en el desarrollo de software y reducir los tiempos y costos en la fase de análisis. Se utiliza ANTLR4 para generar clases y el modelo MOF para el desarrollo dirigido por modelos. La

herramienta SODRA se proyecta como una herramienta sólida para el diseño y generación de proyectos dirigida a modelos. Sin embargo, se reconoce que el metamodelo está limitado al nivel de abstracción y detalle del diseñador y es necesario complementarlo para obtener un nivel de detalle más específico. En la etapa cuatro se realizaron adiciones a la herramienta para incluir diagramas del cliente y navegacional.

### **Recomendaciones**

En la herramienta generadora de metamodelos MOF, se destaca la capacidad de agregar nuevos módulos para la generación de modelos y código en diferentes lenguajes de programación. Sin embargo, es importante tener en cuenta que la interpretación del usuario al crear el modelo es una limitación de la herramienta. Por esta razón, se propone la estandarización de SODRA para regular su proceso.

La estandarización es fundamental para el proceso MDA, y por lo tanto, se considera la integración de SODRA al Grupo de Perfiles Genéricos, que es aplicable a POs que no desarrollan software crítico sino software listo para ser utilizado. El objetivo es adaptar el proceso MDA y la herramienta SODRA al ISO/IEC 29110-5-1-1:2012 y sus dos paquetes de despliegue (Gestión de Proyectos e Implementación de Software) para brindar apoyo a las empresas mexicanas que deseen implementar la norma ISO 29110.

En conclusión, el uso de sistemas basados en modelos y herramientas como SODRA puede ser beneficioso para la automatización y agilización de los procesos de desarrollo de software. La capacidad de SODRA de generar metamodelos MOF y añadir módulos para generar otros modelos o generación de código en otros lenguajes es un aspecto relevante que puede facilitar el trabajo del programador y reducir costos y tiempos de desarrollo. Sin embargo, la interpretación del usuario al realizar el modelo es una limitación importante que se debe tener en cuenta y se propone la estandarización de SODRA para regular su proceso y adaptarla a la norma ISO 29110. Con una mayor estandarización, SODRA puede proporcionar un apoyo importante a las empresas mexicanas interesadas en la implementación de esta norma. En general, el desarrollo dirigido

por modelos y herramientas como SODRA pueden ser una opción valiosa para mejorar la eficiencia y calidad del proceso de desarrollo de software.

## Referencias

- [1] F. Carreón-Díaz de León, «SODRA Arquitectura de una herramienta para crear diagramas del modelo cliente y navegacional», *Desarro. E Implementación Las Cienc. Comput.*, pp. 104-111, 2019.
- [2] «Congreso Internacional de Investigacion Tijuana», jun. 30, 2020. [http://ci2tijuana.org/virtual\\_conference\\_cocsce.php](http://ci2tijuana.org/virtual_conference_cocsce.php) (accedido jun. 30, 2020).
- [3] V. K. Balakrishnan, «Graph Theory».
- [4] J. L. Gross y J. Yellen, *Handbook of Graph Theory*, 2nd Edition. Chapman and Hall/CRC, 2013. Accedido: may 31, 2021. [En línea]. Disponible en: <https://www.routledge.com/Handbook-of-Graph-Theory/Gross-Yellen-Zhang/p/book/9781439880180>
- [5] Booch, Grady, rumbaugh, James, y jacobson, Ivar, *The Unified Modeling Language User Guide . Covers Uml 2.0, 2ª*. Addison-Wesley Professional, 2005. Accedido: ene. 27, 2020. [En línea]. Disponible en: [https://www.todostuslibros.com/libros/the-unified-modeling-language-user-guide-covers-uml-2-0-2a-ed\\_978-0-321-26797-9](https://www.todostuslibros.com/libros/the-unified-modeling-language-user-guide-covers-uml-2-0-2a-ed_978-0-321-26797-9)
- [6] R. S. Pressman, *Ingenieria del Software, Un Enfoque Practico*, Quinta Edicion. McGraw-Hill, 2002. Accedido: ene. 26, 2020. [En línea]. Disponible en: [https://www.academia.edu/7365760/Ing\\_Soft\\_Pressman\\_Quinta\\_Ed](https://www.academia.edu/7365760/Ing_Soft_Pressman_Quinta_Ed)
- [7] C. D. L. Gámez, S. G. Peláez-Camarena, U. Juárez-Martínez, M. A. Abud-Figueroa, y C. R. Torres, «Propuesta de artefactos basados en una notación con grafos y conjuntos para el modelado conceptual de aplicaciones Web», *Res. Comput. Sci.*, vol. 107, pp. 41-50, 2015.
- [8] J. F. Carreón Díaz de León y S. G. S. Peláez Camarena, «DESARROLLO DE UNA HERRAMIENTA PARA LA GENERACIÓN DE DIAGRAMAS DEL MODELO CLIENTE Y NAVEGACIONAL BASADA EN ARTEFACTO CON GRAFOS Y TEORÍA DE CONJUNTOS», Thesis, 2018. Accedido: may 31, 2021. [En línea]. Disponible en: <http://repositorios.orizaba.tecnm.mx:8080/xmlui/handle/123456789/445>
- [9] Ludwig-Maximilians-Universität München, «UWE - Tutorial - Navigation Model (Español)», *UWE – UML-based Web Engineering*, ago. 10, 2016. <http://uwe.pst.ifi.lmu.de/teachingTutorialNavigationSpanish.html> (accedido ene. 27, 2020).
- [10] C. D. L. Gámez, S. G. Peláez-Camarena, U. Juárez-Martínez, M. A. Abud-Figueroa, y C. R. Torres, «Propuesta de artefactos basados en una notación con grafos y conjuntos

- para el modelado conceptual de aplicaciones Web», *Res. Comput. Sci.*, vol. 107, pp. 41-50, 2015.
- [11]B. M. Duc, *Real-Time Object Uniform Design Methodology with UML*. Springer Netherlands, 2007. doi: 10.1007/978-1-4020-5977-3.
- [12]T. Baar, A. Strohmeier, y A. Moreira, Eds., *UML 2004 - The Unified Modeling Language: Modeling Languages and Applications. 7th International Conference, Lisbon, Portugal, October 11-15, 2004. Proceedings*. Berlin Heidelberg: Springer-Verlag, 2004. doi: 10.1007/b101232.
- [13]Object Management Group, «Meta Object Facility», *ABOUT THE META OBJECT FACILITY SPECIFICATION VERSION 2.4.2*, abr. 2014. <https://www.omg.org/spec/MOF/2.4.2/About-MOF/> (accedido ene. 27, 2020).
- [14]Object Management Group, «XML Metadata Interchange», *ABOUT THE XML METADATA INTERCHANGE SPECIFICATION VERSION 2.5.1*, jun. 2015. <https://www.omg.org/spec/XMI/About-XMI/> (accedido ene. 27, 2020).
- [15]«QVTo - Eclipsepedia», abr. 29, 2020. <https://wiki.eclipse.org/QVTo> (accedido abr. 29, 2020).
- [16]F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev, y P. Valduriez, *ATL: a QVT-like transformation language.*, vol. 2006. 2006. doi: 10.1145/1176617.1176691.
- [17]T. Parr, *The definitive ANTLR 4 reference*. Dallas, Texas: The Pragmatic Bookshelf, 2012.
- [18]T. Parr, *Language implementation patterns: create your own domain-specific and general programming languages*. Raleigh, N.C: Pragmatic Bookshelf, 2010.
- [19]«¿Qué es Java y para qué es necesario?» [https://www.java.com/es/download/help/whatis\\_java.html](https://www.java.com/es/download/help/whatis_java.html) (accedido may 31, 2021).
- [20]J. G. M. Mengerink, J. Noten, y A. Serebrenik, «Empowering OCL research: a large-scale corpus of open-source data from GitHub», *Empir. Softw. Eng.*, vol. 24, n.º 3, Art. n.º 3, jun. 2019, doi: 10.1007/s10664-018-9641-6.
- [21]D. Varró, G. Bergmann, Á. Hegedüs, Á. Horváth, I. Ráth, y Z. Ujhelyi, «Road to a reactive and incremental model transformation platform: three generations of the VIATRA framework», *Softw. Syst. Model.*, vol. 15, n.º 3, Art. n.º 3, jul. 2016, doi: 10.1007/s10270-016-0530-4.
- [22]H. Washizaki, Y.-G. Guéhéneuc, y F. Khomh, «ProMeTA: a taxonomy for program metamodels in program reverse engineering», *Empir. Softw. Eng.*, vol. 23, n.º 4, Art. n.º 4, ago. 2018, doi: 10.1007/s10664-017-9592-3.

- [23]R. Hebig, C. Seidl, T. Berger, J. K. Pedersen, y A. Wąsowski, «Model transformation languages under a magnifying glass: a controlled experiment with Xtend, ATL, and QVT», en *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Lake Buena Vista, FL, USA, oct. 2018, pp. 445-455. doi: 10.1145/3236024.3236046.
- [24]O. Semeráth, A. A. Babikian, S. Pilarski, y D. Varró, «VIATRA Solver: A Framework for the Automated Generation of Consistent Domain-Specific Models», en *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, may 2019, pp. 43-46. doi: 10.1109/ICSE-Companion.2019.00034.
- [25]A. Bernal, M. E. Cambronero, A. Núñez, P. C. Cañizares, y V. Valero, «Improving cloud architectures using UML profiles and M2T transformation techniques», *J. Supercomput.*, vol. 75, n.º 12, Art. n.º 12, dic. 2019, doi: 10.1007/s11227-019-02980-w.
- [26]A. Bucchiarone, J. Cabot, R. F. Paige, y A. Pierantonio, «Grand challenges in model-driven engineering: an analysis of the state of the research», *Softw. Syst. Model.*, vol. 19, n.º 1, Art. n.º 1, ene. 2020, doi: 10.1007/s10270-019-00773-6.
- [27]«About the XML Metadata Interchange Specification Version 2.1». <https://www.omg.org/spec/XMI/2.1> (accedido ene. 18, 2021).
- [28]«About the XML Metadata Interchange Specification Version 2.5.1». <https://www.omg.org/spec/XMI/> (accedido ene. 18, 2021).
- [29]S. E. Gámez, «Desarrollo de un generador de Aplicaciones Enriquecidas de Internet modeladas bajo el patrón arquitectónico MVC usando UML e IFML», p. 147.
- [30]I. Sommerville, *Ingeniería del software*. Pearson Educación, 2005.
- [31]A. V. Aho, *Compiladores: principios, técnicas y herramientas*. Mexico: Pearson Educación, 2008. Accedido: mar. 08, 2021. [En línea]. Disponible en: <http://www.pearsonbv.com/integracionIP/?stisbn=9789702611301>
- [32]S. E. Cruz Sánchez y S. G. S. Peláez Camarena, «DESARROLLO DEL MÓDULO PARA LA GENERACIÓN DE CÓDIGO EN LENGUAJE PHP, PARA LA HERRAMIENTA SODRA A PARTIR DEL INTERMEDIARIO XMI DE LOS MODELOS DEL CLIENTE Y NAVEGACIONAL», Thesis, 2020. Accedido: may 26, 2021. [En línea]. Disponible en: <http://repositorios.orizaba.tecnm.mx:8080/xmlui/handle/123456789/442>
- [33]«Fielding Dissertation: CHAPTER 5: Representational State Transfer (REST)». [https://www.ics.uci.edu/~fielding/pubs/dissertation/rest\\_arch\\_style.htm](https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm) (accedido may 31, 2021).

- [34]C. Cuevas, L. Barros, P. L. Martínez, y J. M. Drake, «Beneficios que aporta la metodología MDE a los entornos de desarrollo de sistemas de tiempo real», *Rev. Iberoam. Automática E Informática Ind. RIAI*, vol. 10, n.º 2, pp. 216-227, abr. 2013, doi: 10.1016/j.riai.2013.03.011.
- [35]«Professor Claude Y. Laporte, Eng., Ph.D.»  
<http://profs.etsmtl.ca/claporte/English/VSE/indexS.html> (accedido mar. 09, 2021).