TRABAJO DE FIN DE GRADO

TRANSFORMACIÓN DE LAS RESTRICCIONES OCL DE UN ESQUEMA UML A CONSULTAS DE SQL

UNIVERSIDAD CARLOS III DE MADRID
Grado en Ingeniería Informática
Curso 2011/2012

Tutor: Harith Aljumaily
Autor: Sergio Casillas Cortázar

Fecha de Entrega: 05 de septiembre de 2012

Contenido

Índices de tablas

Índices de ilustraciones

1. Introducción

En este primer capítulo se realiza una breve descripción sobre el mismo, indicando cual es el problema tratado en este Trabajo de Fin de Grado y los objetivos a los que se quieren llegar a su conclusión.

Al final de este apartado se mostrara la estructura de la memoria del Trabajo de Fin de Grado, indicando de los puntos de los que consta y una breve introducción a cada uno de ellos.

1.1. Presentación del problema

En la actualidad existen muchos métodos y herramientas que permiten la generación automática y completa del código de una aplicación a partir de un diagrama en UML. El lenguaje UML (*Unified Modeling Language, Lenguaje Unificado de Modelado)* [1] es un lenguaje visual cuyo uso más extendido es documentar, construir y especificar los modelos del sistema.

Todas estas herramientas son capaces de generar clases en Java o tablas en un Sistema de Gestión de Bases de Datos (*DBMS*, *DataBase Management System*) a partir de un diagrama de clases UML. El problema surge debido a que la mayoría de estas herramientas tienen poca consideración sobre las restricciones de integridad, a pesar que según se define en [2] las restricciones de integridad son una parte fundamental de la especificación de una aplicación y por lo tanto tienen que tenerse en cuenta durante su implementación.

En base a esto, el lenguaje OCL (*Object Constraint Language, Lenguaje de Especificación de Objetos*) [3] es un lenguaje conceptual que complementa al UML y cuyo uso más extendido es el que se da en la definición de restricciones libres de efectos colaterales para el diagrama de clases.

Por todo lo anterior, se ha decidido crear una herramienta para ayudar a los desarrolladores de bases de datos y que sea capaz de transformar las restricciones de integridad realizadas en OCL a consultas SQL Standard que comprueben si esas restricciones se cumplen o no. Como las transformaciones

no son triviales se va a necesitar un programador experto que sea capaz de diseñar las reglas que permitan dicha transformación.

Las consultas generadas contendrán los datos que no cumplen la restricción, por lo tanto si al ejecutar esta consulta obtenemos datos significará que la restricción no se está cumpliendo. En el caso de que no se devuelvan datos significa que la base de datos cumple la restricción.

1.2. Objetivos

Una vez se ha explicado el motivo sobre el porqué de este Trabajo de Fin de Grado, se exponen los objetivos que se han fijado para su realización:

- Estudio del ámbito de las tecnologías en las que se desarrolla el Trabajo de Fin de Grado
    - **SQL Standard** [4], comúnmente llamado SQL (*Structured Query Language, Lenguaje de Consulta Estructurado*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar operaciones sobre éstas.
    - **OCL** es un lenguaje formal que sirve para expresar restricciones libres de efectos colaterales. Se utiliza para definir restricciones de integridad en el diagrama de clases.
- Análisis de las herramientas actuales que permitirá conocer la situación actual del mercado respecto a las herramientas que son capaces de transformar sentencias OCL en SQL. Se evalúa si ya existe alguna que satisfaga dichas necesidades o si existe alguna de código libre a partir de la cual podamos partir.
- Diseñar reglas de transformación. En esta fase se procederá a analizar las sentencias OCL y especificar cuál sería su transformación a consultas SQL.
- Implementar una herramienta que sea capaz de aplicar las reglas de transformación sobre las restricciones definidas en el diagrama de clases.
- Realizar pruebas de validación que permitan comprobar el correcto funcionamiento de la herramienta implementada.

1.3. Estructura del documento

El resto del documento está estructurado en los siguientes capítulos:

Capítulo 2: Estado del arte

Se realiza un análisis de las restricciones de integridad y se estudian las herramientas similares a la desarrollada que existen en el mercado.

Capítulo 3: Gestión del proyecto

Se expone todo lo relacionado con la gestión del proyecto, entre lo que se encuentra la metodología seguida, los diagramas de planificación y los hitos que se han seguido para el desarrollo del proyecto.

Capítulo 4: Análisis del sistema

Se elabora un estudio del problema que se quiere resolver y se define la solución propuesta. Se exponen los requisitos de usuario y de software de la herramienta que se pretende desarrollar.

Capítulo 5: Propuesta de transformar OCL a SQL

Se analiza el ámbito de la tecnología utilizada y se hace un profundo análisis a las distintas sentencias OCL que existen y su transformación a SQL Standard.

Capítulo 6: Diseño e implementación

A partir de los requisitos de software definidos anteriormente, se propone la arquitectura que va a seguir el sistema y se especifican los componentes. En este capítulo también se índica la manera seguida para la implementación de la herramienta.

Capítulo 7: Pruebas

Plan de pruebas realizado para comprobar el correcto funcionamiento de la herramienta.

Capítulo 8: Presupuesto

Desarrollo del coste que ha supuesto la creación de la herramienta desglosado en varios tipos de coste y las fases realizadas.

Capítulo 9: Conclusión

Para finalizar la memoria del proyecto se comentan las conclusiones que se han llegado y los posibles trabajos futuros.

Capítulo 10: Bibliografía

Listado de referencias bibliográficas ordenadas por orden de aparición en el documento.

Anexo 1: Acrónimos y abreviaturas

Listado de los acrónimos y abreviaturas utilizados en todo el documento ordenados alfabéticamente.

Anexo 2: Manual de usuario

Se indican los pasos necesarios para la comprensión de la herramienta y la realización de transformaciones.

2. Estado del arte

En este apartado se analizan las restricciones de integridad que es la base para el desarrollo de la herramienta. También se analizaran las distintas herramientas existentes en el mercado y el motivo por el cual no nos sirven como solución a nuestro problema.

2.1. Restricciones de integridad

Las restricciones de integridad proporcionan un medio para asegurar que la información contenida en la bases de datos cumple ciertas restricciones y que en el caso de que se produzcan modificaciones, éstas no provoquen la pérdida de la consistencia en los datos. Por tanto, las restricciones de integridad aseguran que la información contenida en una base de datos es correcta.

La evolución en el tiempo de una base de datos puede definirse por un diagrama de estados:

$$E_0 \longrightarrow T_1 \longrightarrow E_0 \longrightarrow \ldots \longrightarrow T_n \longrightarrow E_n$$

donde E_n es el estado actual de la base de datos y T_n es la transacción que permite cambiar el estado de una base de datos.

En base al diagrama de estados de la base de datos existen dos tipos de restricciones:

- Restricciones estáticas: hacen referencia a un único estado de la base de datos. Estas restricciones limitan los estados válidos con independencia de la secuencia de los mismos.
- Restricciones dinámicas: hacen referencia a dos o más estados de la base de datos. Estas restricciones limitan las secuencias de estados válidas.

Como en este Trabajo de Fin de Grado solo vamos a transformar las restricciones de integridad estáticas, se procede a analizarlas en detalle. Para realizar un correcto análisis se tendrá en cuenta si la restricción se especifica en el diagrama de clases o mediante sentencias OCL.

2.1.1. Restricciones definidas mediante el diagrama de clases

Las restricciones que se pueden formular mediante el diagrama de clases son las restricciones que se producen en las relaciones entre clases, que son las que hacen posibles que dos o más clases se pueden interrelacionar, cada una con características y objetivos diferentes.

A continuación se explican los tipos de relaciones que existen.

2.1.1.1. Generalización

Una generalización especifica una clase general en otra más específica. Cada instancia de la clase específica es también instancia de la clase general: tiene las características (propiedades, relaciones, métodos) de la clase general además de las características de su propia clase.

Se representa de la siguiente manera:

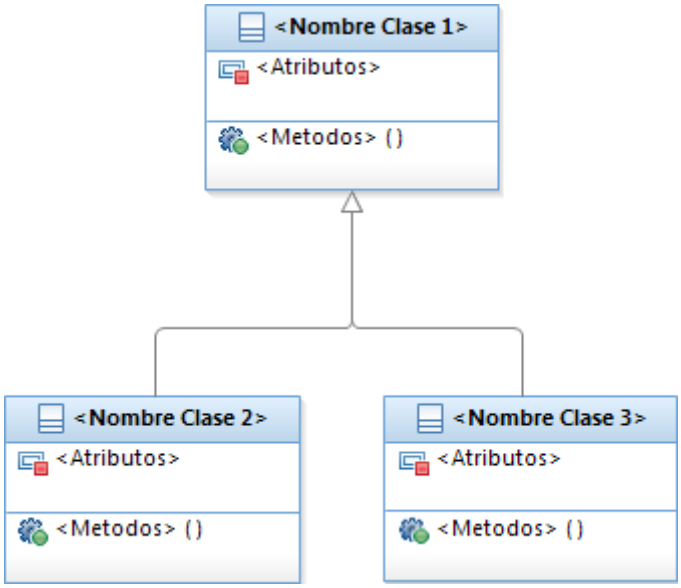


Ilustración 1 - Generalización entre clases

Existen cuatro características que definen los tipos de generalizaciones que existen, estas son:

- total: la generalización ya no permite más hijos.
- partial: se pueden incorporar más tipos a la generalización.
- disjoint: solo puede tener un tipo en tiempo de ejecución, una instancia padre solo podrá ser un tipo de hijo.

- overlapping: puede cambiar de tipo durante su vida, una instancia del padre puede ir cambiando de tipo entre los hijos.

Estas cuatro características nos permiten definir cuatro tipos de generalizaciones: total-disjoint, partial-disjoint, total-overlapping y partial-overlapping, pero solo la generalización total-disjoint es la que se va a tener en cuenta.

2.1.1.2. Asociación

Una asociación permite asociar clases de manera estructural que colaboran entre sí. Se asume que una asociación es bidireccional, es decir, que se puede navegar desde cualquiera de las clases implicadas a la otra, pero es posible indicar que la navegación solo pueda existir en una sola dirección. A continuación se exponen ejemplos de ambos casos.

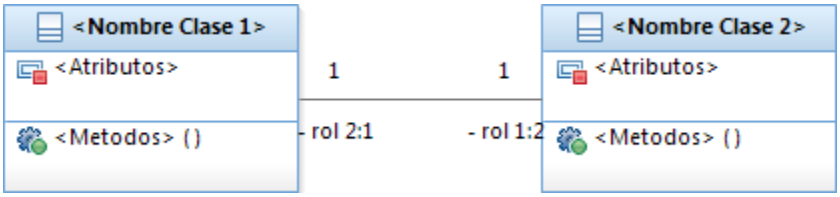


Ilustración 2 - Asociación entre clases bidireccional

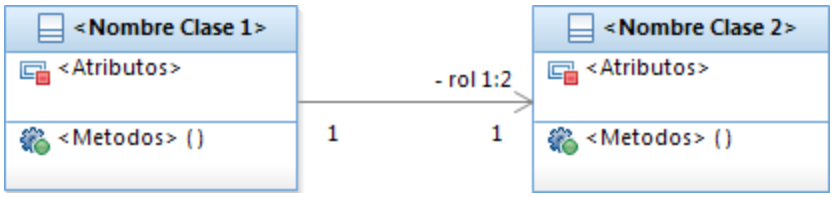


Ilustración 3 - Asociación entre clases una dirección

En estas imágenes podemos comprobar que existen el campo "*rol 2:1*" y el "*rol 1:2*". Este campo identifica el rol, que sirve para especificar el papel que juega cada clase dentro de la transformación. "*1*" es la multiplicidad de la relación que será explicada a continuación junto con el grado de la relación, que son las características que tienen las relaciones entre clases.

Grado de una relación

El grado de relación es el número del conjunto de clases que participan en la asociación y se pueden clasificar de la siguiente manera.

Unaria: participa una única clase.

Binaria: participan dos clases.

N-aria: participan más de dos conjuntos de entidades.

En este Trabajo de Fin de Grado el segundo caso, binaria, y el tercero, n-aria, se tratan como si fuera el mismo caso.

Multiplicidad de relaciones

Aplicado a un diagrama de clase, que es donde se definen este tipo de sentencias, la multiplicidad de relaciones indica el grado y nivel de dependencia entre las clases. Se muestra en cada uno de los extremos de la relación, al igual que el rol. La multiplicidad puede ser:

Uno a uno. 1:1. Una ocurrencia de la clase <Nombre Clase 1> solo se puede relacionar con una ocurrencia la clase <Nombre Clase 2>, y viceversa.

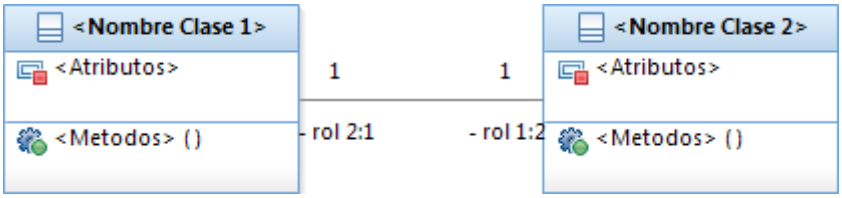


Ilustración 4 - Relación 1:1

Uno a muchos. 1:n. Significa que una ocurrencia de la clase <Nombre Clase 1> puede relacionarse con n ocurrencias de la clase <Nombre Clase 2>, y que una ocurrencia de la clase <Nombre Clase 2> solo pueda estar relacionada con una ocurrencia de la clase <Nombre Clase 1>.

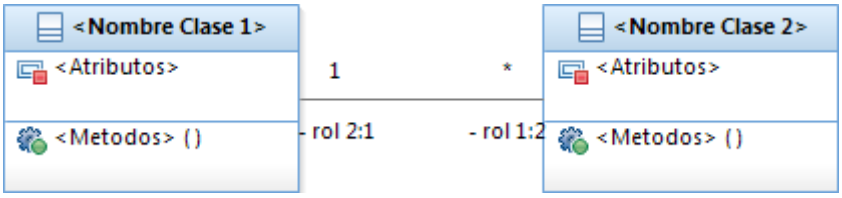


Ilustración 5 - Relación 1:n

Muchos a uno. n:1. Significa que una ocurrencia de la clase <Nombre Clase 2> puede relacionarse con n ocurrencias de la clase <Nombre Clase 1>, y una ocurrencia de la clase <Nombre Clase 1> solo puede estar relacionada con una ocurrencia de la clase <Nombre Clase 2>.

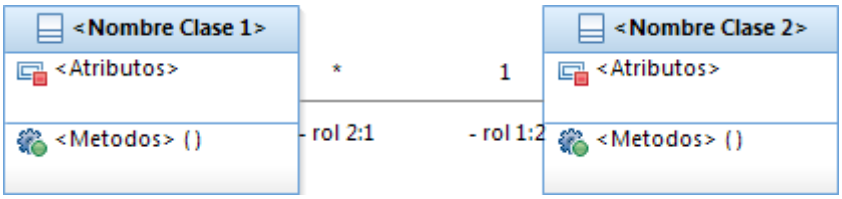


Ilustración 6 - Relación n:1

Muchos a muchos. n:n. Significa que una ocurrencia de la clase <Nombre Clase 1> se puede relacionarse con n ocurrencias de la clase <Nombre Clase 2>, y viceversa.

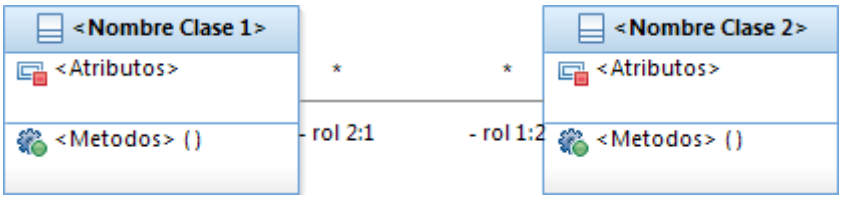


Ilustración 7 - Relación n:n

- **Agregación**

La agregación es un caso particular de la asociación. La agregación es una relación dentro de la cual una o más clases son partes de un conjunto. Se representa de la siguiente manera:

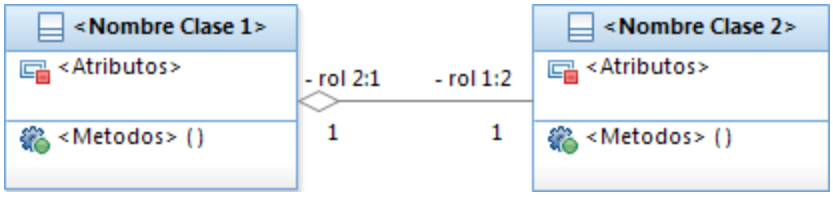


Ilustración 8 - Agregación entre clases

En este caso la clase <*Nombre Clase 2*> seria una parte del conjunto que forma la clase <*Nombre Clase 1*>.

- **Composición**

La composición es un tipo especial de la agregación no compartida. La diferencia con la agregación es que tanto el conjunto como las partes tienen el mismo ciclo de vida. Se representa de la siguiente manera:

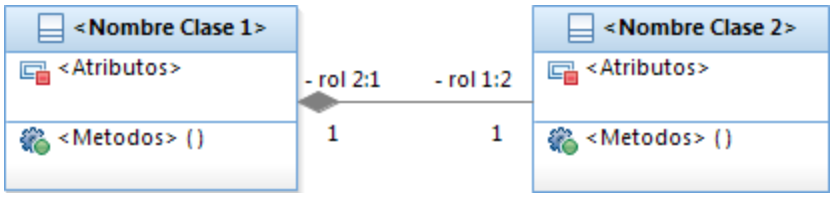


Ilustración 9 - Composición entre clases

En este Trabajo de Fin de Grado solo se van a tener las asociaciones y no sus casos particulares como la agregación y la composición.

2.1.2. Restricciones de integridad definidas mediante sentencias OCL

Las restricciones que se pueden formular mediante las sentencias OCL son aquéllas que mediante el diagrama de clases no son posibles de definir. Aunque existen distintos tipos de sentencias OCL: invariantes, pre condición, post condición y otros tipos de restricción; en este Trabajo de Fin de Grado solo se van a tener en cuenta las invariantes porque son las que representan las restricciones estáticas.

A continuación analizamos los distintos tipos de restricciones estáticas que podemos realizar mediante las sentencias invariantes.

- **Concepto de clave primaria**

Determina que cada objeto tiene que identificarse de forma unívocamente, diferenciando cada objeto del resto de objetos del mismo tipo. Para ello cada objeto debe ser identificable por un determinado atributo.

Ejemplo:

"El objeto Publicación se identifica por su atributo de clave primaria id_pub".

Formalmente se definiría como:

$$\neg\exists t1\ \exists t2\ (Publicación(t1) \wedge Publicación(t2) \wedge t1 \neq t2 \wedge t1.id\_pub = t2.id\_pub) \wedge \forall t\ (Publicación(t) \rightarrow \neg nulo(t.id\_pub))$$

- **Restricciones de Unicidad**

Estas restricciones determinan que si tenemos un objeto con varios atributos y varias filas con datos, no pueden existir dos filas iguales, es decir, que dos filas no pueden tener todos sus atributos iguales.

Ejemplo:

"En el objeto Publicación no puede haber dos periódicos que tengan el mismo título".

Formalmente se definiría como:

$$\neg\exists t1\ \exists t2\ (Publicación(t1) \wedge Publicación(t2) \wedge t1 \neq t2 \wedge t1.título = t2.título \wedge \neg nulo(t1.título) \wedge \neg nulo(t2.título))$$

- **Concepto de clave ajena**

Determina que si un atributo A1 es el inverso de otro atributo A2, implica que, si el objeto O2 es un valor del atributo A1 de algún objeto O1 entonces el objeto O1 es un valor del atributo A2 para el objeto O2.

Ejemplo:

"El objeto Publicación tiene un atributo autor_id que es clave ajena de el objeto Autor"

Si fuera sobre una base de datos:

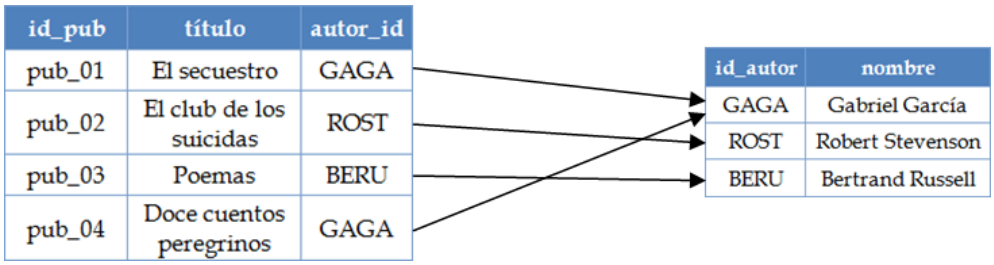


Ilustración 10 - Ejemplo de clave ajena sobre base de datos

- **Restricciones de valor no nulo**

Determina que atributos no pueden contener un valor nulo para cada objeto.

Ejemplo:

"En el objeto Publicación el atributo título no puede contener ningún valor nulo".

Formalmente se definiría como:

$$\forall t\ (Publicación(t) \rightarrow \neg nulo(t.Título))$$

- **Restricciones sobre una clase**

Estas restricciones permiten hacer una limitación sobre cualquier atributo o atributos que pertenezca a una misma clase. Entre estas restricciones también se encuentran aquellas en las que se instancia dos veces la misma clase y se comparan los atributos de ambas instancias.

- **Restricciones con navegación**

Estas restricciones permiten navegar entre las clases para poder realizar restricciones sobre sus atributos. La navegación entre clases se hace con el rol de la asociación. En este tipo de restricciones también se encuentran aquellas restricciones que después de realizar la navegación se les aplica una función [5] sobre los atributos o la clase hasta la que se ha navegado.

2.2. Trabajos relacionados

A continuación se relacionan las herramientas similares a la desarrollada que existían en el momento del análisis del estado del arte. También se explica el motivo por el cual estas herramientas no se adaptan a las necesidades requeridas.

- **Dresden OCL**

Dresden OCL [6] es la más conocida. Permite la generación de las clases Java correspondientes al diagrama de clases. También transforma las restricciones de integridad definidas excepto el operador allInstances que no se genera correctamente.

Se ha decidido descartar esta herramienta porque las restricciones generadas a partir de las sentencias OCL no se generan sobre una base de datos relacional sino sobre una clase Java.

- **OCLtoSQL**

OCLtoSQL es una herramienta incluida en el anterior kit y basada en el método propuesto en [7]. Para cada restricción crea una vista de forma que si al finalizar

una transacción la vista no está vacía significa que la transacción ha violado la restricción representada por la vista.

Esta herramienta sería la que buscamos de no ser porque la generación de sentencias es muy ineficiente y en bastantes casos no se obtiene el resultado esperado.

- **Boldsoft**

BoldSoft [8] es una herramienta pensada para la definición de elementos derivados en vez de para la definición de restricciones de integridad que es el principal objetivo. Además esta herramienta no permite operaciones sobre colecciones muy utilizadas como *count*, *collect*, *asSet* y *asBag*.

- **MySQL4OCL**

MySQL4OCL [9] es un compilador que permite la evaluación automática de expresiones OCL sobre una RDB *(Relational DataBase, Base de Datos Relacional)* ya definida [10].
Esta herramienta ha sido descartada porque se basa en una RBD ya definida que no se puede modificar y además transforma las sentencias OCL a procedimientos y no a consultas.

- **OCL2Trigger**

OCL2Trigger [11] es una herramienta que transforma sentencias OCL a disparadores. Es una herramienta muy básica que solo cubre tres tipos de restricciones: sobre un valor, multiplicidad y generalización. Se ha descartado porque esta herramienta no permite ninguna operación sobre colecciones.

3.Gestión del proyecto

En este capítulo se analizan y se muestran las distintas fases e hitos por los que ha transcurrido la planificación del proyecto.

3.1. Gestión del proyecto software

En este primer apartado se ha decido cuales van a ser las principales fases del desarrollo del proyecto. En la ilustración que se muestra a continuación se expone el ciclo de vida del proyecto.

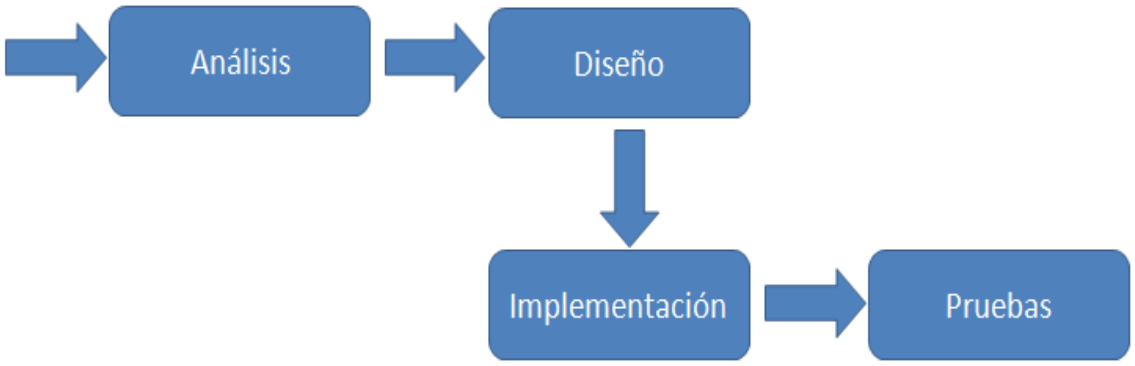


Ilustración 11 - Ciclo de vida del proyecto

Durante la fase de implementación se irán desarrollando pruebas parciales del desarrollo. Al finalizar la implementación se realizarán pruebas que permitirán comprobar el correcto funcionamiento de la herramienta desarrollada.

3.2. Metodología

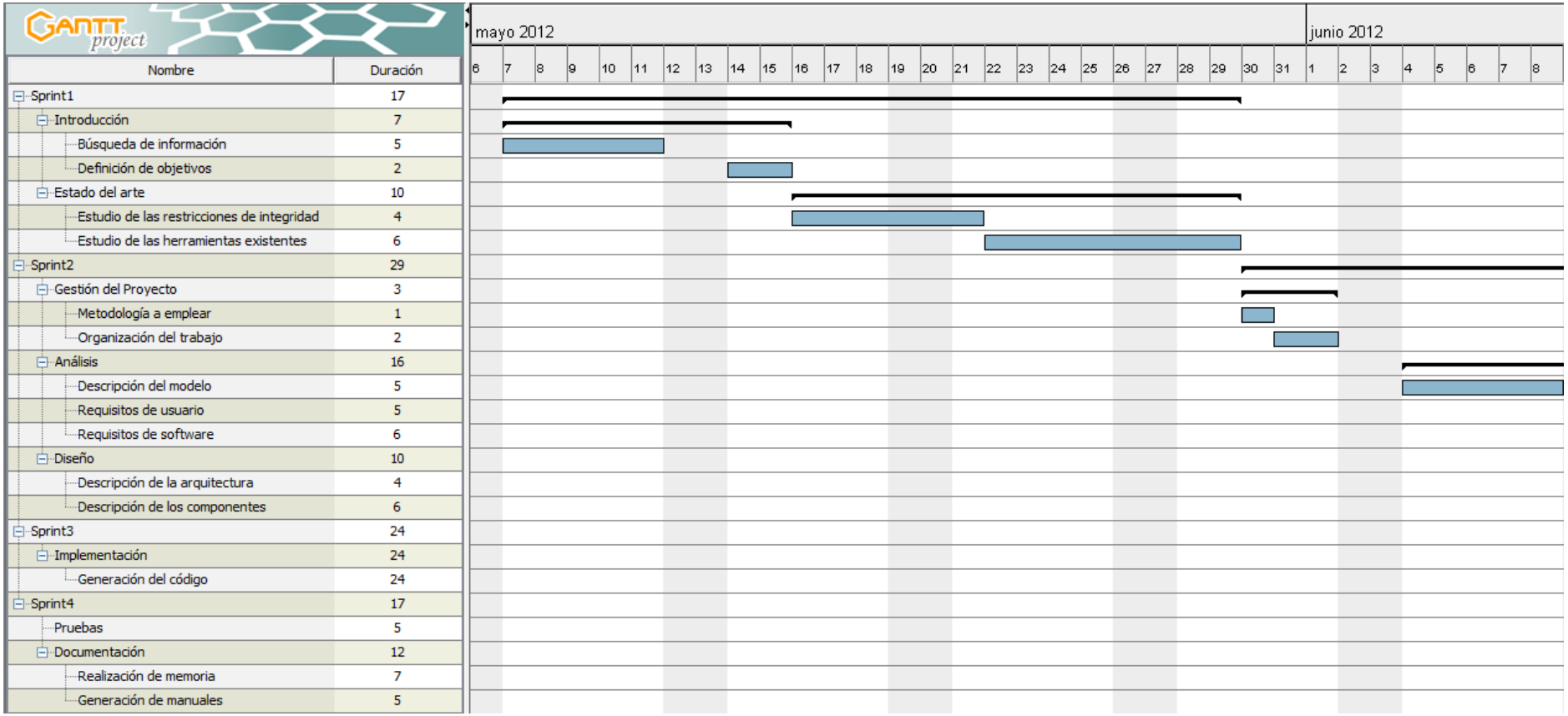
Para el desarrollo de esta herramienta se utiliza una adaptación de la metodología Scrum. Esta metodología consiste en desarrollar un proyecto mediante sprints que permiten dividir y evaluar de manera independiente el mismo. La duración de los sprints y las tareas que se deben realizar en los mismos aparecerán definidas en el diagrama de Gantt.

3.3. Organización del trabajo

En este apartando se exponen las distintas tareas y subtareas que han constituido este Trabajo de Fin de Grado. Para ello, se ha utilizado un diagrama de Gantt que permite ver la duración de estas tareas a lo largo del tiempo. También se exponen las tareas y subtareas que va a contener cada uno de los sprints

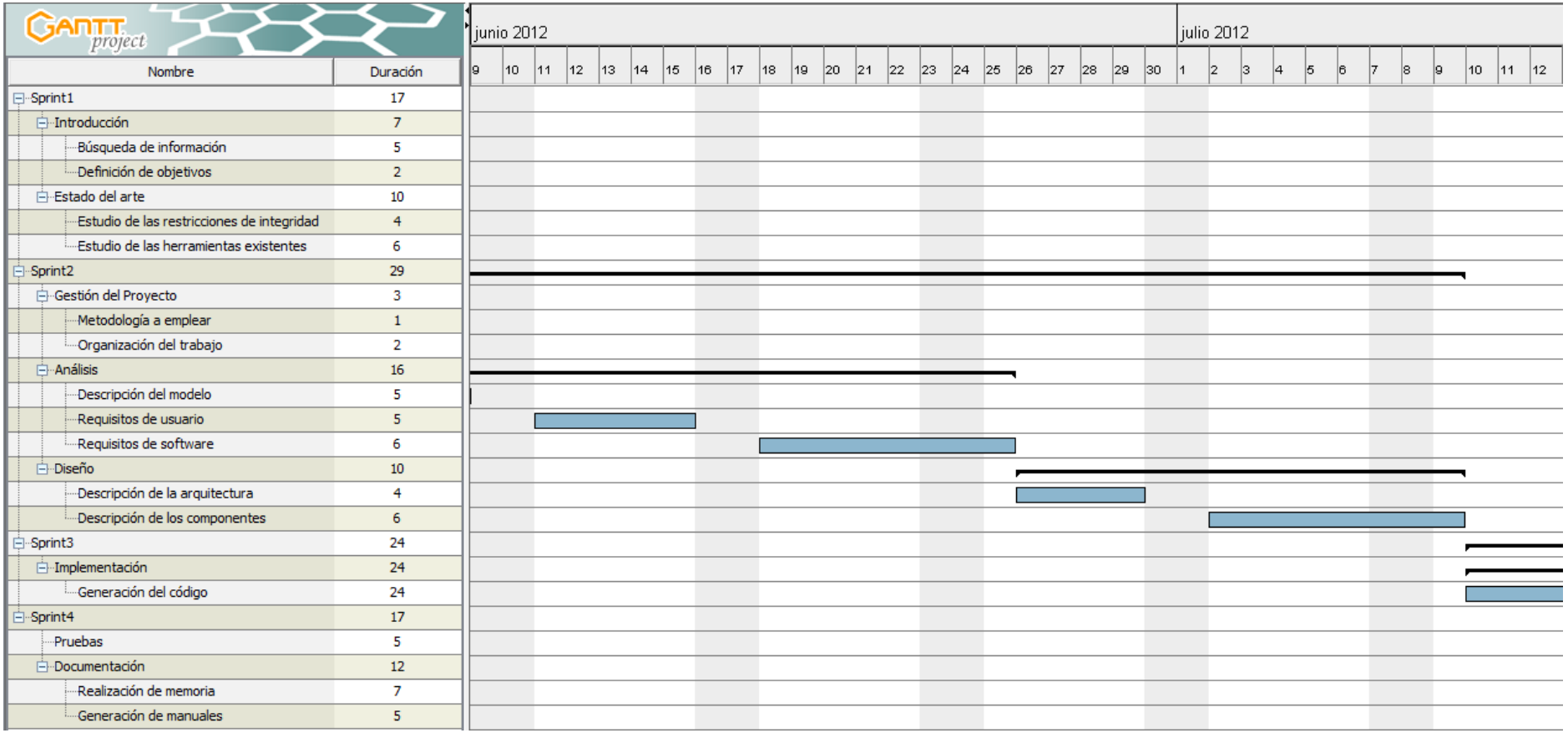
- **Sprint 1**
    - Introducción
        - Búsqueda de información
        - Definición de objetivos
    - Estado del arte
        - Estudio de las restricciones de integridad
        - Estudio de las herramientas existentes
- **Sprint 2**
    - Gestión del proyecto
        - Metodología a emplear
        - Organización del trabajo
    - Análisis
        - Descripción del modelo
        - Requisitos de usuario
        - Requisitos de software
    - Diseño
        - Descripción de la arquitectura
        - Descripción de los componentes
- **Sprint 3**
    - Implementación
        - Generación del código
    - Pruebas
- **Sprint 4**
    - Documentación
        - Realización de la memoria
        - Generación de manuales

A continuación se muestra el diagrama de Gantt con la programación seguida a lo largo de este Trabajo de Fin de Grado. Cabe indicar que la duración de cada tarea se expresa en días donde la media diaria de horas empleadas es de cuatro.



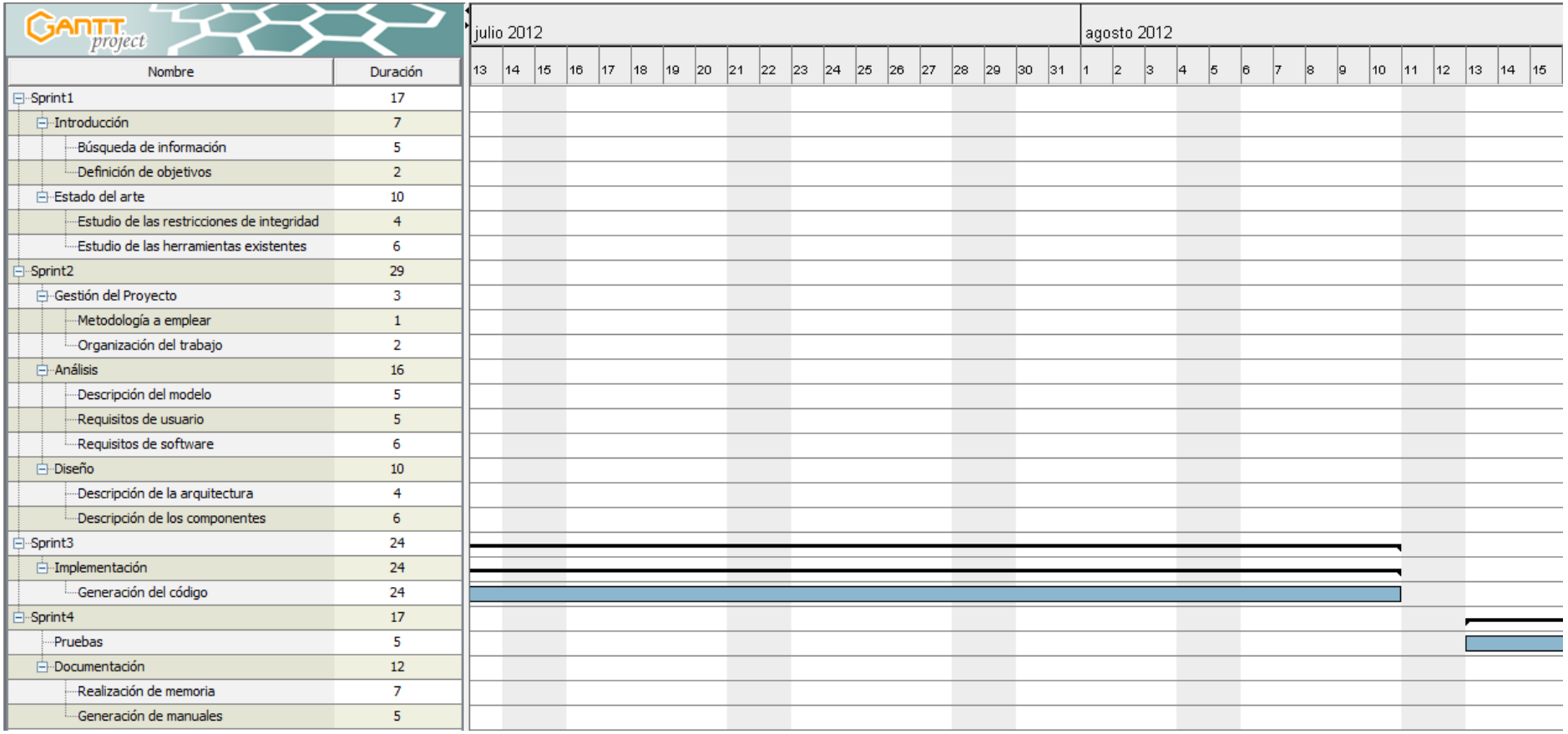
Nombre	Duración
Sprint1	17
Introducción	7
Búsqueda de información	5
Definición de objetivos	2
Estado del arte	10
Estudio de las restricciones de integridad	4
Estudio de las herramientas existentes	6
Sprint2	29
Gestión del Proyecto	3
Metodología a emplear	1
Organización del trabajo	2
Análisis	16
Descripción del modelo	5
Requisitos de usuario	5
Requisitos de software	6
Diseño	10
Descripción de la arquitectura	4
Descripción de los componentes	6
Sprint3	24
Implementación	24
Generación del código	24
Sprint4	17
Pruebas	5
Documentación	12
Realización de memoria	7
Generación de manuales	5

Ilustración 12 - Diagrama de Gantt 1



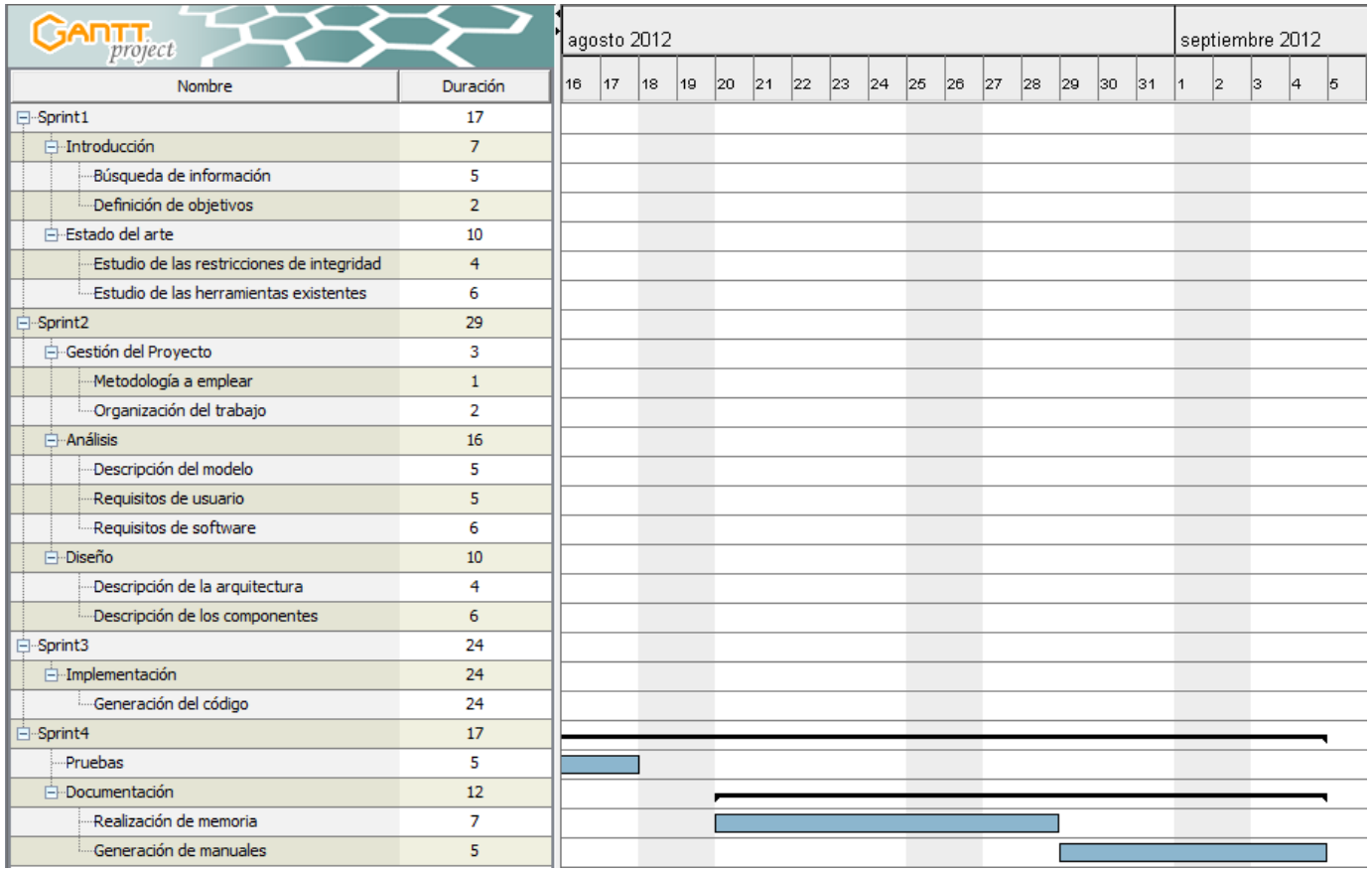
Nombre	Duración
Sprint1	17
Introducción	7
Búsqueda de información	5
Definición de objetivos	2
Estado del arte	10
Estudio de las restricciones de integridad	4
Estudio de las herramientas existentes	6
Sprint2	29
Gestión del Proyecto	3
Metodología a emplear	1
Organización del trabajo	2
Análisis	16
Descripción del modelo	5
Requisitos de usuario	5
Requisitos de software	6
Diseño	10
Descripción de la arquitectura	4
Descripción de los componentes	6
Sprint3	24
Implementación	24
Generación del código	24
Sprint4	17
Pruebas	5
Documentación	12
Realización de memoria	7
Generación de manuales	5

Ilustración 13 - Diagrama de Gantt 2



Nombre	Duración
Sprint1	17
Introducción	7
Búsqueda de información	5
Definición de objetivos	2
Estado del arte	10
Estudio de las restricciones de integridad	4
Estudio de las herramientas existentes	6
Sprint2	29
Gestión del Proyecto	3
Metodología a emplear	1
Organización del trabajo	2
Análisis	16
Descripción del modelo	5
Requisitos de usuario	5
Requisitos de software	6
Diseño	10
Descripción de la arquitectura	4
Descripción de los componentes	6
Sprint3	24
Implementación	24
Generación del código	24
Sprint4	17
Pruebas	5
Documentación	12
Realización de memoria	7
Generación de manuales	5

Ilustración 14 - Diagrama de Gantt 3



Nombre	Duración
Sprint1	17
Introducción	7
Búsqueda de información	5
Definición de objetivos	2
Estado del arte	10
Estudio de las restricciones de integridad	4
Estudio de las herramientas existentes	6
Sprint2	29
Gestión del Proyecto	3
Metodología a emplear	1
Organización del trabajo	2
Análisis	16
Descripción del modelo	5
Requisitos de usuario	5
Requisitos de software	6
Diseño	10
Descripción de la arquitectura	4
Descripción de los componentes	6
Sprint3	24
Implementación	24
Generación del código	24
Sprint4	17
Pruebas	5
Documentación	12
Realización de memoria	7
Generación de manuales	5

Ilustración 15 - Diagrama de Gantt 4

4. Análisis del sistema

En este capítulo se va a realiza un estudio del problema que se quiere resolver y se define la solución propuesta.

4.1. Planteamiento del problema

El problema del que trata este Trabajo de Fin de Grado es el diseño e implementación de una herramienta capaz de transformar sentencias de restricción de integridad estáticas en el lenguaje OCL a consulta en SQL Standard. Para ello se ha estudiado en profundidad las herramientas existentes y la tecnología actual.

Aunque existan diversas herramientas, explicadas en el punto *2.2. Herramientas encontradas*, que realizan una función similar a la que se va a desarrollar en este Trabajo de Fin de Grado, ninguna de ellas realiza la transformación deseada.

4.1.1. Propósito y funcionalidad

El sistema consiste en implementar una herramienta desde cero que sea capaz de transformar sentencias OCL que se basan en un modelo UML, en consultas SQL Standard que se basan en una Base de Datos Relacional. Las funcionalidades de esta herramienta son:

- Analizar el diagrama UML
- Validar la sentencia OCL
- Transformar la sentencia OCL a SQL Standard
- Validar la sentencia SQL Standard

Se ha decidido realizar la herramienta íntegra dado que las soluciones de código abierto existentes eran demasiado complejas y no se disponía de la API necesaria para poder modificarlas.

4.1.2. Condiciones de entorno

Para utilizar el sistema, los usuarios tendrán que tener, como mínimo, instalado en su ordenador la Version 7 Update 6 de Java Runtime Environement (JRE) [12] que permite la ejecución de la herramienta implementada.

4.2. Especificación de requisitos

Se seguirá las normas contenidas en el estándar PSS-05-0 de la ESA [13] para la definición de los requisitos.

A continuación se exponen los requisitos obtenidos después de haber realizado el análisis del sistema. Estos requisitos van desde los que recogen las características más generales (requisitos de usuario) hasta los que se especifican con un mayor grado de detalle (requisitos de software). Para terminar se muestra una matriz de trazabilidad en la cual se relacionan ambos requisitos, los de usuario con los de software, para poder saber de qué requisitos de usuario proceden los requisitos de software.

4.2.1. Requisitos de usuario

Antes de comenzar a identificar los requisitos de usuario obtenidos del análisis del sistema, se procede a identificar su notación de nombrado.

RU-Tnnn

Donde:

RU - Significa requisito de usuario.

T - Admite los siguientes valores:

C - Requisito de Capacidad.

R - Requisito de Restricción.

nnn - Es el número identificativo del requisito dentro de su tipo de requisito.

En lo referente a los atributos de cada requisito, indicamos el significado de cada uno de los atributos.

- **Fuente**: indica el origen del requisito.
- **Necesidad**: puede tener tres valores: esencial (que el requisito es imprescindible), deseable (que el requisito conviene incluirlo), opcional (que no es obligatorio incluirlo).
- **Prioridad**: orden de implementación del requisito.
- **Estabilidad**: informa de la probabilidad de cambio del requisito durante el desarrollo del sistema. Si es alta significa que es muy difícil que vaya a cambiar, y si es baja muy probable que cambie el requisito.

4.2.2. Capacidades generales

El propósito del sistema es crear una herramienta capaz de transformar restricciones estáticas en lenguaje OCL a consultas en el lenguaje SQL Standard. Comprobar que las sentencias de restricción de integridad estáticas del lenguaje OCL están correctamente formadas de acuerdo a un diagrama de clases UML y que las consultas generadas a partir de la transformación están correctamente formadas.

4.2.3. Restricciones generales

La restricción existente en el desarrollo del sistema es utilizar como mínimo la Version 7 Update 6 de Java Development Kit (JDK) [12].

4.2.3.1. Requisitos de capacidad

RU-C001	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Tiene que permitir seleccionar un archivo con el diagrama de clases UML.

Tabla 1 - RU-C001

RU-C002	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Tiene que permitir seleccionar un archivo con las sentencias de restricciones de integridad estáticas OCL.

Tabla 2 - RU-C002

RU-C003	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Tiene que validar las sentencias OCL con el diagrama de clases.

Tabla 3 - RU-C003

RU-C004	
Fuente:	Cliente
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Alta
Descripción:	Tiene que permitir modificar el archivo con el diagrama de clases UML seleccionado.

Tabla 4 - RU-C004

RU-C005	
Fuente:	Cliente
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Alta
Descripción:	Tiene que permitir modificar el archivo con las sentencias de restricciones de integridad estáticas OCL seleccionado.

Tabla 5 - RU-C005

RU-C006	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Tiene que permitir iniciar la transformación.

Tabla 6 - RU-C006

RU-C007	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Tiene que permitir cancelar la transformación.

Tabla 7 - RU-C007

RU-C008	
Fuente:	Cliente
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Alta
Descripción:	La herramienta avisará cuando el archivo de salida vaya a ser sobrescrito.

Tabla 8 - RU-C008

RU-C009	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Las transformaciones SQL-Standard deberán estar en un archivo.

Tabla 9 - RU-C009

RU-C010	
Fuente:	Cliente
Necesidad:	Opcional
Prioridad:	Baja
Estabilidad:	Media
Descripción:	Tienen que aparecer las estadísticas después de haber realizado todas las transformaciones.

Tabla 10 - RU-C010

4.2.3.2. Requisitos de restricción

RU-R001	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	La herramienta tiene que ejecutarse, como mínimo, con JRE Version 7 Update 7.

Tabla 11 - RU-R001

RU-R002	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se tiene que utilizar el estándar UML 2.2 [1] (URI: "http://schema.omg.org/spec/UML/2.2") para el archivo del diagrama de clases.

Tabla 12 - RU-R002

RU-R003	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El archivo del diagrama de clases UML tiene que estar en formato XMI [14].

Tabla 13 - RU-R003

RU-R004	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Las restricciones OCL que contiene el archivo con las restricciones estáticas tienen que cumplir el estándar OCL 2.0 [2].

Tabla 14 - RU-R004

RU-R005	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Las consultas generadas estarán en el lenguaje SQL-Standard [4].

Tabla 15 - RU-R005

RU-R006	
Fuente:	Cliente
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Media
Descripción:	El nombre del archivo con las transformaciones en SQL-Standard, será el mismo que el archivo con las restricciones estáticas OCL, pero con la extensión ".sql".

Tabla 16 - RU-R006

RU-R007	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El porcentaje de errores no controlados tiene que ser inferior al 1%, en un mínimo de 100 sentencias.

Tabla 17 - RU-R007

RU-R008	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El porcentaje de sentencias correctas tiene que ser, al menos, del 75% [15].

Tabla 18 - RU-R008

RU-R009	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Tiene que haber un archivo con el diagrama de clases UML seleccionado para poder comenzar la transformación.

Tabla 19 - RU-R009

RU-R010	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Tiene que haber un archivo con las sentencias de restricciones de integridad estáticas OCL seleccionado para poder comenzar la transformación.

Tabla 20 - RU-R010

RU-R011	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Las sentencias OCL tienen que ser restricciones de integridad estáticas.

Tabla 21 - RU-R011

RU-R012	
Fuente:	Cliente
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Media
Descripción:	En las estadísticas aparecerán las transformaciones que han acabado satisfactoriamente.

Tabla 22 - RU-R012

RU-R013	
Fuente:	Cliente
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Media
Descripción:	En las estadísticas aparecerán las transformaciones que han acabado con un fallo controlado.

Tabla 23 - RU-R013

RU-R014	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Media
Descripción:	En las estadísticas aparecerán las transformaciones que han acabado con un fallo no controlado.

Tabla 24 - RU-R014

RU-R015	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Deberá existir una documentación con el desarrollo de la herramienta.

Tabla 25 - RU-R015

RU-R016	
Fuente:	Cliente
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se deberán probar al menos 100 transformaciones de OCL a SQL-Standard.

Tabla 26 - RU-R016

4.2.4. Requisitos de software

Antes de comenzar a identificar los requisitos de software obtenidos del análisis del sistema, se procede a identificar su notación de nombrado.

RS-Tnnn

Donde:

RS - Significa requisito de software.

T - Admite los siguientes valores:

F - Requisito de Funcionalidad.

P - Requisito de Prestaciones.

I - Requisito de Interfaz.

O - Requisito de Operación.

R - Requisito de Recursos.

V - Requisito de Verificación.

A - Requisito de pruebas de Aceptación.

D - Requisito de Documentación.

nnn - Es el número identificativo del requisito dentro de su tipo de requisito.

En lo referente a los atributos de cada requisito, indicamos el significado de cada uno de los atributos.

- **Fuente**: indica el origen del requisito.
- **Necesidad**: puede tener tres valores: esencial (que el requisito es imprescindible), deseable (que el requisito conviene incluirlo), opcional (que no es obligatorio incluirlo).
- **Prioridad**: orden de implementación del requisito.
- **Estabilidad**: informa de la probabilidad de cambio del requisito durante el desarrollo del sistema. Si es alta significa que es muy difícil que vaya a cambiar, y si es baja muy probable que cambie el requisito.

4.2.4.1. Requisitos de funcionalidad

RS-F001	
Fuente:	RU-R011
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Solo se transformaran las sentencias OCL que representen restricciones de integridad estáticas.

Tabla 27 - RS-F001

RS-F002	
Fuente:	RU-C008 y RU-C009
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se generará un archivo de salida con las transformaciones en SQL-Standard.

Tabla 28 - RS-F002

RS-F003	
Fuente:	RU-C001
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El usuario podrá seleccionar el archivo con el diagrama de clases UML.

Tabla 29 - RS-F003

RS-F004	
Fuente:	RU-C002
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El usuario podrá seleccionar el archivo con las restricciones de integridad estáticas OCL.

Tabla 30 - RS-F004

RS-F005	
Fuente:	RU-C004
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Media
Descripción:	El usuario podrá modificar el archivo con el diagrama de clases UML seleccionado.

Tabla 31 - RS-F005

RS-F006	
Fuente:	RU-C005
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Media
Descripción:	El usuario podrá modificar el archivo con las restricciones de integridad estáticas OCL seleccionado.

Tabla 32 - RS-F006

RS-F007	
Fuente:	RU-R005
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se comprobara que las consultas SQL-Standard generadas por las transformaciones están correctamente formadas.

Tabla 33 - RS-F007

RS-F008	
Fuente:	RU-C003
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se comprobara que las sentencias OCL están correctamente formadas en relación al diagrama de clases.

Tabla 34 - RS-F008

4.2.4.2. Requisitos de prestaciones

RS-P001	
Fuente:	RU-R007
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El porcentaje de errores no controlados tiene que ser menor al 1%, en un mínimo de 100 sentencias.

Tabla 35 - RS-P001

RS-P002	
Fuente:	RU-R008
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El porcentaje de sentencias correctas tiene que ser por lo menos el 75%, en un mínimo de 100 sentencias.

Tabla 36 - RS-P002

4.2.4.3. Requisitos de interfaz

RS-I001	
Fuente:	RU-C001, RU-C002, RU-C004 y RU-C005
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Para seleccionar los archivos, se utilizará la librería javaz.swing.JFileChooser.

Tabla 37 - RS-I001

RS-I002	
Fuente:	RU-C001, RU-C002, RU-C004 y RU-C005
Necesidad:	Deseable
Prioridad:	Baja
Estabilidad:	Alta
Descripción:	Se podrán ver las rutas absolutas de los archivos seleccionados.

Tabla 38 - RS-I002

RS-I003	
Fuente:	RU-C006
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Existirá un botón para cancelar la transformación.

Tabla 39 - RS-I003

RS-I004	
Fuente:	RU-C007
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Existirá un botón para iniciar la transformación.

Tabla 40 - RS-I004

RS-I005	
Fuente:	RU-C010
Necesidad:	Opcional
Prioridad:	Media
Estabilidad:	Media
Descripción:	Se mostraran las estadísticas de las transformaciones al terminar todas las transformaciones.

Tabla 41 - RS-I005

RS-I006	
Fuente:	RU-C008
Necesidad:	Deseable
Prioridad:	Baja
Estabilidad:	Media
Descripción:	Si el archivo de salida ya existe, el usuario deberá confirmar si lo desea sobrescribir.

Tabla 42 - RS-I006

4.2.4.4. Requisitos de operación

RS-O001	
Fuente:	RU-C010 y RU-R012
Necesidad:	Deseable
Prioridad:	Baja
Estabilidad:	Alta
Descripción:	En las estadísticas aparecerá el porcentaje de las transformaciones correctas.

Tabla 43 - RS-O001

RS-O002	
Fuente:	RU-C010 y RU-R013
Necesidad:	Deseable
Prioridad:	Baja
Estabilidad:	Alta
Descripción:	En las estadísticas aparecerá el porcentaje de las transformaciones con fallos controlados.

Tabla 44 - RS-O002

RS-O003	
Fuente:	RU-C010 y RU-R014
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	En las estadísticas aparecerá el porcentaje de las transformaciones con fallos no controlados.

Tabla 45 - RS-O003

RS-O004	
Fuente:	RU-C001, RU-C002, RU-C004 y RU-C005
Necesidad:	Deseable
Prioridad:	Baja
Estabilidad:	Media
Descripción:	En la interfaz deberá aparecer la ruta absoluta de los archivos seleccionados.

Tabla 46 - RS-O004

4.2.4.5. Requisitos de recursos

RS-R001	
Fuente:	RU-R001
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	La herramienta deberá ejecutarse, como mínimo, con JRE Version 7 Update 6.

Tabla 47 - RS-R001

RS-R002	
Fuente:	RU-R002
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El archivo con el diagrama de clases UML tiene que cumplir el estándar UML 2.2 (URI: "http://schema.omg.org/spec/UML/2.2")

Tabla 48 - RS-R002

RS-R003	
Fuente:	RU-R004
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Las restricciones que contiene el archivo con las restricciones de integridad estáticas OCL tienen que cumplir el estándar OCL 2.0.

Tabla 49 - RS-R003

RS-R004	
Fuente:	RU-C003
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se necesita que el archivo con el diagrama de clases UML para validar las sentencias OCL.

Tabla 50 - RS-R004

RS-R005	
Fuente:	RU-C006 y RU-R009
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se necesita que el archivo con el diagrama de clases UML este seleccionado para iniciar la transformación.

Tabla 51 - RS-R005

RS-R006	
Fuente:	RU-C006 y RU-R010
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se necesita que el archivo con las restricciones de integridad estáticas OCL este seleccionado para iniciar la transformación.

Tabla 52 - RS-R006

RS-R007	
Fuente:	RU-R003
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El archivo con el diagrama de clases UML tiene que estar en formato XMI.

Tabla 53 - RS-R007

RS-R008	
Fuente:	RU-R005
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Las transformaciones del archivo de salida tienen que estar en lenguaje SQL-Standard.

Tabla 54 - RS-R008

RS-R009	
Fuente:	RU-C009 y RU-R005
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El archivo con las transformaciones tendrá el formato: *-- TRANSFORMACIÓN n --* *--Restricción de integridad estática OCL* *Transformación SQL-Estándar o Error Encontrado.*

Tabla 55 - RS-R009

RS-R010	
Fuente:	RU-R006
Necesidad:	Deseable
Prioridad:	Media
Estabilidad:	Media
Descripción:	El nombre del archivo con las transformaciones, será el mismo que el que contenía las restricciones de integridad estáticas OCL, pero cambiando la extensión por ".sql".

Tabla 56 - RS-R010

4.2.4.6. Requisitos de verificación

RS-V001	
Fuente:	RU-C003
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se comprobaran que las sentencias de restricción de integridad estáticas OCL, son validas para el diagrama de clases UML del archivo seleccionado.

Tabla 57 - RS-V001

RS-V002	
Fuente:	RU-R016
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se deberán probar al menos 1000 transformaciones de restricciones estáticas OCL a consultas SQL-Standard.

Tabla 58 - RS-V002

RS-V003	
Fuente:	RU-R005
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Se deberán comprobar que las consultas SQL-Standard generadas son validas.

Tabla 59 - RS-V003

4.2.4.7. Requisitos de pruebas de aceptación

RS-A001	
Fuente:	RU-R007
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Solo el uno por ciento de las pruebas podrá dar error no controlado, en un mínimo de 100 sentencias.

Tabla 60 - RS-A001

RS-A002	
Fuente:	RU-R008
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	El 75% de la pruebas, en un mínimos de 100 sentencias, deberán dar transformaciones correctas.

Tabla 61 - RS-A002

4.2.4.8. Requisitos de documentación

RS-D001	
Fuente:	RU-R015
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	Los requisitos deberán de cumplir el estándar ESA PSS-05-0.

Tabla 62 - RS-D001

RS-D002	
Fuente:	RU-R015
Necesidad:	Esencial
Prioridad:	Alta
Estabilidad:	Alta
Descripción:	La documentación deberá contener los puntos mínimos exigidos por la Universidad Carlos III [16].

Tabla 63 - RS-D002

4.2.5. Matriz de trazabilidad

En las siguientes tablas se puede comprobar que todos los requisitos de software provienen de uno o varios requisitos de usuario, y que todos los requisitos de usuario tienen su correspondencia con algún requisito de software.

	RS-F001	RS-F002	RS-F003	RS-F004	RS-F005	RS-F006	RS-F007	RS-F008	RS-P001	RS-P002	RS-I001	RS-I002	RS-I003
RU-C001			X								X	X	
RU-C002				X							X	X	
RU-C003								X					
RU-C004					X						X	X	
RU-C005						X					X	X	
RU-C006													X
RU-C007													
RU-C008		X											
RU-C009		X											
RU-C010													
RU-R001													
RU-R002													
RU-R003													
RU-R004													
RU-R005							X						
RU-R006													
RU-R007									X				
RU-R008										X			
RU-R009													
RU-R010													
RU-R011	X												
RU-R012													
RU-R013													
RU-R014													
RU-R015													
RU-R016													

Tabla 64 - Matriz de trazabilidad 1

	RS-I004	RS-I005	RS-I006	RS-O001	RS-O002	RS-O003	RS-O004	RS-R001	RS-R002	RS-R003	RS-R004	RS-R005	RS-R006
RU-C001							X						
RU-C002							X						
RU-C003											X		
RU-C004							X						
RU-C005							X						
RU-C006												X	X
RU-C007	X												
RU-C008			X										
RU-C009													
RU-C010		X		X	X	X							
RU-R001								X					
RU-R002									X				
RU-R003													
RU-R004										X			
RU-R005													
RU-R006													
RU-R007													
RU-R008													
RU-R009												X	
RU-R010													X
RU-R011													
RU-R012				X									
RU-R013					X								
RU-R014						X							
RU-R015													
RU-R016													

Tabla 65 - Matriz de trazabilidad 2

	RS-R007	RS-R008	RS-R009	RS-R010	RS-V001	RS-V002	RS-V003	RS-A001	RS-A002	RS-D001	RS-D002
RU-C001											
RU-C002											
RU-C003					X						
RU-C004											
RU-C005											
RU-C006											
RU-C007											
RU-C008											
RU-C009			X								
RU-C010											
RU-R001											
RU-R002											
RU-R003	X										
RU-R004											
RU-R005		X	X				X				
RU-R006				X							
RU-R007								X			
RU-R008									X		
RU-R009											
RU-R010											
RU-R011											
RU-R012											
RU-R013											
RU-R014											
RU-R015										X	X
RU-R016						X					

Tabla 66 - Matriz de trazabilidad 3

4.3. Casos de uso

A continuación se detallan los casos de usos de la herramienta desarrollada. Solo se cuenta con un actor (“*Usuario*”) que es la persona que está utilizando la herramienta desarrollada.

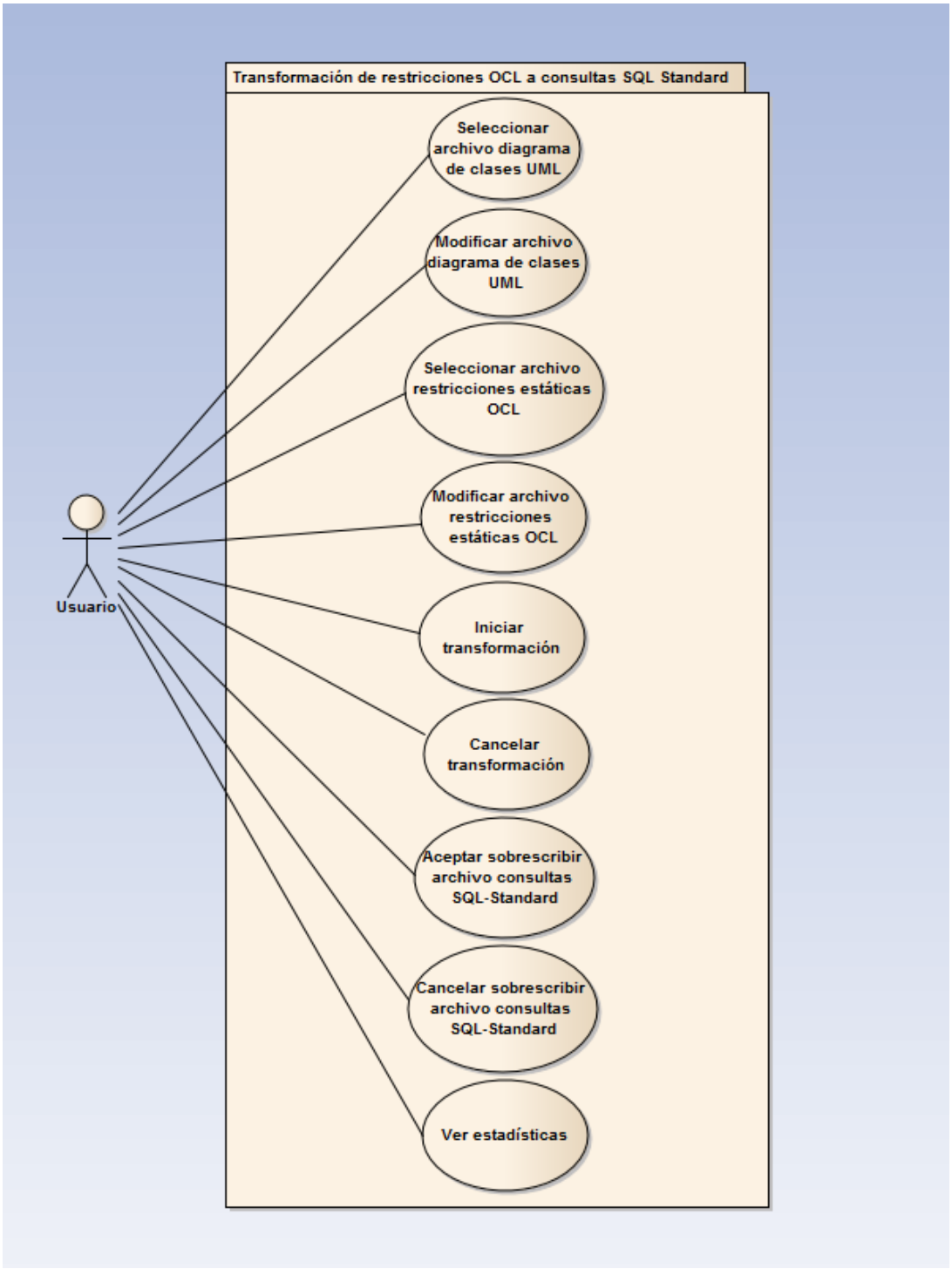


Ilustración 16 - Casos de uso

A continuación se detalla un esquema resumen con los casos de uso del sistema donde se describe la funcionalidad de cada uno de los casos de uso:

Caso de Uso: seleccionar archivo diagrama de clases UML.

Actores: Usuario.

Objetivo: seleccionar el archivo donde se encuentra el diagrama de clases UML.

Precondiciones: la herramienta debe estar iniciada.

Postcondiciones: poder iniciar la transformación.

Caso de Uso: modificar archivo diagrama de clases UML.

Actores: Usuario.

Objetivo: modificar el archivo donde se encuentra el diagrama de clases UML que ya había sido seleccionado.

Precondiciones: el archivo donde se encuentra el diagrama de clases UML debe estar seleccionado.

Postcondiciones: poder iniciar la transformación.

Caso de Uso: seleccionar archivo restricciones de integridad estáticas OCL.

Actores: Usuario.

Objetivo: seleccionar el archivo donde se encuentran las restricciones de integridad estáticas OCL.

Precondiciones: la herramienta debe estar iniciada.

Postcondiciones: poder iniciar la transformación.

Caso de Uso: modificar archivo restricciones de integridad estáticas OCL.

Actores: Usuario.

Objetivo: modificar el archivo donde se encuentran las restricciones de integridad estáticas OCL.

Precondiciones: el archivo donde se encuentran las restricciones de integridad estáticas OCL debe estar seleccionado.

Postcondiciones: poder iniciar la transformación.

Caso de Uso: iniciar transformación.

Actores: Usuario.

Objetivo: comenzar a realizar las transformaciones de las restricciones OCL a las consultas SQL-Standard.

Precondiciones: que los archivos que contienen el diagrama de clases UML y las restricciones de integridad estáticas OCL estén seleccionados.

Postcondiciones: generar un archivo con las transformaciones, consultas SQL-Standard.

Caso de Uso: cancelar transformación.

Actores: Usuario.

Objetivo: cerrar la herramienta desarrollada.

Precondiciones: la herramienta debe estar iniciada.

Postcondiciones: la herramienta queda cerrada.

Caso de Uso: aceptar sobrescribir archivo consultas SQL-Standard.

Actores: Usuario.

Objetivo: cuando ya existe el archivo donde se van a guardar las transformaciones, se solicita permiso al usuario para sobrescribir dicho fichero.

Precondiciones: iniciar la transformación.

Postcondiciones: transformar las restricciones OCL a las consultas SQL-Standard.

Caso de Uso: cancelar sobrescribir archivo consultas SQL-Standard.

Actores: Usuario.

Objetivo: cuando ya existe el archivo donde se van a guardar las transformaciones, se solicita permiso al usuario para sobrescribir dicho fichero.

Precondiciones: iniciar la transformación.

Postcondiciones: guardar el archivo que iba a ser sobrescrito e iniciar de nuevo las transformaciones.

Caso de Uso: ver estadísticas.

Actores: Usuario.

Objetivo: ver el porcentaje de transformaciones que han acabado satisfactoriamente y cuales han acabado con errores.

Precondiciones: iniciar las transformaciones.

Postcondiciones: consultar las consultas SQL-Standard generadas a partir de las restricciones estáticas OCL.

5. Propuesta de transformar OCL a SQL

En este capítulo se realizará una introducción sobre las sentencias OCL y sobre el SQL Standard. Posteriormente se definirán las reglas de transformación entre OCL y SQL, y por último se expondrán ejemplos donde se aplican estas reglas de transformación

5.1. Ámbito de la tecnología

5.1.1. Sentencias OCL

A continuación se procede a realizar un análisis de los tipos de datos que tiene el lenguaje OCL y la estructura de las sentencias.

- **Tipos de datos**

Tipos básicos: existen una seria de tipos básico predefinidos, que son: *Real*, *Integer*, *Boolean*, *String* y *Enumeration*.

Tipos clase: es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno del estudio.

Tipos colección: son los tipos predefinidos *Set*, *Bag*, *OrderedSet* y *Sequence*. A continuación se muestra una tabla resumen con las características estos tipos. Dichas características se basan en si la colección tiene los elementos ordenados o si estos son únicos o no.

Tipos básicos	¿Ordenado?	¿Único?
Set	No	Si
Bag	No	No
OrderedSet	Si	Si
Sequence	Si	No

Tabla 67 - Tipos colección en OCL

- **Estructura sentencias OCL**

La estructura general que tiene una sentencia OCL es:

```
context <nombre_clase> inv <nombre_restricción>:
<expresión_OCL(self)>
```

self es una instancia de un tipo (por ejemplo, Compañía). *context* significa la clase en la que está definida la expresión OCL. *nombre_restricción* indica el nombre que se le quiere dar a la restricción. *expresión_OCL* es una expresión lógica que describe una relación entre dos expresiones. La expresión se evalúa como verdadero si la relación se cumple y como falso en caso contrario.

5.1.2. SQL Standard

Tal y como se ha dicho en la punto *1. Introducción:* el lenguaje SQL Standard "*es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar operaciones sobre éstas*". El SQL fue un lenguaje que pasó en los años 80 a ser ampliamente utilizado en diversos sistemas de gestión de bases de datos, lo que llevó al Instituto Nacional Estadounidense de Estándares (*ANSI, American National Standars Institute*) [17] a estandarizarlo bajo el estándar ISO/IEC 9075. Actualmente se encuentra en la versión ISO/IEC 9075:2011 [4].

En la actualidad, existen diversos sistemas de gestión de bases de datos (MySQL [18], Oracle [19], PostgreSQL [20], SQLite [21], Microsoft SQL Server [22]), tanto libres como de pago, cuyas diferencias sintácticas son mínimas entre ellos al basarse todos en el SQL Standard.

A continuación, se procede a analizar las partes que van a ser utilizadas en este Trabajo de Fin de Grado.

- **Tablas**

El lenguaje de definición de datos (*DDL, Data Definition Language*) es el encargado de permitir la descripción de los objetos que forman una base de datos. Uno de estos objetos son las tablas, que son elementos que contienen los datos almacenados de la base de datos.

En este caso, cada clase del diagrama de clases se convertiría en una tabla.

- **Sentencias**

El lenguaje de manipulación de datos (*DML, Data Management Language*), permite crear sentencias que sirvan para recuperar, insertar, borrar o modificar los datos almacenados en la base de datos. Una sentencia es una o varias expresiones que especifican una o varias operaciones sobre tablas.

- **Consultas**

Una consulta es un tipo de sentencia que permite acceder a los datos que se encuentran en las tablas de la bases de datos. Una consulta se puede dividir muchas partes pero solo se van a analizar las partes que son necesarias para esta herramienta. Por lo tanto, con esta restricción, la consulta tendría la siguiente la estructura, siendo obligatorias solo dos, SELECT y FROM, y teniendo que ir cada una de las partes en el orden que se muestra:

```
SELECT <nombre_campo> [{,<nombre_campo>}]
FROM <nombre_tabla>|<nombre_vista>
        [{,<nombre_tabla>|<nombre_vista>}]
[JOIN <nombre_tabla>|<nombre_vista>
 ON <condición de combinación>]
[WHERE <condición> [{AND|OR <condición>}]]
[GROUP BY <nombre_campo> [{,<nombre_campo >}]]
```

SELECT - Palabra clave que indica que la sentencia SQL que queremos ejecutar es de selección.

FROM - Indica la tabla (o tablas) desde la que queremos recuperar los datos.

JOIN – Es una consulta combinada que nos permite unir varias tablas. La unión de los campos de dichas tablas se realiza en el **ON.**

WHERE - Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos *AND* y *OR*.

GROUP BY - Especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas.

5.2. Reglas de transformación

Basándose en la estructura de una sentencia OCL.

```
context <nombre_clase> inv <nombre_restricción>:
<expresión_OCL(self)>
```

La transformación directa de esta sentencia OCL a SQL Standard, sería:

```
SELECT * FROM tabla_contexto SELF
WHERE NOT expresión_OCL
```

Donde la *tabla_contexto* de la consulta sería sustituida por el *nombre_clase* de la sentencia OCL y la *expresión_OCL* es la que se obtiene de aplicar las reglas de transformación a la *expresión_OCL(self)*. Hay que destacar que la *expresión_OCL* va negada ya que la sentencia OCL es una restricción de los datos que no queremos que contenga la base de datos.

Ejemplo:

Restricción OCL: "Las Publicaciones tienen que ser del autor Bertrand Rusbelt".

Consulta SQL: "Publicaciones que no sean del autor Bertrand Rusbelt".

De esta manera si a la hora de comprobar la consulta, esta devuelve datos, eso significa que la restricción está siendo violada.

A continuación, analizamos los distintos tipos de expresiones OCL que existen y las operaciones que se pueden realizar sobre ellas.

Todos los ejemplos realizados de este apartado, se van a basar en el siguiente diagrama de clases:

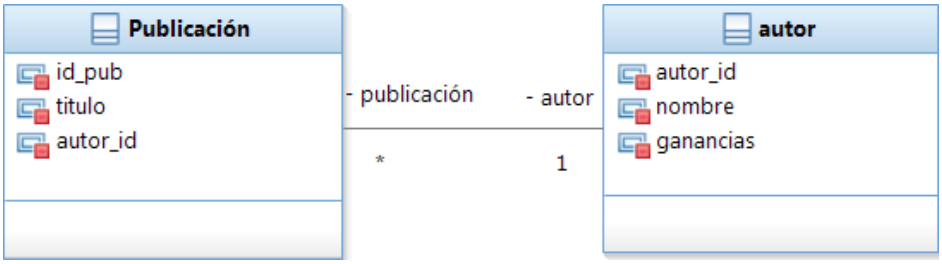


Ilustración 17 - Diagrama de clases ejemplo

5.2.1. Restricciones sobre un objeto de una clase

Para este tipo de expresiones, la estructura de las sentencias es el siguiente:

```
context <nombre_clase> inv <nombre_restricción>:
<expresión_OCL> [OR||AND||XOR||IMPLIES <expresión_OCL>]^n
```

Siendo n un número natural entero.

La *expresión_OCL*, puede ser de la siguiente manera:

elemento1 operación elemento2

El *elemento1* y el *elemento2* pueden tener la siguiente estructura:

- *Término*: elemento que puede tomar cualquier valor de los tipos básicos de datos, ya definidos en el punto anterior, excepto *Enumeration*.
- *self.atributo*: con *self* queremos decir que se instancia la clase cuyo nombre es *nombre_clase* y después, mediante el *atributo* seleccionamos un atributo que pertenezca a esa clase.

La *operación* puede tener los siguientes valores: <, >, =, <>, <=, >=.

Después de haber analizado la sentencia OCL que restringe a un objeto de una clase procedemos a analizar la transformación de esta sentencia. El patrón de la consulta SQL Standard, sería la siguiente:

```
SELECT * FROM tabla_contexto self
WHERE NOT <expresión_OCL>
(OR||AND [NOT] < expresión_OCL >)^n
```

Siendo n un número natural entero.

Si se deciden hacer expresiones compuestas, es decir, utilizando los operadores *OR, AND, XOR* e *IMPLIES*, la transformación cambia ya que el lenguaje SQL Stantard solo permite los operadores AND y OR. Para realizar dicha transformación se aplican las reglas definidas en la siguiente tabla:

Expresión OCL	Expresión SQL
exprOCL1 **OR** exprOCL2	NOT exprSQL1 AND NOT exprSQL2
exprOCL1 **AND** exprOCL2	NOT exprOCL1 OR NOT exprOCL2
exprOCL1 **XOR** exprOCL2	(exprSQL1 AND exprSQL2) OR (NOT exprSQL1 AND NOT exprSQL2)
exprOCL1 **IMPLIES** exprOCL2	exprSQL1 AND NOT exprSQL2

Tabla 68 - Transformación de los operandos OR, AND, XOR e IMPLIES

Ejemplo:

"Personas que estén en activo, es decir, que estén entre los 18 y 67 años"

```
context Persona inv menores:
self.edad > 18 AND self.edad < 67
```

Su transformación, aplicando las reglas vistas sería:

```
SELECT * FROM Persona self
WHERE NOT self.edad > 18 OR NOT self.edad < 67
```

5.2.2. Restricciones sobre dos objetos de una misma

Para realizar este tipo de restricciones hay que utilizar la operación *allInstances,* cuyo resultado es el conjunto de todas las instancias del tipo existente en un momento determinado a la hora de evaluar una expresión.

Basándose en el estudio realizado por la Universidad Politécnica de Valencia: *"El uso de allInstances tiene algunos problemas y su uso está desaconsejado en muchos casos. Para los tipos integer, real y string el significado de allInstances está indefinido. Además, la existencia de objetos debe ser considerada en algunos contextos globales, como un sistema o un modelo. Este contexto global debe ser definido, lo que no se hace con OCL."* [23], esta operación no va a ser trasformada.

5.2.3. Restricciones con navegación

En este caso la estructura de la sentencia OCL es la misma que en las restricciones sobre un objeto en una clase. Lo único que cambia en el análisis de la *expresión_OCL* es el valor que pueden tomar el *elemento1* y el *elemento2*, que además de los valores de ese caso, puede tener el siguiente:

- *self.navegación.atributo*: con *self* queremos decir que instanciamos la clase cuyo nombre es *nombre_clase*; *navegación* indica el rol de la clase hacia la que queremos ir; *atributo* es un atributo de la clase hacia la que hemos navegado.

Si analizamos como sería su transformación a SQL Standard hay que fijarse en la parte *JOIN* de la consulta, porque es la que nos va a permitir realizar esta navegación.

Como se produce una unión entre dos clases tiene que existir un atributo en la clase desde la que navegamos, que sea clave ajena de la case hacia la que

navegamos, y que la clase hacia la que navegamos tenga una clave primaria. Como estas claves puede que no existan, se van a crear para que siempre que haya una asociación se pueda realizar una navegación. El nombre de dichos atributos se definirá en la estructura de esta transformación a una consulta SQL Standard, que es:

```
SELECT * FROM tabla_contexto self
JOIN tabla_navegación a
ON self.tabla_navegacion_id = a.id
WHERE NOT <expressión_OCL>
(OR||AND [NOT] < expressión_OCL >)^n
```

Siendo n un número natural entero.

Ejemplo:

"Las publicaciones escritas solo pueden estar escritas por Robert Steinball"

```
context Publicación inv resPub:
self.autor.nombre = "Robert Steinball"
```

Basándonos en su regla su transformación sería:

```
SELECT * FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE NOT aut.nombre = "Robert Steinball"
```

5.2.4. Restricciones con navegación más una función de agregación

Este tipo de restricciones se basan en la anterior, dado que, para que se produzca tiene que existir una navegación. Lo único que cambiaría en el análisis de la *expresión_OCL* es el valor que pueden tomar el *elemento1* y el *elemento2*. La estructura que tienen que seguir es:

- *self.navegación.atributo->función()*: con *self* queremos decir que instanciamos la clase cuyo nombre es *nombre_clase*; *navegación* indica el rol de la clase hacia la que queremos ir; *atributo* es un atributo de la clase hacia la que hemos navegado; *función* especifica la operación que se quiere realizar sobre la navegación.
- *self.navegacion->función()*: es idéntico al caso anterior, lo único que la función se realiza sobre una clase y no sobre un atributo.

Antes de realizar un análisis sobre las distintas funciones hay que aclarar que si una expresión OCL se pasa como parámetro a una función, ésta puede tener una de las tres siguientes estructuras:

Estructura 1:

```
elemento1 operación elemento2
```

Es igual que la expresión OCL definida en el punto "Sobre un atributo". Su transformación es la misma que una expresión normal.

Estructura 2:

```
e | e.elemento1 operación elemento2
```

En este caso '*e*' indica el nombre corto que le debemos dar a la última clase hacia la que hemos navegado en la consultas SQL.

Estructura 3:

```
e : clase | e.elemento1 operación elemento2
```

En este caso '*e*' indica el nombre corto que le tenemos que asociar a una clase en la consulta SQL y *clase* es el nombre de la clase a la que le vamos a asignar el nombre corto ('*e*').

Una vez aclarado este punto, se va realizar un análisis de cada una de las funciones que existen explicando su significado y si es una sentencia que se deba transformar, la estructura de la consulta SQL Standard a la que es transformada. En este análisis se van a dividir las funciones dependiendo de lo que éstas devuelvan:

5.2.4.1. Funciones que devuelven un Integer

count (object : T) : Integer

Devuelve el número de veces que aparece el objeto *object* en la colección. Esta sentencia se puede realizar de dos maneras. Si se tiene un conjunto de elementos *object* tendría que ser uno de los elementos posibles. Si solo se tiene un elemento *object* será un tipo básico u otro elemento con el que le queremos comparar.

Su transformación es:

Si se tienen un conjunto de elementos:

```
SELECT count(object) FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
```

Ejemplo:

"Número de autores que tienen alguna publicación"

```
context Publicación inv:
self.autor->count(autor_id)
```

Su transformación sería:

```
SELECT count(aut.autor_id)
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
```

Si se tiene un elemento:

```
SELECT count(*) FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE a.atributo = object
```

Ejemplo:

"Numero de autores que tienen alguna publicación y su identificación es 'GAGA'"

```
context Publicación inv:
self.autor.autor_id->count('GAGA')
```

Su transformación sería:

```
SELECT count(*)
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE aut.autor_id = 'GAGA'
```

size () : Integer

Devuelve el número de elementos que existen en la colección. Esta sentencia solo se puede realizar si se tiene un conjunto de elementos. La estructura de su transformación es:

```
SELECT count(*) FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
```

Ejemplo:

"Número de autores que tienen alguna publicación"

```
context Publicación inv:
self.autor->size()
```

Su transformación sería:

```
SELECT count(*)
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
```

sum () : Integer

Devuelve la suma de los elementos de la colección. Esta sentencia se realiza sobre un elemento. Su transformación es:

```
SELECT sum(atributo) FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
```

Ejemplo:

"Suma de las ganancias de los autores que tienen alguna publicación"

```
context Publicación inv:
self.autor.ganancias->sum()
```

Su transformación sería:

```
SELECT sum(aut.ganancias)
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
```

5.2.4.2. Funciones que devuelven un Boolean

excludes (object : T) : Boolean

Es una función que devuelve *true* si el *object* es el único que está contenido en la colección y *false* en caso contrario. Al igual que la función *sum,* el último término tiene que ser un atributo. La estructura de su transformación es:

```
SELECT case count(*) when 0 then true else false end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE NOT a.atributo = object
```

Ejemplo:

"¿Es el autor con identificación 'GAGA' el único que tiene alguna publicación?"

```
context Publicación inv:
self.autor.autor_id->excludes('GAGA')
```

Su *transformación* sería:

```
SELECT case count(*) when 0 then true else false end
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE NOT aut.autor_id = 'GAGA'
```

excludesAll (c2 : Collection(T)) : Boolean

Devuelve *true* si ningún elemento de la colección *c2* está contenido en la colección sobre la que operamos y *false* en caso contrario. Al igual que la función *exclude*, el último término de ambas colecciones tiene que ser un atributo. Su estructura tras realizar la transformación es:

```
SELECT case count(*) when 0 then true else false end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE a.atributo = object
```

Ejemplo:

"En una publicación, el nombre del autor tiene que ser el mismos que el de el título de la publicación."

```
context Publicación inv:
self.autor.nombre->excludesAll(self.título)
```

Su transformación sería:

```
SELECT case count(*) when 0 then true else false end
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE aut.nombre = self.títitulo
```

exists (expr : expresiónOCL) : Boolean

Devuelve *true* si existe, al menos, un elemento de la colección que cumpla la expresión *expr*. Devuelve *false* en caso contrario. *expr* puede tener una de las tres estructuras ya definidas. La estructura genérica de su transformación es:

```
SELECT case count(*) when 0 then false else true end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE expresiónOCL
```

Existen dos maneras distintas de realizar la misma sentencia. Esto se debe a que si el último término de la colección es un atributo, en la expresión OCL, se refiere al mismo con el término '*self*' mientas que si el último término es una navegación se tendrá que definir a qué atributo de la clase se quiere acceder.

Ejemplo:

"¿Existe un autor que tenga alguna publicación y cuyo nombre sea Robert Steinball?"

```
context Publicación inv:
self.autor ->exists( e | e.nombre = 'Robert Steinball')
```

Su transformación sería:

```
SELECT case count(*) when 0 then false else true end
FROM Publicación self
JOIN autor e
ON self.autor_id = e.id
WHERE e.nombre = 'Robert Steinball'
```

forAll (expr : expresiónOCL) : Boolean

Devuelve *true* si todos los elementos de la colección cumplen la expresión *expr*. Devuelve *false* en caso contrario. Aunque, al igual que en el caso anterior (*exists)* existen distintas maneras de realizar la misma expresión OCL, la estructura de la transformación es la misma.

```
SELECT case count(*) when 0 then true else false end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE NOT expresiónOCL
```

Ejemplo:

"¿Todas las publicaciones han sido escritas por el autor cuyo nombre sea Robert Steinball?"

```
context Publicación inv:
self.autor ->forAll(e : autor | e.autor = 'Robert Steinball')
```

Su transformación sería:

```
SELECT case count(*) when 0 then true else false end
FROM Publicación self
JOIN autor e
ON self.autor_id = e.id
WHERE NOT e.nombre = 'Robert Steinball'
```

includes (object : T) : Boolean

Es una función que devuelve *true* si el *object* está contenido en la colección y *false* en caso contrario. Al igual que la función *excludes*, el último término tiene que ser un atributo. La estructura de la transformación es:

```
SELECT case count(*) when 0 then false else true end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE a.atributo = object
```

Ejemplo:

"¿Existe alguna publicación del autor con identificación 'GAGA'?"

```
context Publicación inv:
self.autor.autor_id->includes('GAGA')
```

Su transformación sería:

```
SELECT case count(*) when 0 then false else true end
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE aut.autor_id = 'GAGA'
```

includesAll (c2 : Collection(T)) : Boolean

Devuelve *true* si todos los elementos de la colección *c2* están contenidos en la colección sobre la que operamos y *false* en caso contrario. Es lo contrario a

excludesAll. Al igual que ésta, el último término de ambas colecciones tiene que ser un atributo. La estructura de su transformación es:

```
SELECT case count(*) when 0 then true else false end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE NOT a.atributo = object
```

Ejemplo:

"En una publicación, el nombre del autor nunca pude ser el mismo que el de el título de la publicación."

```
context Publicación inv:
self.autor.nombre->includesAll(self.título)
```

Su transformación sería:

```
SELECT case count(*) when 0 then true else false end
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE NOT aut.nombre = self.tititulo
```

En este caso, título es una colección porque contiene varios elementos.

isEmpty () : Boolean

Devuelve *true* si la colección está vacía y *false* en caso contrario. En este caso el último elemento tiene que ser una navegación. La estructura de su transformación es:

```
SELECT case count(*) when 0 then true else false end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
```

Ejemplo:

"¿Existe algún autor con alguna publicación?"

```
context Publicación inv:
self.autor ->isEmpty()
```

Su transformación sería:

```
SELECT case count(*) when 0 then true else false end
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
```

isUnique (expr : expresiónOCL) : Boolean

Devuelve *true* si solo existe un elemento, o es el mismo elemento repetido varias veces, que cumple la expresión *expr*. Devuelve *false* en caso contrario. Al igual que en el caso *exists* existen distintas maneras de realizar la misma expresión OCL. La estructura de su transformación es:

```
SELECT case count(*) when 1 then true else false end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE expresiónOCL
GROUP BY a.atributo
```

Ejemplo:

"El autor cuyo nombre es Robert Steinball, ¿es el único que tiene publicaciones?"

```
context Publicación inv:
self.autor ->isUnique(nombre = 'Robert Steinball')
```

Su transformación sería:

```
SELECT case count(*) when 1 then true else false end
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE aut.nombre = 'Robert Steinball'
GROUP BY aut.nombre
```

notEmpty () : Boolean

Es la función contraria a *isEmpty y d*evuelve *true* si la colección está no vacía y *false* en caso contrario. Al igual que en la función *isEmpty,* el último elemento tiene que ser una navegación. La estructura de su transformación es:

```
SELECT case count(*) when 0 then false else true end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
```

Ejemplo:

"¿Existe alguna publicación que tenga algún autor?"

```
context Publicación inv:
self.autor ->notEmpty()
```

Su transformación sería:

```
SELECT case count(*) when 0 then false else true end
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
```

one (expr : expresiónOCL) : Boolean

Devuelve *true* si solo existe un elemento que cumple la expresión *expr*. Devuelve *false* en caso contrario. Es una función parecida a *isUnique*, pero en esta función si se repite el mismo elemento se devuelve *false*. Al igual que en el caso *exists* existen distintas maneras de realizar la misma expresión OCL. La estructura de su transformación es:

```
SELECT case count(*) when 1 then true else false end
FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE expresiónOCL
```

Ejemplo:

"El autor cuyo nombre es Robert Steinball, ¿tiene solo una publicación?"

```
context Publicación inv:
self.autor.nombre ->one(self = 'Robert Steinball')
```

Su transformación sería:

```
SELECT case count(*) when 1 then true else false end
FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE aut.nombre = 'Robert Steinball'
```

5.2.4.3. Funciones que devuelven una Colección

asSet () : Set(T)

Devuelve los elementos únicos que tiene la colección sobre la que operamos. Al igual que la función *sum*, el último término tiene que ser un atributo. La estructura de su transformación es:

```
SELECT * FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
GROUP BY a.atributo
```

Ejemplo:

"Distintos nombre de autores que hayan realizado alguna publicación"

```
context Publicación inv:
self.autor.nombre ->asSet()
```

Su transformación sería:

```
SELECT * FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
GROUP BY aut.nombre
```

collect (expr : expresiónOCL) : Collection(T2)

Devuelve una nueva colección con los elementos que cumplen la expresiónOCL. Al igual que en el caso *exists* existen distintas maneras de realizar la misma expresión OCL. La estructura de su transformación es:

```
SELECT * FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE expresiónOCL
```

Ejemplo:

"Autor que tengan alguna publicación y cuyo nombre sea Robert Steinball"

```
context Publicación inv:
self.autor ->collect(nombre = 'Robert Steinball')
```

Su transformación sería:

```
SELECT * FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE aut.nombre = 'Robert Steinball'
```

excluding (object : T) : Collection(T)

Devuelve la misma colección pero con los elementos que no son iguales que el *object*. Al igual que la función *sum*, el último término tiene que ser un atributo. La estructura de su transformación es:

```
SELECT * FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE NOT a.atributo = object
```

Ejemplo:

"Autor que tengan alguna publicación y cuyo nombre no sea Robert Steinball"

```
context Publicación inv:
self.autor.nombre->excluding('Robert Steinball')
```

Su transformación sería:

```
SELECT * FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE NOT aut.nombre = 'Robert Steinball'
```

reject (expr : expresiónOCL) : Collection(T)

Devuelve la misma colección pero con los elementos que no cumplen la expresiónOCL. Al igual que en el caso *exists* existen distintas maneras de realizar la misma expresión OCL. La estructura de su transformación es:

```
SELECT * FROM tabla_contexto self
JOIN tabla_navegación a ON self.tabla_navegacion_id = a.id
WHERE NOT expresiónOCL
```

Ejemplo:

"Autor que tengan alguna publicación y cuyo nombre no sea Robert Steinball"

```
context Publicación inv:
self.autor ->reject(nombre = 'Robert Steinball')
```

Su transformación sería:

```
SELECT * FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE NOT aut.nombre = 'Robert Steinball'
```

select (expr : expresiónOCL) : Collection(T)

Devuelve la misma colección pero con los elementos que cumplen la expresiónOCL. Es la expresión contraria a *reject* y, en el ámbito de la transformación, es igual que *collect.* Por lo tanto tienes sus mismas características y su misma estructura de transformación.

Ejemplo:

"Autor que tengan alguna publicación y cuyo nombre sea Robert Steinball"

```
context Publicación inv:
self.autor ->select(nombre = 'Robert Steinball')
```

Su transformación sería:

```
SELECT * FROM Publicación self
JOIN autor aut
ON self.autor_id = aut.id
WHERE aut.nombre = 'Robert Steinball'
```

- **Funciones sin transformación**

A continuación, se van a analizar las expresiones que no se van a transformar debido a que para las sentencias de restricciones de integridad estáticas, no tienen sentido.

any (expr : expresiónOCL) : T

Devuelve un objeto de todos aquellos que cumplen la expresión *expr.* No tiene sentido obtener un objeto aleatorio de la colección para comprobar si se cumple la restricción.

collectNested (expr : expresiónOCL) : Collection (T2)

Devuelve una nueva colección con los elementos que cumplen *expr*. Los resultados no se unirán, es decir, que será una colección de objetos donde cada objeto es una colección. Esta fundición no se transforma porque no tiene sentido para sentencias de restricción aunque sí que lo tiene consultas.

flatten () : Collection (T2)

Devuelve una colección que es el resultado de la unión de dos colecciones. Es una función que solo se utiliza para crear consultas OCL.

including (object : T) : Collection (T2)

Es similar a la función anterior, *flatten,* lo único que en este caso en vez de unir dos colecciones, agrega a una colección el elemento *object.* Al igual que *flatten,* es una función que solo se utiliza para crear consultas OCL.

product (Collection (T2)) : Set(Tuple(first : T, second : T2))

Esta función devuelve las tuplas que se producen al realizar el producto cartesiano de la colección sobre la que se realizar la función junto con la colección que se le pasa por parámetro *T2*. Siendo una tupla, una lista con un número determinado de elementos. Esta función no es tenida en cuenta porque no se puede operar con las tuplas, solo sirven para mostrar el resultado, es decir, para consultas OCL.

sortedBy (expr : expresiónOCL) : Sequence(T)

Esta función devuelve una colección de tipo *Sequence* con todos los elementos de los que cumplan la expresión OCL, *expr.* Cada elemento es una instancia de una clase que contiene los un elemento que cumple *expr.* Esta función al igual que las anteriores, no tiene sentido en las restricciones pero si en las consultas.

Las siguiente tres funciones corresponden a transformación entre los distintos tipos de colección anteriormente explicados en el punto *Tipos de datos*. Estas transformaciones no tienen sentido porque devolverían la misma colección pero ordenada o con elementos únicos. La única función similar a éstas que tiene sentido trasformar es *asSet* y ya ha sido explicada.

asBag () : Bag(T)

Convierte una colección en una colección de tipo *Bag*.

asOrderedSet () : OrderedSet (T)

Convierte una colección en una colección de tipo *OrderedSet*.

asSequence () : Sequence(T)

Convierte una colección en una colección de tipo *Sequence*.

6. Diseño e implementación

En este capítulo se define en detalle el diseño de la arquitectura del sistema, se especifican los componentes y índica la manera seguida para implementar el sistema.

6.1. Diseño del sistema

6.1.1. Arquitectura del sistema

La arquitectura que se ha utilizado para el desarrollo de la herramienta ha sido la arquitectura MDA (*Model-Driven Architecture, Arquitectura Dirigida por Modelos*). MDA es una metodología para el desarrollo del software, que igual que los estándares de UML y OCL, ha sido definida por OMG (*Object Management Group*) [24]. La ventaja de la arquitectura MDA es que les confiere importancia a los modelos en el proceso de desarrollo del software. En la especificación MDA [25] se define un modelo del sistema como una descripción o especificación tanto del sistema como de su entorno en un lenguaje (gráfico y/o textual) bien definido para un propósito determinado.

Para MDA, el proceso del desarrollo del software consiste en la transformación sucesiva de modelos para llegar al producto final. En este proceso se distingue entre modelos PIM y modelos PSM.

- Un PIM (*Platform Independent Model*) es un modelo independiente de la plataforma, es decir, es un modelo que describe un sistema sin hacer referencia a una plataforma concreta.
- Un PSM (*Platform Specific Model*) es un modelo específico de la plataforma, es decir, es un modelo que describe un producto para una plataforma en concreto.

En base a estos modelos, la arquitectura del sistema quedaría de la siguiente manera:

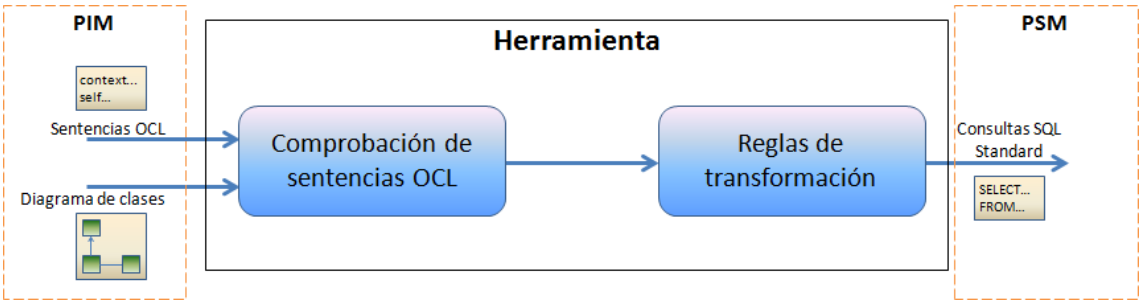


Ilustración 18 - Arquitectura del sistema

Los datos de entrada para el sistema son:

- Diagrama de clases UML.
- Sentencias de restricciones de integridad estáticas OCL.

La salida que obtenemos el sistema es:

- Consultas SQL Standard.

Los componentes del sistema son:

- Comprobación de sentencias OCL. Se encarga de comprobar que las sentencias OCL están bien formadas y tienes sentido con el diagrama de clases UML.
- Reglas de transformación. Se encarga de aplicas las reglas definidas en el apartado anterior para transformar las sentencias OCL, que han sido comprobadas, en consultas SQL Standard.

6.1.2. Diagrama de clases

Antes de mostrar el diagrama de clases, se va a especificar el estándar de nombrado de paquetes y clases del sistema.

- Los paquetes son nombrados con letras minúsculas, y tienen que empezar con es.scasillas.ocltosql.
- Las clases son nombradas con la primera letra en mayúscula.

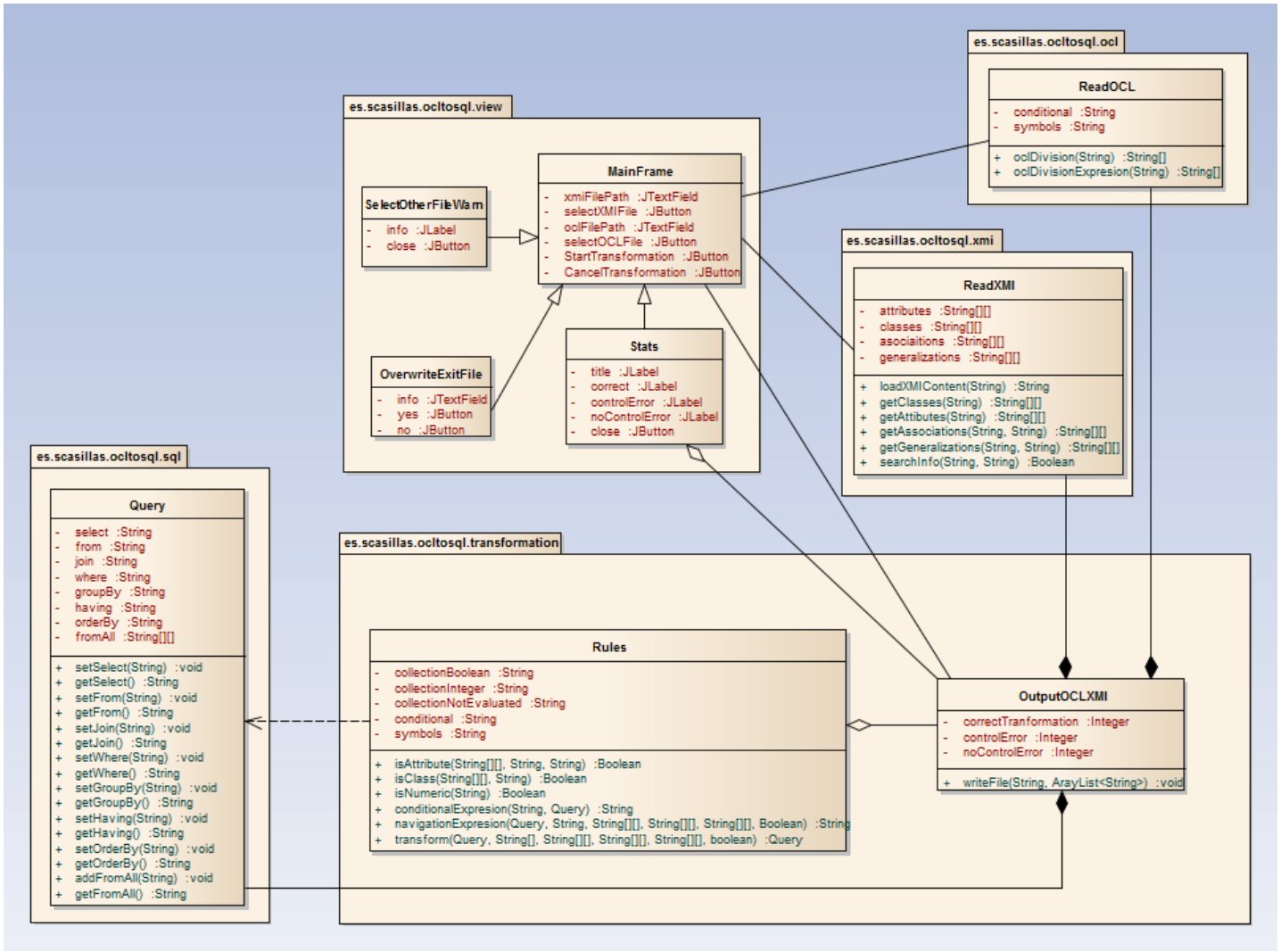


Ilustración 19 - Diagrama de clases

Una vez mostrado el esquema con el diagrama de clases se procede a detallar el contenido de los paquetes y las funcionalidades de cada clase.

Paquete es.scasillas.ocltosql.view

Este paquete contiene todas las clases relacionadas con la interfaz gráfica:

- *MainFrame*

  Es la clase principal dentro de la interfaz gráfica. Es la que hace posible la interactuación del usuario con el sistema y es la que, dependiendo de las acciones realizadas, ejecuta las otras clases de este paquete. Esta clase permite al usuario seleccionar el archivo que contiene el diagrama de clases UML y el archivo con las sentencias OCL. A su vez también permite al usuario comenzar la transformación o cerrar la herramienta.
- *SelectOtherFileWarn*

  Esta clase muestra una advertencia al usuario si una vez que ha pulsado el botón de iniciar la transformación se comprueba que no ha seleccionado el archivo con el diagrama de clases UML o el archivo con las sentencias OCL, dado que sin ambos archivos no puede comenzar la transformación.
- *OverwriteExitFile*

  Esta clase avisa al usuario en el caso de que el archivo de salida exista y este vaya a ser sobrescrito. Hay que recordar que el archivo de salida estará en la misma ruta y tendrá el mismo nombre que el archivo de sentencias OCL pero cambiando la extensión por ".sql". Con esta clase se permite al usuario decidir si quiere o no sobrescribir el archivo. En el caso de que seleccione no, no se realizará la transformación.
- *Stats*

  Esta clase muestra, una vez que se han realizado todas las transformaciones, las estadísticas obtenidas de las transformaciones de las sentencias OCL. Una sentencia OCL puede obtener tres estados después de haber realizado la transformación: transformación correcta,

transformación con errores controlados y transformación con errores no controlados.

Paquete es.scasillas.ocltosql.xmi

Este paquete contiene la clase que permite las operaciones sobre el archivo con el diagrama de clases UML:

- ReadXMI

  Esta clase es la que a través de la ruta obtenida por la interfaz grafica, obtiene el archivo con el diagrama de clases UML en formato XMI y extrae los datos necesarios del diagrama de clases, es decir, las clases, los atributos, las asociaciones y las generalizaciones.

Paquete es.scasillas.ocltosql.ocl

Este paquete contiene la clase que extrae las sentencias OCL del archivo que contiene:

- ReadOCL

  Esta es la clase que obtiene y divide las sentencias que están contenidas el archivo con las sentencias OCL. Para obtener la ruta del archivo, se basa en la interfaz gráfica, donde el usuario ya ha seleccionado el archivo que contiene dichas sentencias.

Paquete es.scasillas.ocltosql.sql

Este paquete contiene la clase que define la estructura de la consulta SQL:

- Query

  Esta clase define la estructura de la sentencia de salida SQL Standard, además también incluye algunos atributos auxiliares. Esta clase también comprueba que la consulta este bien formada.

Paquete es.scasillas.ocltosql.transformation

Este paquete contiene las clases más importantes de la herramienta que son las que sirven para realizar la transformación:

- *Rules*

Esta clase contiene las reglas de transformación definidas en el punto *5.2 Reglas de transformación*. Esta clase también es la que comprueba si las sentencias OCL están correctamente formadas.

- *OutputOCLXMI*

  Esta clase es la que permite relacionar los datos extraídos del diagrama de clases con las sentencias y a su vez relacionarlo con la clase que permite la transformación de las sentencias. Por otra parte, esta clase también genera el archivo de salida con las transformaciones realizadas y recopila las estadísticas que van a ser mostradas mediante la interfaz gráfica.

6.1.3. Diagramas de secuencia

Para mostrar las relaciones existentes entre las clases diseñadas y los casos de uso se va a proceder a realizar un diagrama de secuencia por cada caso de uso. Como simplificación, se asume que el usuario se encuentra en la vista necesaria para realizar la acción, haciendo referencia a la precondiciones nombradas en los casos de uso.

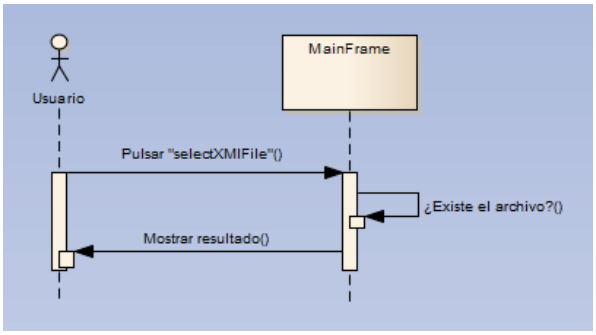


Ilustración 20 - Diagrama de secuencia – Seleccionar archivo diagrama de clases UML

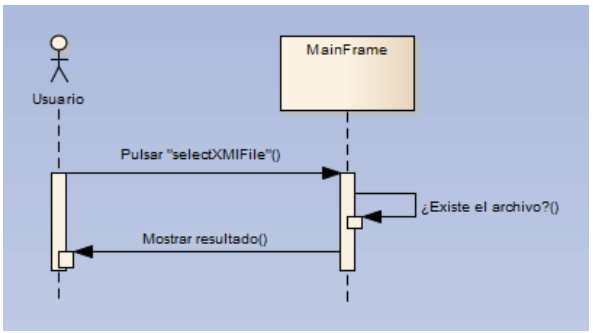


Ilustración 21 - Diagrama de secuencia - Modificar archivo diagrama de clases UML

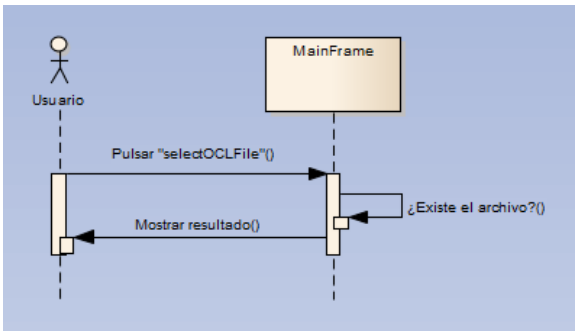


Ilustración 22 - Diagrama de secuencia - Seleccionar archivo restricciones de integridad estáticas OCL

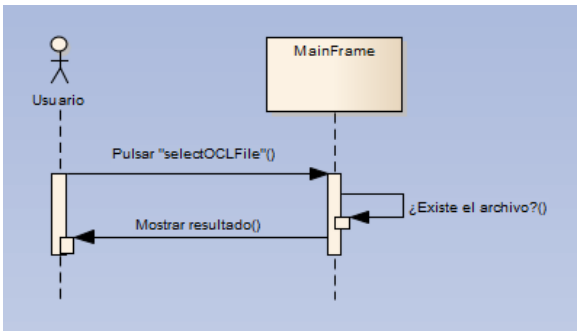


Ilustración 23 - Diagrama de secuencia - Modificar archivo restricciones de integridad estáticas OCL

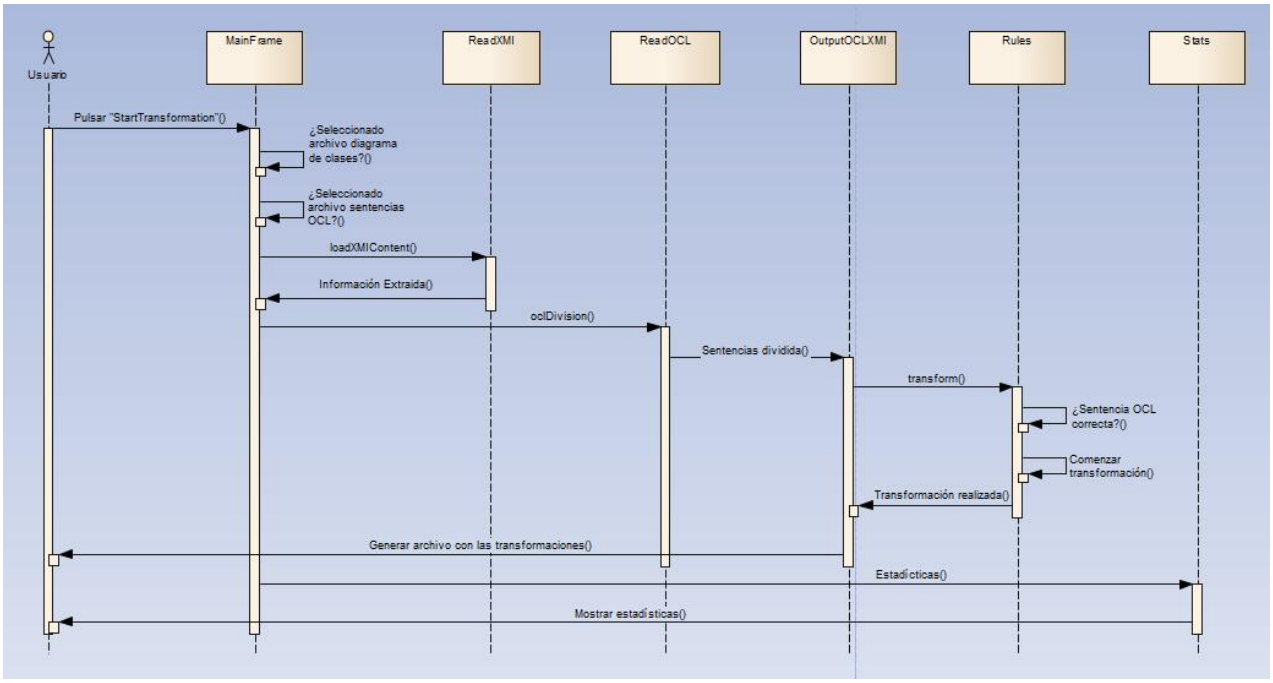


Ilustración 24 - Diagrama de secuencia - Iniciar Transformación

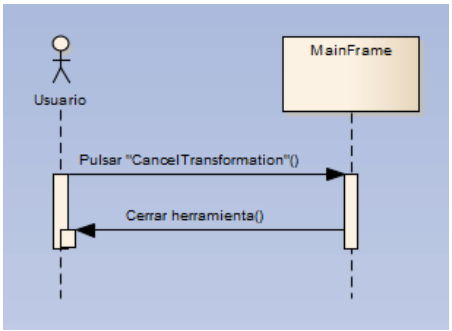


Ilustración 25 - Diagrama de secuencia - Cancelar Transformación

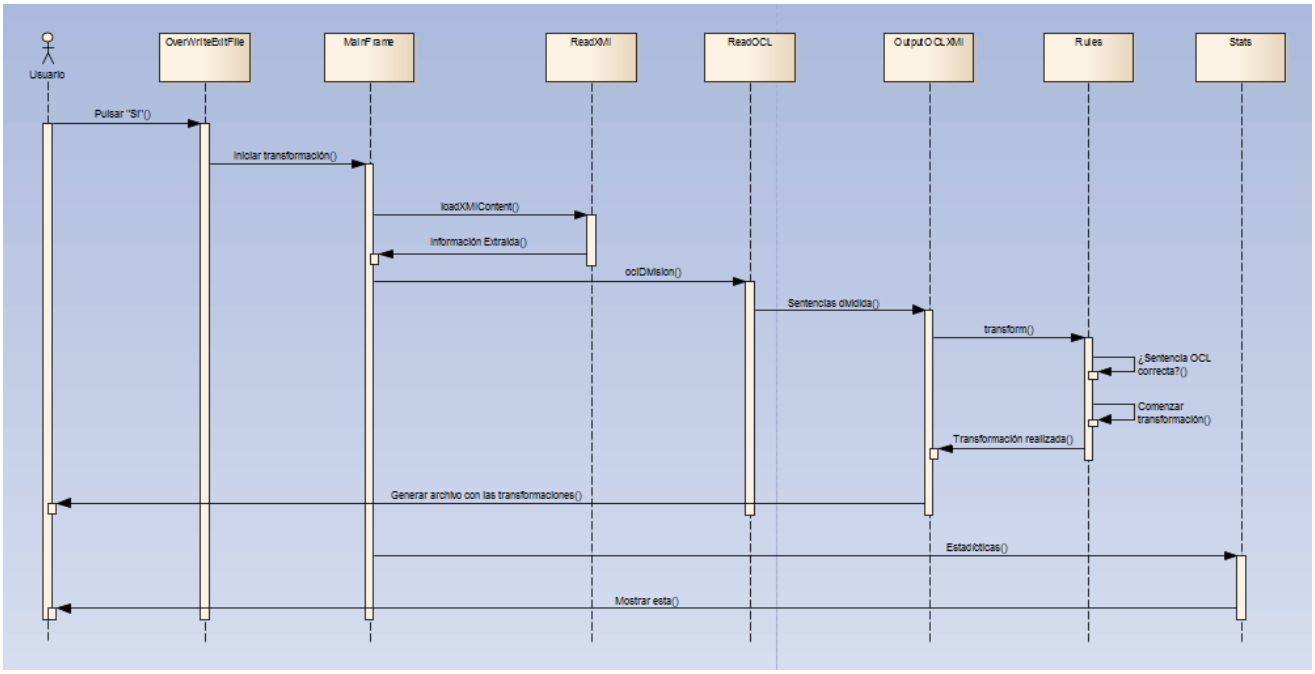


Ilustración 26 - Diagrama de secuencia - Aceptar sobrescribir archivo consultas SQL-Standard

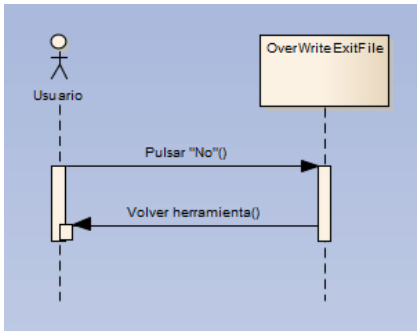


Ilustración 27 - Diagrama de secuencia - Cancelar sobrescribir archivo consultas SQL-Standard

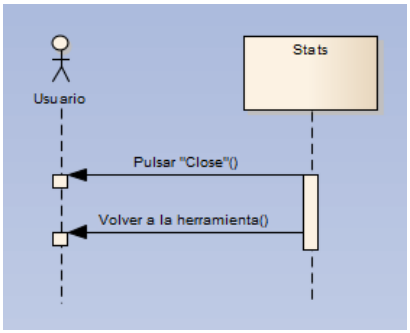


Ilustración 28 - Diagrama de secuencia - Ver estadísticas

6.2. Implementación

6.2.1. Entorno de desarrollo

El desarrollo del trabajo se ha llevado a cabo en un ordenador con las siguientes características.

- Procesador Intel Core i5 M430 2.27GHz.
- 4 GB de RAM.
- 12 GB de disco duro libre.

Y el software necesario para su desarrollo ha sido:

- Sistema operativo Windows 7.
- Java SDK Versión 7 Update 6 (Incluye tanto la versión JDK como la JRE).
- IBM Rational Software Architect.
- Plu-ing Modelado para IBM Rational Software Architect.
- Notepad ++
- Microsoft Office.
- Enterprise Architect.
- Gantt Project.

6.2.2. Descripción de la implementación

A la hora de implementar el sistema, éste se ha dividido en tres subsistemas para que se pudiera implementar de manera más sencilla. Los tres subsistemas se exponen a continuación:

- **Extracción de datos del diagrama de clases UML**

Este subsistema realiza la tarea de extraer las clases, los atributos, las asociaciones y las generalizaciones del diagrama de clases. A continuación se analiza el diagrama de clases en formato XMI explicando cómo se puede identificar cada uno de los elementos que son extraídos.

En primer lugar, aparecen los datos del archivo y el estándar que soporta la especificación del archivo. Esto se define mediante el *tag "uml:Package"*.

```
<uml:Package name="gr81_09" xmi:id="_1O0FmeNEEeGbitkuxMAOsQ" xsi:schemaLocation="http://schema.omg.org/spec/UML/2.2 http://www.eclipse.org/uml2/3.0.0/UML"
xmlns:uml="http://schema.omg.org/spec/UML/2.2" xmlns:ecore="http://www.eclipse.org/emf/2002/Ecore" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xmi="http://schema.omg.org/spec/XMI/2.1" xmi:version="2.1">
```

Ilustración 29 - uml:Package diagrama de clases XMI

En segundo lugar, está la información del tipo de archivo exportado, viene definido mediante el *tag "packageImport"*.

```
<packageImport xmi:id="_1O0FmuNEEeGbitkuxMAOsQ" xmi:type="uml:PackageImport">
    <importedPackage xmi:type="uml:Model" href="http://schema.omg.org/spec/UML/2.2/uml.xml#_0"/>
</packageImport>
```

Ilustración 30 - packageImport diagrama de clases XMI

A partir de ahora vienen todos los elementos definidos en el diagrama de clases. Para identificar cada uno de los elementos hay que basarse en los distintos atributos que tienen. Para identificar una clase, el elemento tiene que empezar por el *tag "packagedElement"* y que el atributo *xmi:type* sea *"uml:Class"*.

```
<packagedElement name="pelicula" xmi:id="_1O0Fm-NEEeGbitkuxMAOsQ" xmi:type="uml:Class">
```

Ilustración 31 - Clase en el diagrama de clases XMI

Como se puede ver la clase tiene dos atributos principales, el nombre (*name)* y un id (*xmi:id*) que permite identificar de manera única a cada elemento dentro del XMI.

Para identificar los atributos que contiene la clase, dentro del *tag "packagedElement"* existe el *tag "ownedAttibute"* que identifica cada uno de los elementos. Para saber si se trata de un atributo de la clase, este elemento tiene que tener el atributo *xmi:type* igual a *"uml:Property"*, y no tiene que tener el atributo *asociation.*

```
<ownedAttribute name="nombre" xmi:id="_FRqypvMGEeGYSMQUh6hW_g" xmi:type="uml:Property" visibility="private">
    <type xmi:type="uml:PrimitiveType" href="http://schema.omg.org/spec/UML/2.2/uml.xml#String"/>
</ownedAttribute>
```

Ilustración 32 - Atributo en el diagrama de clases XMI

Como se puede ver, el elemento tiene dos atributos que lo identifican, el nombre (*name)* y el *xmi:id*.

En el caso de que el elemento que contiene el *tag "ownedAttibute"* tenga un atributo que se llame *"asociation"* significa que se trata de una asociación.

```
<ownedAttribute name="profesor" xmi:id="_FRqyjfMGEeGYSMQUh6hW_g" type="_FRqymPMGEeGYSMQUh6hW_g" xmi:type="uml:Property" visibility="private"
association="_FRqypPMGEeGYSMQUh6hW_g">
    <upperValue xmi:id="_FRqyjvMGEeGYSMQUh6hW_g" xmi:type="uml:LiteralUnlimitedNatural" value="*"/>
    <lowerValue xmi:id="_FRqyj_MGEeGYSMQUh6hW_g" xmi:type="uml:LiteralInteger" value="1"/>
</ownedAttribute>
```

Ilustración 33 - Asociación en el diagrama de clases XMI

En este caso, se puede observar que los atributos que tiene este elementos son: *name* que indica el rol la asociación de la clase hacia la que se navega; *xmi:id*; *type* que contiene el *xmi:id* de la clase hacia la que se navega y *asociation* que contiene un id de asociación. También se puede observar que existen unos subelementos que tienen como *tag "upperValue"* y *"lowerValue"* que indican la multiplicidad de la relación.

Para identificar una generalización, dentro del *tag "packagedElement"* existe un *tag "generalization"*.

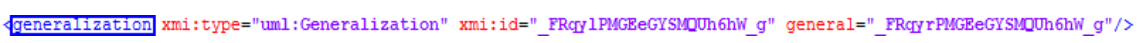


Ilustración 34 - Generalización en diagrama de clases XMI

En este caso, los atributos que tiene el elemento son el atributo *xmi:type* que tiene que ser del tipo *"uml:Generalization"*, el *xmi:id* que identifica al elemento y *general* que contiene el *xmi:id* de la clase que generaliza.

- **Extracción de las sentencias OCL**

Este subsistema realiza la tarea de dividir el archivo con las sentencias OCL y analizar y dividir las sentencias contenidas en el mismo, para que posteriormente puedan ser comprobadas con el diagrama de clases y transformadas a consultas SQL Standard.

Para obtener una a una las sentencias del archivo, lo que se hace es ir analizando palabra a palabra el archivo. Como las sentencias OCL tienen que empezar por *context* significa que desde que encontremos la palabra *context* hasta la siguiente vez que aparezca, se trata de la misma sentencia OCL.

Ahora procedemos a dividir las sentencias para que sea más fácil su transformación. Como se ha visto en el punto *5.1.1. Sentencias OCL*, las sentencias tienen la siguiente estructura.

```
context <nombre_clase> inv <nombre_restricción>:
<expresión_OCL> [OR||AND||XOR||IMPLIES <expresión_OCL>]^n
```

Siendo n un número natural entero.

Para poder dividirla lo primero que se hace es obtener la cabecera que contiene la clase desde la que se parte.

```
context <nombre_clase> inv <nombre_restricción>:
```

Y ahora, con lo que queda de sentencia, lo que se hace es comprueba palabra a palabra, hasta encontrar cualquiera de los operadores OR, AND, XOR o IMPLIES. De manera que quedaría de la siguiente manera:

<expresión_OCL>
[OR||AND||XOR||IMPLIES]^{n}
[<expresión_OCL>]^{n}

Siendo n un número natural entero.

- **Aplicación Reglas de transformación**

Una vez extraídos los datos de ambos archivos, se procede a realizar la transformación de las sentencias OCL. Para ello se sigue el diagrama de estado de la *Ilustración 38*. A continuación explicamos el diagrama de estados.

El primer paso es obtener los datos de la cabecera de la sentencia OCL y crear una consulta que contenga el *nombre_clase* en el FROM.

El segundo paso es analizar la expresión OCL. Primero se analiza si la expresión tiene la estructura correcta, si no la tiene se cancela la transformación y se devuelve el error. Lo siguiente es recorrer la expresión elemento a elemento. Si el elemento es un tipo básico (*5.1.1. Sentencias OCL – Tipos de datos*) o es un operador de estos: <, >, =, <>, <=, >=, el elemento se pone directamente en el WHERE de la consulta. Si no es el caso, se tiene que comprobar si ese elemento tiene alguna operación de tipo colección.

Si no tiene una operación de tipo colección, se tiene que comprobar que el elemento acaba en un atributo y si no lo es se cancela la transformación y se devuelve el error. Se recorre el elemento y se comprueba si es una navegación o es un atributo. Si es una navegación se comprueba si existe esta navegación entre clases. Si existe se aplican las reglas de navegación entre clases y si no tiene se cancela la transformación y se devuelve el error. Si es un atributo, es similar, se comprueba si existe el atributo. Si existe el atributo se aplican las reglas sobre un atributo, en caso contrario se cancela la transformación y se devuelve el error.

Si contiene una operación de tipo colección, se crea una nueva consulta que va a ser comparada en el *WHERE* de la consulta anterior, esto es debido a que si una

expresiónOCL contiene una función de tipo colección, la última función que tenga tiene que devolver *Integer* o *Boolean* para que se pueda comparar con algo. Por lo tanto primero tenemos que identificar si la última función devuelve *Integer* o *Boolean*. En el caso de que no lo haga se cancela la transformación y se devuelve el error.

Si la última función es *Integer* o *Boolean* se tiene que comprobar si las funciones reciben la colección adecuada. En el caso de que no lo haga, se cancela la transformación y se devuelve el error. Si todas reciben las colecciones adecuadas, se recorre el elemento hasta la primera operación de tipo colección y se aplican las reglas de navegación entre clases o sobre un atributo como correspondan, al igual que en el caso anterior.

Después, se recorren las operaciones de tipo colección. Si la operación de tipo colección devuelve *Integer* o *Boolean*, se tiene que comprobar si esta función es la última. Si es la última función, se aplican las reglas de transformación de tipo colección. En caso de que no lo sea, se cancela la transformación y se devuelve el error. Si la función devuelve una colección, aplicamos las reglas de transformación de tipo colección y se sigue recorriendo el elemento.

Otra posibilidad es que el elemento sea uno de estos operadores: *OR, AND, XOR* e *IMPLIES*. En ese caso se aplican las reglas de transformación de la *Tabla 65 - Transformación de los operandos OR, AND, XOR e IMPLIES*.

El tercer y último paso comprueba que las consultas SQL Standard estén bien generadas

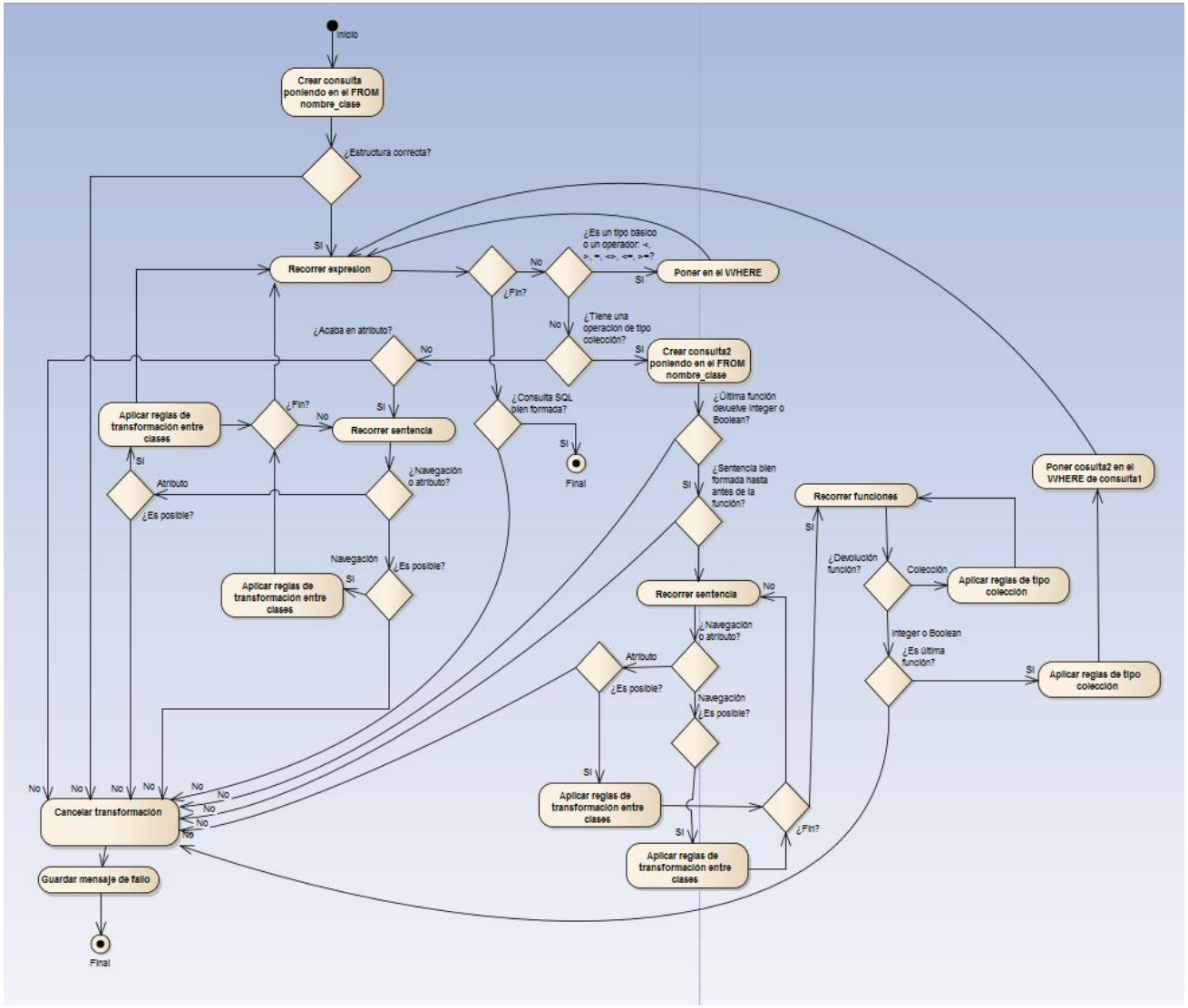


Ilustración 35 - Diagrama de estado de la implementación desarrollada

7.Pruebas

En este capítulo se definen las pruebas realizadas para comprobar que el funcionamiento de la herramienta es correcto

7.1. Pruebas de funcionalidad

Para asegurar que el producto final cumple con los requisitos de usuario, se realizan las siguientes pruebas de funcionalidad.

Antes de comenzar se procede a definir la información de las pruebas.

- Identificador de la prueba: formado por PA- y tres dígitos
- ID del requisito de usuario
- Resumen del requisito
- Especificaciones de entrada
- Especificaciones de salida
- Criterio de aceptación

Las pruebas se han realizado en el entorno de desarrollo descrito en el punto *6.2.1. Entorno de desarrollo.*

Para que la prueba se considere como aceptada, se tendrá que cumplir el criterio de aceptación. Esto implica que el sistema se comporta como se esperaba.

PA-001	
Requisito de usuario	RU-C001, RU-C002, RU-C004 y RU-C005
Resumen requisito	Se tiene que poder seleccionar/modificar el archivo con el diagrama de clases y el archivo con las sentencias OCL.
Especificaciones de entrada	Se selecciona un archivo con un diagrama de clases UML.
Especificaciones de salida	Se guarda la ruta del archivo que aparece en la interfaz.
Criterio de aceptación	La interfaz muestra el archivo seleccionado.

Tabla 69 - PA-001

PA-002	
Requisito de usuario	RU-C001, RU-C002, RU-C004 y RU-C005
Resumen requisito	Se tiene que poder seleccionar/modificar el archivo con el diagrama de clases y el archivo con las sentencias OCL.
Especificaciones de entrada	Se selecciona un archivo con sentencias OCL.
Especificaciones de salida	Se guarda la ruta del archivo que aparece en la interfaz.
Criterio de aceptación	La interfaz muestra el archivo seleccionado.

Tabla 70 - PA-002

PA-003	
Requisito de usuario	RU-C001, RU-C002, RU-C004 y RU-C005
Resumen requisito	Se tiene que poder seleccionar/modificar el archivo con el diagrama de clases y el archivo con las sentencias OCL.
Especificaciones de entrada	Se selecciona un archivo cualquiera.
Especificaciones de salida	Se guarda la ruta del archivo que aparece en la interfaz.
Criterio de aceptación	La interfaz muestra el archivo seleccionado.

Tabla 71 - PA-003

PA-004	
Requisito de usuario	RU-C001, RU-C002, RU-C004 y RU-C005
Resumen requisito	Se tiene que poder seleccionar/modificar el archivo con el diagrama de clases y el archivo con las sentencias OCL.
Especificaciones de entrada	Se selecciona un archivo manualmente.
Especificaciones de salida	Se guarda la ruta del archivo que aparece en la interfaz.
Criterio de aceptación	La interfaz muestra el archivo seleccionado.

Tabla 72 - PA-004

PA-005	
Requisito de usuario	RU-C003
Resumen requisito	Se tienen que validar las sentencias OCL con el diagrama de clases.
Especificaciones de entrada	Archivo con el diagrama de clases y archivo con sentencias OCL de todo tipo.
Especificaciones de salida	Archivo con consultas SQL Standard o los errores producidos en las asociaciones o atributos.
Criterio de aceptación	Todas las sentencias OCL que no tenga asociaciones o atributos válidos tienen que salir con error.

Tabla 73 - PA-005

PA-006	
Requisito de usuario	RU-C006
Resumen requisito	Se tiene que permitir iniciar la transformación.
Especificaciones de entrada	Ningún archivo seleccionado.
Especificaciones de salida	Información de que no tienes el archivo con el diagrama de clases seleccionado.
Criterio de aceptación	La interfaz muestra un aviso diciendo que para iniciar la transformación necesitas el archivo con el diagrama de clases seleccionado.

Tabla 74 - PA-006

PA-007	
Requisito de usuario	RU-C006
Resumen requisito	Se tiene que permitir iniciar la transformación.
Especificaciones de entrada	Seleccionado el archivo con el diagrama de clases.
Especificaciones de salida	Información de que no tienes el archivo con las sentencias OCL.
Criterio de aceptación	La interfaz muestra un aviso diciendo que para iniciar la transformación necesitas el archivo con las sentencias OCL.

Tabla 75 - PA-007

PA-008	
Requisito de usuario	RU-C006
Resumen requisito	Se tiene que permitir iniciar la transformación.
Especificaciones de entrada	Seleccionado el archivo con las sentencias OCL.
Especificaciones de salida	Información de que no tienes el archivo con el diagrama de clases seleccionado.
Criterio de aceptación	La interfaz muestra un aviso diciendo que para iniciar la transformación necesitas el archivo con el diagrama de clases seleccionado.

Tabla 76 - PA-008

PA-009	
Requisito de usuario	RU-C006
Resumen requisito	Se tiene que permitir iniciar la transformación.
Especificaciones de entrada	Seleccionado el archivo con el diagrama de clases y el archivo con las sentencias OCL.
Especificaciones de salida	Trasformación de las sentencias OCL a consultas SQL Standard.
Criterio de aceptación	Se crea un archivo con las consultas SQL Standard o los errores surgidos.

Tabla 77 - PA-009

PA-010	
Requisito de usuario	RU-C007
Resumen requisito	Se tiene que permitir cancelar la transformación.
Especificaciones de entrada	-
Especificaciones de salida	La herramienta se tiene que cerrar.
Criterio de aceptación	La herramienta finaliza su ejecución.

Tabla 78 - PA-010

PA-011	
Requisito de usuario	RU-C008 y RU-C009
Resumen requisito	La herramienta avisara cuando el archivo de salida vaya a ser sobrescrito.
Especificaciones de entrada	El usuario pulsa iniciar la transformación y el archivo de salida no existe.
Especificaciones de salida	Se crea un archivo con las transformaciones.
Criterio de aceptación	Se crea un archivo con el nombre correcto y este contiene las transformaciones realizadas a las sentencias OCL.

Tabla 79 - PA-011

PA-012	
Requisito de usuario	RU-C008
Resumen requisito	La herramienta avisara cuando el archivo de salida vaya a ser sobrescrito.
Especificaciones de entrada	El usuario pulsa iniciar la transformación y el archivo de salida existe.
Especificaciones de salida	Información de que el archivo va de salida va a ser sobrescrito.
Criterio de aceptación	La interfaz muestra un aviso indicando si se quiere o no sobrescribir el archivo.

Tabla 80 - PA-012

PA-013	
Requisito de usuario	RU-C010
Resumen requisito	Tienen que aparecer las estadísticas después de realizar las transformaciones.
Especificaciones de entrada	El usuario pulsa iniciar la transformación
Especificaciones de salida	Información sobre las estadísticas producidas en la transformación de las sentencias OCL.
Criterio de aceptación	La interfaz muestra un aviso indicando cuales son las estadísticas obtenidas en la transformación de las sentencias OCL.

Tabla 81 - PA-013

7.2. Pruebas de aceptación

Además de realizar pruebas para comprobar que cumple con las funcionalidades definidas en los requisitos de usuario, también se han realizado pruebas de datos masivas para comprobar que los requisitos de software de aceptación también se cumplen. En este tipo de pruebas se tiene en cuenta la información que contienen los archivos de entrada, el archivo de salida y las estadísticas obtenidas.

PA-014	
Descripción	Comprobar si las restricciones con navegación y sobre un atributo funcionan.
Especificación diagrama de clases	Cinco diagramas de clases de distintos ámbitos.
Especificación sentencias OCL	250 sentencias OCL bien formadas para cada diagrama.
Estadísticas obtenidas	100% de transformaciones correctas.
Criterio de aceptación	Se tiene que generar un archivo de salida con las consultas SQL Standard o los errores surgidos. El porcentaje de errores no contralados tiene que ser inferior al 1% y el de transformaciones correctas superior al 75%.

Tabla 82 - PA-014

PA-015	
Descripción	Comprobar si las operaciones de tipo colección que devuelve un Integer funcionan.
Especificación diagrama de clases	Cinco diagramas de clases de distintos ámbitos.
Especificación sentencias OCL	285 sentencias OCL con operaciones de tipo colección que devuelve un Integer.
Estadísticas obtenidas	97% de transformaciones correctas, 3% con errores controlados y 0% con errores no controlados.
Criterio de aceptación	Se tiene que generar un archivo de salida con las consultas SQL Standard o los errores surgidos. El porcentaje de errores no contralados tiene que ser inferior al 1% y el de transformaciones correctas superior al 75%.

Tabla 83 - PA-015

PA-016	
Descripción	Comprobar si las operaciones de tipo colección que devuelve un Boolean funcionan.
Especificación diagrama de clases	Cinco diagramas de clases de distintos ámbitos.
Especificación sentencias OCL	120 sentencias OCL con operaciones de tipo colección que devuelve un Boolean.
Estadísticas obtenidas	100% de transformaciones correctas
Criterio de aceptación	Se tiene que generar un archivo de salida con las consultas SQL Standard o los errores surgidos. El porcentaje de errores no contralados tiene que ser inferior al 1% y el de transformaciones correctas superior al 75%.

Tabla 84 - PA-016

PA-017	
Descripción	Comprobar si las operaciones de tipo colección que devuelve una colección funcionan.
Especificación diagrama de clases	Diagrama de clases completo.
Especificación sentencias OCL	200 sentencias OCL con operaciones de tipo colección que devuelve una colección.
Estadísticas obtenidas	93% de transformaciones correctas, 7% con errores controlados y 0% con errores no controlados.
Criterio de aceptación	Se tiene que generar un archivo de salida con las consultas SQL Standard o los errores surgidos. El porcentaje de errores no contralados tiene que ser inferior al 1% y el de transformaciones correctas superior al 75%.

Tabla 85 - PA-017

PA-018	
Descripción	Comprobar que la herramienta soporta distintos diagramas de clase.
Especificación diagrama de clases	Dos diagramas de clase distintos.
Especificación sentencias OCL	250 sentencias OCL de navegación bien formadas para cada uno de los diagramas de clase.
Estadísticas obtenidas	100% de transacciones correctas.
Criterio de aceptación	Se tiene que generar un archivo de salida con las consultas SQL Standard o los errores surgidos. El porcentaje de errores no contralados tiene que ser inferior al 1% y el de transformaciones correctas superior al 75%.

Tabla 86 - PA-018

PA-019	
Descripción	Comprobar con un número elevado de sentencias.
Especificación diagrama de clases	Dos diagramas de clases completos.
Especificación sentencias OCL	973 sentencias OCL de todo tipo.
Estadísticas obtenidas	912 sentencias correctamente transformadas, 71 con errores controlados y 0 con errores no controlados.
Criterio de aceptación	Se tiene que generar un archivo de salida con las consultas SQL Standard o los errores surgidos. El porcentaje de errores no contralados tiene que ser inferior al 1% y el de transformaciones correctas superior al 75%.

Tabla 87 - PA-019

A continuación se expone lo que significa cada una de las salidas posibles para una transformación:

- Sentencias correctamente transformadas – Estas sentencias son las que se han comprobado que están bien formadas respecto al diagrama de clases, se ha verificado que cumplen los requisitos definidos en las reglas de transformación y por último, se ha comprobado que la consulta SQL Standard generada está bien formada.

- Sentencias con errores controlados – Estas sentencias son las que han producido un fallo mientras se comprobaba si estaban bien formadas con el diagrama de clases o no cumplía los requisitos definidos en las reglas de transformación o a la hora de comprobar la consulta SQL Standard, ésta ha dado errónea.
- Sentencias con errores no controlados – Estas sentencias son las que han producido un error no controlado (excepción) en Java.

A continuación se muestran tres tablas resumen, una con las características de los diagramas de clase utilizados para las pruebas, otra con los resultados obtenidos en la transformación de las sentencias definidas en las pruebas y otra con los resultados obtenidos divididos en los tipos de restricciones definidos.

Diagrama de clases	Clases	Asociaciones	Generalizaciones	Restricciones sobre un objeto de una clase	Restricciones con navegación	Restricciones con navegación más una función de agregación
Biblioteca	8	7	1	221	328	434
Comercio electrónico	8	6	2	175	143	522
Universidad	6	6	1	98	102	321
Animales	5	3	3	115	127	317
Aplicación Android	7	7	0	193	174	430

Tabla 88 - Pruebas - Características

Diagrama de clases	Restricciones sobre un objeto de una clase				Restricciones sobre un objeto de una clase				Restricciones sobre un objeto de una clase			
	Correctas	Error controlado	Error no controlado	Total	Correctas	Error controlado	Error no controlado	Total	Correctas	Error controlado	Error no controlado	Total
Biblioteca	221 (100%)	0 (0%)	0 (0%)	221	328 (100%)	0 (0%)	0 (0%)	328	363 (83,6%)	71 (16,4%)	0 (0%)	434
Comercio electrónico	175 (100%)	0 (0%)	0 (0%)	175	143 (100%)	0 (0%)	0 (0%)	143	459 (87,9%)	63 (12,1%)	0 (0%)	522
Universidad	98 (100%)	0 (0%)	0 (0%)	98	102 (100%)	0 (0%)	0 (0%)	102	264 (82,3%)	56 (17,4%)	1 (0,3%)	321
Animales	115 (100%)	0 (0%)	0 (0%)	115	127 (100%)	0 (0%)	0 (0%)	127	266 (83,9%)	51 (16,1%)	0 (0%)	317
Aplicación Android	193 (100%)	0 (0%)	0 (0%)	193	4 (100%)	0 (0%)	0 (0%)	174	363 (84,4%)	67 (15,6%)	0 (0%)	430
Total	**100%**	**0%**	**0%**	**802**	**100%**	**0%**	**0%**	**874**	**84,73%**	**15,22%**	**0,05%**	**2024**

Tabla 89 - Pruebas - Resumen por tipos de restricción

Diagrama de clases	Transformaciones correcta	Transformaciones con error controlado	Transformaciones con error no controlado
Biblioteca	912 (92,8%)	71 (7,2%)	0 (0%)
Comercio electrónico	777 (92,5%)	63 (7,5%)	0 (0%)
Universidad	464 (89,1%)	56 (10,8%)	1 (0,1%)
Animales	508 (90,9%)	51 (9,1%)	0 (0%)
Aplicación Android	730 (91,6%)	67 (8,4%)	0 (0%)
Total	**91,65%**	**8,32%**	**0,03%**

Tabla 90 - Pruebas - Resumen por resultado

Analizando los resultados, las sentencias que no se han transformado correctamente no superan el 10%. Todas estas sentencias que han producido un error controlado, son sentencias que tienen operaciones con funciones de agregación. Existen dos tipos de fallos:

El primer fallo se debe a que si se concatenan muchas funciones de agregación, la herramienta produce una sentencia tan amplia que es muy difícil de manejar para realizar la siguiente función de agregación.

El segundo fallo que se produce es debido a que si una función de agregación recibe como parámetro una expresión que tiene otra función de agregación a la que se le pasa un parámetro que tiene otra expresión con otra función de agregación y así sucesivas veces. La herramienta no tiene capacidad para manejar tantas consultas SQL y posteriormente unirlas.

En resumen, de un total de 3700 restricciones analizadas (802 sobre un objeto de una clase, 874 con navegación y 2024 con navegación más una función de agregación) el 91,65 % se transforman correctamente, 8,32% se transforman con errores controlados y el 0,03% con errores no controlados. Por lo tanto, se supera ampliamente las metas propuestas del 75% de sentencias transformadas correctamente y un máximo del 1% en errores no controlados.

8. Presupuesto

A continuación se expone el presupuesto del sistema creado. Este presupuesto está dividido en tres partes: costes fijos, costes variables y costes indirectos.

8.1. Recursos materiales

En este apartado se tendrá en cuenta el hardware y el software que se ha necesitado para crear el sistema. Se ha tenido en cuenta que el material tiene un periodo de amortización, por lo que en el presupuesto solo se incluirá la parte proporcional al periodo de desarrollo que se ha estimado en 15 semanas.

Elemento	Cantidad	Coste	Tiempo de uso	Periodo de amortización	Subtotal
Ordenador	**1**	**600 €**	**15 semanas**	**3 años**	**57,70 €**
Microsoft Windows 7	**1**	**0 € (Licencia para estudiantes)**	**15 semanas**	**-**	**0 €**
Microsoft Office 2010	**1**	**99 €**	**15 semanas**	**-**	**9,52 €**
Rational Software Architect	**1**	**3.068 €**	**15 semanas**	**-**	**295 €**
Total					**362,22 €**

Tabla 91 - Recursos materiales

8.2. Recursos humanos

En este apartando se analizaran las horas empleadas en el desarrollo del sistema en cada una de sus fases. Tal y como se dijo en el punto 3.3. Organización del trabajo, la media diaria de horas trabajadas es de 4 horas. Teniendo en cuenta este dato, se expone la siguiente tabla con la distribución de horas en cada fase del proyecto.

Según este estudio realizado por la Universidad Carlos III [26], los recién graduados obtienen un ingreso aproximado de 1.600 €/netos el primer año. Por lo tanto el precio por hora, suponiendo que un mes tiene 160 horas laborables, es de 10 €/hora.

Fase	Precio/hora	Horas	Subtotal
Análisis del entorno	10 €	68	680 €
Gestión del proyecto	10 €	12	120 €
Análisis del sistema	10 €	64	640 €
Diseño del sistema	10 €	40	400 €
Implementación	10 €	96	960 €
Pruebas	10 €	20	200 €
Documentación	10 €	48	480 €
Total		348	3.480 €

Tabla 92 - Recursos humanos

8.3. Costes indirectos

En este apartado se ha establecido una lista con los costes indirectos aplicables al trabajo. Solo se han tenido en cuenta los gastos derivados de la luz y de la cuota de Internet.

Concepto	Precio/mes	Tiempo de uso	Subtotal
Conexión a Internet	50 €	4 meses	200 €
Luz	70 €	4 meses	280 €
Total			480 €

Tabla 93 - Costes indirectos

8.4. Costes totales

Para obtener la suma total del presupuesto, hay que realizar la suma de las partes descritas anteriormente y sumarle el I.V.A. que se le aplica al proyecto. Siendo el resultado:

Recurso	Coste
Recursos materiales	**362,22 €**
Recursos humanos	**3.480,00 €**
Costes indirectos	**480,00 €**
Subtotal	**4.322,22 €**
I.V.A. (21 %)	**907,67 €**
Total	**5.229,89 €**

Tabla 94 - Presupuesto total

9. Conclusión

En este capítulo se van a definir las conclusiones obtenidas tras la realización de este Trabajo de Fin de Grado y sus posibles trabajos futuros.

9.1. Conclusiones generales

En este Trabajo de fin de grado se ha llevado a cabo el análisis, diseño e implementación de una herramienta capaz de transformar sentencias OCL a consultas SQL Standard. Aunque existen varios estudios que han concluido en resultados importantes como la creación de las herramientas existentes en mercado, creemos que en el contexto de las bases de datos, hay necesidad de desarrollar más herramientas para tratar el tema de las restricciones de integridad.

Esta herramienta se ha realizado para cubrir ciertos vacios que las herramientas actuales no son capaces de solventar como la transformación de las sentencias SQL correctamente en función a un diagrama de clases.

Esta herramienta también se ha realizado pensando en las facilidades que les va a ofrecer a los desarrolladores de bases de datos, dado que hace más fácil el diseño de una base de datos a partir de un diagrama de clases donde se puedan definir las restricciones. Por otra parte, en el caso de que estas restricciones de integridad se modificasen o el diagrama se escalase, las sentencias SQL se podrían cambiar de manera automática permitiendo un ahorro de trabajo.

Aun permitiendo la transformación de un gran número de sentencias, esta herramienta muestra ciertas limitaciones dado que no permite transformar todos los tipos de restricciones OCL existentes, tales como, pre-condiciones o post-condiciones. Esta herramienta también tiene la limitación de que el usuario no puede definir las sentencias sobre el diagrama de clases sino que tiene realizarlas de forma manual, lo que provoca que los usuarios tengan que tener unos conocimientos mínimos sobre el lenguaje OCL.

En lo referente al aprendizaje personal, puedo decir que este Trabajo de Fin de Grado me ha puesto un reto de análisis, diseño y desarrollo, puesto que ha sido

la primera vez que me enfrentaba a un proyecto de esta magnitud. También ha servido para el aprendizaje de nuevas maneras de definir una base de datos y las amplias posibilidades que ofrece un diagrama de clases.
Parta terminar decir que se va a liberar el código de la herramienta con licencia libre, aunque todavía se tiene que definir qué tipo de licencia elegir, para quien quiere, pueda seguir desarrollando la herramienta para que ofrezca más posibilidades o funcionalidad.

9.2. Trabajos futuros

En relación a los posibles trabajos futuros, podemos destacar los siguientes:

- Ampliar el rango de sentencias OCL que permite transformar, no centrándose únicamente en las restricciones de integridad estáticas sino también permitiendo la transformación de las restricciones de integridad dinámicas.
- Realización de un análisis de las posibles optimizaciones que se pueden realizar sobre la herramienta, como pueden ser, optimización de las consultas SQL o aumento del porcentaje de transformaciones realizadas satisfactoriamente.
- Dado que el lenguaje OCL es complejo y los diseñadores nuevos no lo aplican, se propone desarrollar un modulo que permita a los usuarios menos expertos introducir las sentencias OCL en el propio diagrama de clases.
- Acoplar una herramienta que permita la creación de la base de datos a partir del diagrama de clases UML. Una vez realizada esta acción se podría permitir al usuario la posibilidad de introducir los datos de conexión a un sistema gestor de bases de datos para pudiera ejecutar las consultas generadas y comprobar si estas están correctamente formadas.
- Realizar un análisis sobre la usabilidad que tiene la herramienta desarrollada. Este análisis permitirá conocer cómo es posible adaptar la herramienta para que esta sea más fácil de utilizar para los usuarios.

10. Bibliografía

[1] Object Management Group: Unified Modeling Language, Mar 2011 [en línea]

<http://www.omg.org/spec/UML/2.2> [Consulta: Mayo, 2012].

[2] ISO/TC97/SC5/WG3, "Concepts and Terminology for the Conceptual Schema and Information Base", ISO 1982. [Consulta: Mayo, 2012].

[3] Object Management Group: Object Constraint Language, Feb 2010 [en línea]

<http://www.omg.org/spec/OCL/2.0> [Consulta: Mayo, 2012].

[4] SQL Standard ANSI/ISO/IEC 9075:2011 [en línea]

<http://www.iso.org/iso/home/store/catalogue_tc/catalogue_tc_browse.htm?commid=45342> [Consulta: Mayo, 2012].

[5] Acceleo/OCL Operations Reference [en línea]

<http://wiki.eclipse.org/Acceleo/OCL_Operations_Reference#Ocl_operations_for_type_.2ACollection.2A> [Consulta: Mayo, 2012].

[6] Dresden OCL Toolkit

<http://dresdenocl.sourceforge.net/index.html> [Consulta: Mayo, 2012].

[7] B. Demuth, H. Hussmann, and S. Loecher, "OCL as a Specification Language for Business Rules in Database Applications", 2001. [Consulta: Mayo, 2012].

[8] Borland, Bold for Delphi

<http://info.borland.com/techpubs/delphi/boldfordelphi/> [Consulta: Mayo, 2012].

[9] MySQL4OCL: un compilador de OCL a MySQL

<http://eprints.ucm.es/13800/> [Consulta: Mayo, 2012].

[10] MySQL4OCL code generator

<http://www.bm1software.com/mysql-ocl/tables.html> [Consulta: Mayo, 2012].

[11] H.T. Al-Jumaily, D. Cuadra, and P. Martínez, "OCL2Trigger: Deriving active mechanisms for relational databases using Model-Driven Architecture", pp.2299-2314, 2008 [Consulta: Mayo, 2012].

[12] Java Development Kit Version 7 Update 6

<http://www.oracle.com/technetwork/java/javase/downloads/index.html> [Consulta: Mayo, 2012].

[13] PSS download page [en línea]

<http://www.esa.int/TEC/Software_engineering_and_standardisation/TECBUCUXBQE_2.html> [Consulta: Mayo, 2012].

[14] Object Management Group: MOF/XMI Mapping, Agosto 2011. OMG [en línea]

<http://www.omg.org/spec/XMI/2.4.1/> [Consulta: Mayo, 2012].

[15] Validating and Improving Test-Case Effectiveness, Enero/Febrero 2011 [en línea]

<http://www.inf.utfsm.cl/~visconti/titulacion/ValidacionMejoramientoEfectividadPruebas.pdf> [Consulta: Mayo, 2012].

[16] Contenido mínimo del Trabajo de Fin de Grado, Enero 2012 [en línea]

<https://aulaglobal2.uc3m.es/file.php/32039/Indicaciones_elaboracion_TFG.pdf> [Consulta: Mayo, 2012].

[17] American National Standards Institute

<http://www.ansi.org/> [Consulta: Mayo, 2012].

[18] MySQL

<http://www.mysql.com/> [Consulta: Mayo, 2012].

[19] Oracle

<http://www.oracle.com/> [Consulta: Mayo, 2012].

[20] PostgreSQL

<http://www.postgresql.org> [Consulta: Mayo, 2012].

[21] SQLite

<http://www.sqlite.org/> [Consulta: Mayo, 2012].

[22] Microsoft SQL Server

<http://www.microsoft.com/sqlserver/> [Consulta: Mayo, 2012].

[23] J. Casas Cuevas, M. Arenas Fernández, "OCL (Object Constraint Language)", Enero 2003 [en línea]

<http://users.dsic.upv.es/asignaturas/facultad/lsi/trabajos/132000.doc> [Consulta: Mayo, 2012].

[24] Object Management Group

<http://www.omg.org> [Consulta: Mayo, 2012].

[25] Object Management Group: Model Driven Architecture Guide v. 1.0.1.Informe técnico, OMG, 2003 [en línea]

<http://www.omg.org/cgi-bin/doc?omg/03-06-01. Cap 2> [Consulta: Mayo, 2012].

[26] XV Estudio de Inserción Profesional de los Titulados de la Universidad Carlos III de Madrid Promoción 2009, Mayo 2011 [en línea]

<http://www.uc3m.es/portal/page/portal/sopp/XIV_Estudio_Insercion_Profesional/estudio%20EIPXV_FOLLETO.pdf> [Consulta: Mayo, 2012].

11. Anexo I: Acrónimos y abreviaturas

ANSI: *American National Standars Institute, Instituto Nacional Estadounidense de Estándares.*

API: *Application Programming Interface, Interfaz de Programación de Aplicaciones.*

DBMS: *DataBase Management System, Sistema de Gestión de Bases de Datos.*

DDL: *Data Definition Language, Lenguaje de Definición de Datos.*

DML: *Data Management Language, Lenguaje de Manipulación de datos.*

ESA: *European Space Agency, Agencia Espacial Europea.*

GB: *Giga Byte.*

GHz: *Giga Hercio.*

IBM: *International Business Machines.*

IVA: *Impuesto sobre el Valor Añadido.*

JDK: *Java Runtime Development, Entorno de Desarrollo de Java.*

JRE: *Java Runtime Environement, Entorno de Ejecución de Java.*

MDA: *Model-Driven Architecture, Arquitectura Dirigida por Modelos.*

OCL: *Object Constraint Language, Lenguaje de Especificación de Objetos.*

OMG: *Object Management Group.*

PIM: *Platform Independent Mode, Modelo Independiente de la Plataforma.*

PSM: *Platform Specific Model, Modelo Específico de la Plataforma.*

RAM: *Random Access Memory, Memoria de Acceso Aleatorio.*

RDB: *Relational DataBase, Base de Datos Relacional.*

SDK: *Software Development Kit, Kit de Desarrollo de Software.*

SQL: *Structured Query Language, Lenguaje de Consulta Estructurado.*

UML: *Unified Modeling Language, Lenguaje Unificado de Modelado.*

***URI**: Uniform Resource Identifier, Identificador Uniforme de Recurso.*

XMI: *XML Metadata Interchange, Intercambio de Metadatos XML.*

12. Anexo II: Manual de usuario

En este manual se explican la forma de realizar una transformación de manera correcta.

En primer lugar el usuario tiene que iniciar la herramienta. La pantalla que le aparecerá será la siguiente:

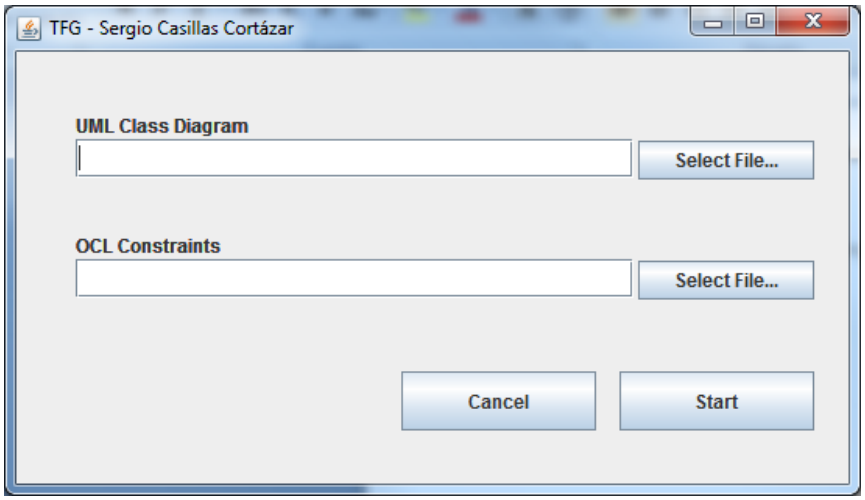


Ilustración 36 - Ventana principal de la herramienta

Para poder iniciar la transformación, el usuario tendrá que seleccionar el archivo con el diagrama de clases UML y el archivo con las sentencias OCL.

Para seleccionar el diagrama de clases UML, tenemos que pulsar sobre el botón "Select File…" que está a la derecha de la etiqueta "UML Class Diagram". Una vez pulsamos este botón, se mostrará un gestor de ficheros que permite seleccionar el fichero con el diagrama de clases, por defecto tiene van a mostrar solo los archivos cuya extensión es ".xmi", para modificarlo tenemos que pulsar sobre el botón desplegable que encuentra a la derecha de la etiqueta "Archivos de tipo:".

Para seleccionar el archivo con las sentencias OCL, es un caso similar, pulsamos el botón "Select File…" que está a la derecha de la etiqueta "OCL Constraints". Este mostrará el mismo gestor de ficheros que en el caso anterior pero, en este caso, por defecto, solo se mostrarán los archivos cuyas extensiones sean ".ocl".

A continuación se muestran sendas imágenes para poder seleccionar los archivos.

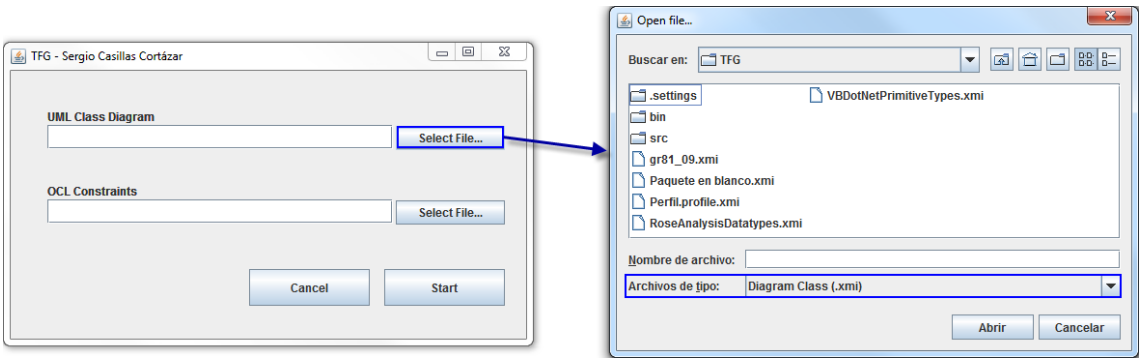


Ilustración 37 - Seleccionar archivo con el diagrama de clases UML

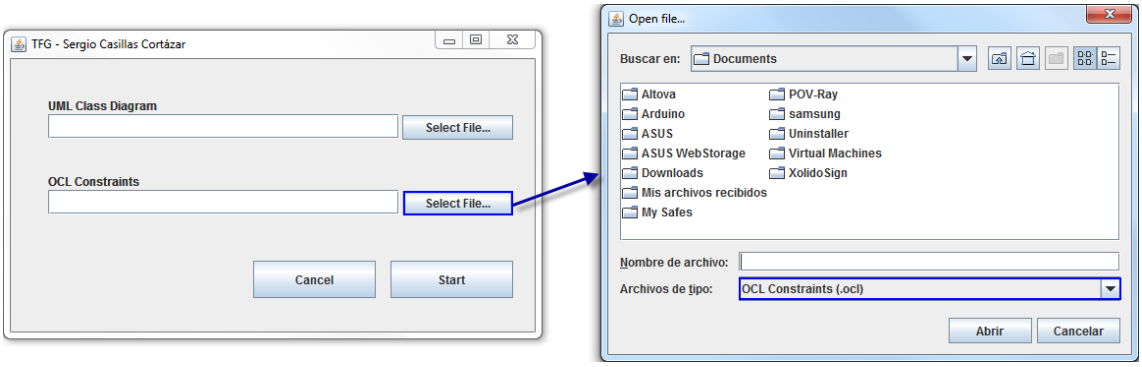


Ilustración 38 - Seleccionar archivo con las sentencias OCL

Una vez seleccionamos el archivo que deseamos abrir, se mostrará la ruta del archivo debajo de su etiqueta para cada caso. La siguiente imagen muestra el resultado.

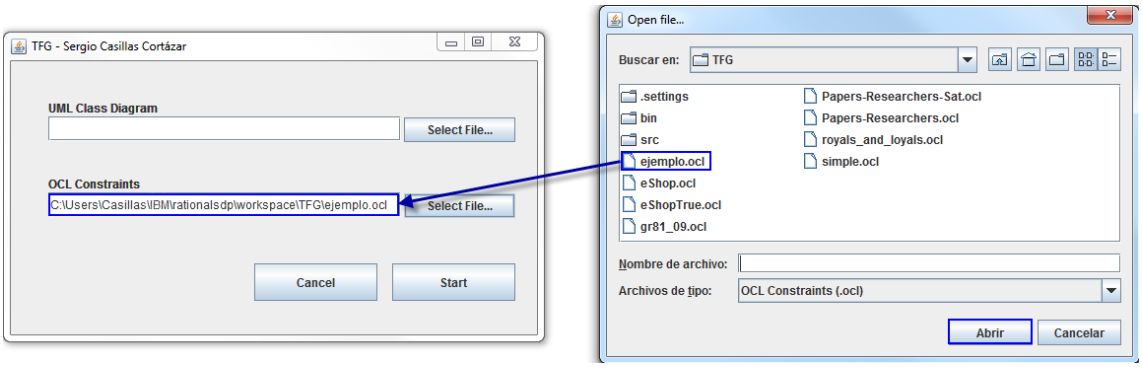


Ilustración 39 - Vista de la selección del archivo en la ventana principal

Hasta que no tengamos ambos archivos seleccionados, aunque queramos iniciar la transformación la herramienta, ésta no nos lo permitirá y nos mostrará un aviso diciéndonos el archivo que falta. Se muestra un ejemplo a continuación.

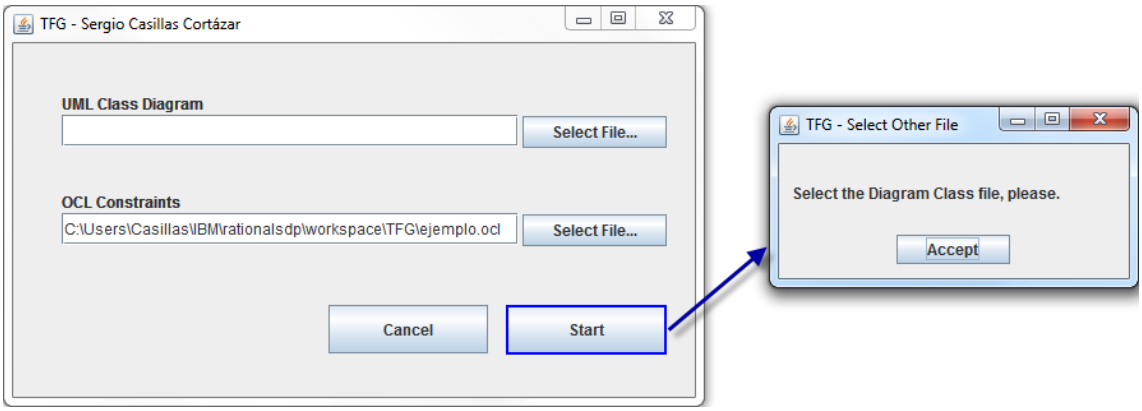


Ilustración 40 - Iniciar la transformación - Faltan archivos

Una vez que se tengan los dos archivos seleccionados, si pulsamos el botón "Start", en el caso de que exista un archivo con el mismo nombre que el archivo que contiene las sentencias OCL pero con la extensión ".slq", se mostrará un aviso preguntado si se desea sobrescribir el archivo. A continuación se muestra un ejemplo.

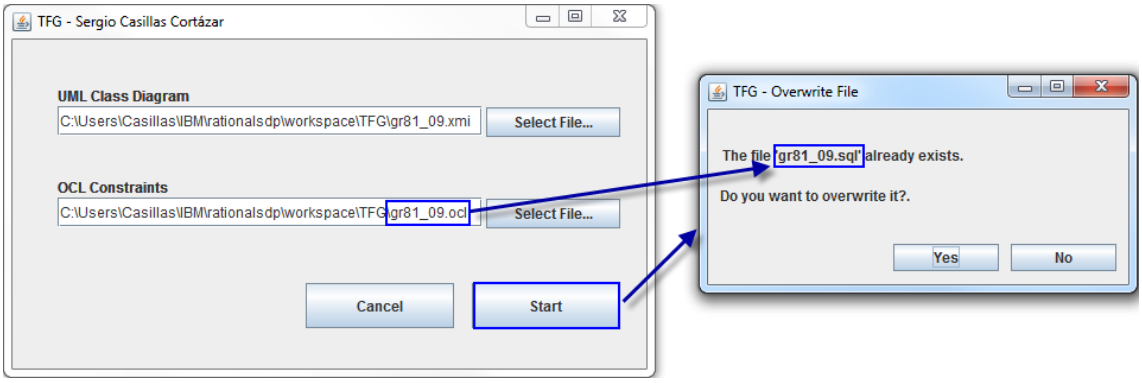


Ilustración 41 - Iniciar la transformación - Información archivo de salida sobrescrito

Si seleccionamos que no deseamos sobrescribir el archivo se volverá a la vista anterior. Si pulsamos que sí, al igual que si no hubiera existido el archivo de salida, si iniciará el proceso de transformación y una vez finalice se mostrarán las estadísticas obtenidas con la transformación.

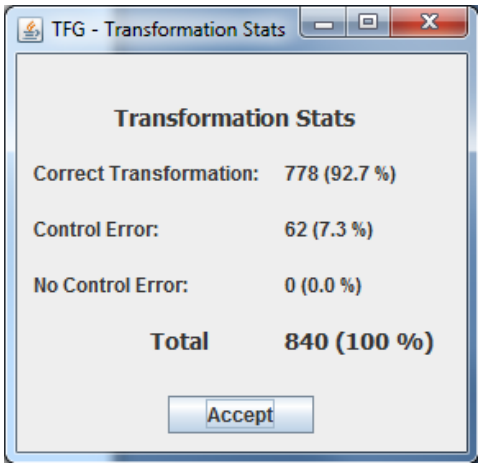


Ilustración 42 - Estadísticas de la transformación

Por último hay que recordar, que si en cualquier momento hubiésemos pulsado el botón "Cancel" la herramienta se hubiera cerrado.