DESARROLLO DE FAMILIAS DE PRODUCTOS BASADO EN MDA PARA SISTEMAS TELEMÁTICOS

Yuli Garcés Bolaños
Alejandra Reyes Reina

Anexos

Director
Dr. Rodrigo Cerón

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Línea de investigación en Ingeniería de Sistemas Telemáticos
Departamento de Telemática
Popayán, Octubre de 2009

ANEXOS

ÍNDICE DE CONTENIDO

Página

ÍNDICE DE FIGURAS

Página

ANEXO A
Análisis y Diseño del caso de estudio

ANEXO B
Modelado de arquitectura de Líneas de Productos Software

ANEXO C
Manual de usuario

ÍNDICE DE TABLAS

Página

ANEXO A

ANEXO A

Análisis y Diseño del caso de estudio

1. Lista de acrónimos

ARES	Architectural Reasoning for Embedded Systems
ATL	Atlas Transformation Languaje
BAPO	Business Architecture Process Organisation
CAFÉ	From Concepts to Application in System-Family Engineering
CASE	Computer Aided Software Engineering
CIM	Computational Independent Model
CORBA	Common Object Request Broker Architecture
CWM	Common Warehouse Metamodel
DSL	Domain Specific Languaje
ECMDA-FA	European Conference on Model Driven Architecture - Fondations and Applications
EMF	Eclipse Modeling Framework
ER	Entity-Relationship
ESAPS	Engineering Software Architectures, Processes and Platforms for System-Families
ESI	European Software Institute
FAMILIES	FAct-based Maturity through Institutionalisation Lessons-learned and Involved Exploration of System-family engineering
FFRDC	Federally Funded Research and Development Center
IST	Infromation Society Technologies
IT	Information Technology
ITEA	Information Technology for European Advancement
JDK	Java Development Kit
MDA	Model Driven Development
MDD	Model Driven Development
MDDi	Model Driven Develpment integration
MOF	Meta-Object Facility
OCL	Object Constraint Language
OMG	Object Management Group
OSGI	Open Source Gateway Initiative
PDE	Plug-in Development Environment
PIM	Plataform Independent Model
PRAISE	Product-line Realization and Assessment in Industrial Settings
PSI	Platform Specific Implementation
PSM	Plataform Specific Model
QVT	Query, Views, and Transformations
RTSS	Research, Technology, and System Solutions Program
SDE	Smart Development Environment
SEI	Software Engineering Institute
SPEM	Software Process Engineering Metamodel
SPL	Software Product Line
UML	Unified Modeling Languaje
XMI	XML Metadata Interchange
XML	Extensible Markup Language

2. Documentación detallada de los casos de uso

Nota: los casos de uso detallados corresponden a los casos de uso sencillos descritos en la vista de la funcionalidad.

NOMBRE	NÚMERO	PRIORIDAD
Leer diagrama	1	Alta
PROPÓSITO		**ACTOR O ROL**
Leer la información contenida en el diagrama UML (PIM)		Arquitecto de productos
DESCRIPCIÓN		
Pre-condición:		
El usuario debe haber solicitado abrir un diagrama UML, dicho diagrama debe estar almacenado en el lugar que la herramienta dispone para tal fin y debe haber sido construido tomando en consideración todos los conceptos relacionados con SPL, siguiendo el proceso descrito en el anexo B sobre cómo definir una arquitectura para SPL, Nota: la herramienta SDE fue utilizada para el desarrollo del caso de estudio en el presente trabajo.		
Post-condición:		
El diagrama UML debe haber sido importado al plugin en un archivo XMI que contiene la información sobre el diagrama, la información que contiene leída y el diagrama desplegado		
Flujo Principal:		
1. Se importa el diagrama desde la herramienta de modelado en la cual fue creado, en un archivo XMI. **2.** El plugin lee la información que contiene el archivo XMI		
Flujos secundarios:		
Ninguno		
Flujos de excepción:		
Error al importar archivo XMI. Página de error: una página es visualizada, en ella está indicado que no existe el archivo solicitado o el tipo de error presentado a la petición.		

Tabla 1. Descripción detallada del caso de uso 1 “leer diagrama”

NOMBRE	NÚMERO	PRIORIDAD
Capturar reglas	2	Alta
PROPÓSITO		**ACTOR O ROL**
Capturar las reglas de variabilidad aplicables a la SPL bajo desarrollo.		Arquitecto de línea
DESCRIPCIÓN		
Pre-condición:		
El usuario debe seleccionar las reglas de variabilidad que desea aplicar a la SPL bajo desarrollo. Este caso de uso posee una relación de inclusión (<<include>>) con el caso de uso leer diagrama. Es recomendable que el usuario revise el diagrama, antes de seleccionar las reglas.		
Post-condición:		
Las reglas de variabilidad aplicables a la SPL son almacenadas.		
Flujo Principal:		
1. El usuario introduce las reglas que desea aplicar para la construcción de la SPL. **2.** El plugin captura las reglas. **3.** Las reglas son almacenadas.		
Flujos secundarios:		

Ninguno
Flujos de excepción:
Error al capturar regla Página de error: una página es visualizada explicando el tipo de error presentado.

Tabla 2. Descripción detallada del caso de uso 2 “capturar reglas”

NOMBRE	NÚMERO	PRIORIDAD
Procesar diagramas	3	Alta
PROPÓSITO		**ACTOR O ROL**
Procesar la información leída del diagrama seleccionado y de las reglas capturadas		Arquitecto de productos
DESCRIPCIÓN		
Pre-condición:		
Este caso de uso posee una relación de inclusión con el caso de uso leer diagrama, por tanto, antes de procesar el diagrama, éste debe ser leído y las reglas de variabilidad del usuario capturadas, con el fin de lograr un procesamiento adecuado.		
Post-condición:		
El diagrama con información específica (PIM más detallado que el del caso de uso 1) relativa a la variabilidad de la SPL bajo desarrollo		
Flujo Principal:		
1. El usuario aplica un conjunto de reglas lógicas y de selección a los diagramas adecuados. **2.** Se incorpora información acerca de la variabilidad de la SPL, de acuerdo a las reglas capturadas. **3.** Se agrega especificidad por primera vez al diagrama de la SPL bajo desarrollo.		
Flujos secundarios:		
Ninguno		
Flujos de excepción:		
Ninguno		

Tabla 3. Descripción detallada del caso de uso 3 “procesar diagramas”

NOMBRE	NÚMERO	PRIORIDAD
Seleccionar decisiones	4	Alta
PROPÓSITO		**ACTOR O ROL**
Incorporar información sobre el núcleo común y variabilidad de la SPL.		Arquitecto de productos
DESCRIPCIÓN		
Pre-condición:		
Deben existir opciones que permitan incorporar información sobre el núcleo común y variabilidad de la SPL, bien sea por selección o por procesamiento lógico.		
Post-condición:		
La información resultante del caso de uso anterior (3), contiene datos adicionales relacionados con el núcleo común y variabilidad de la SPL.		
Flujo Principal:		

1. El usuario toma decisiones sobre la SPL bajo desarrollo. **2.** El usuario selecciona algunas de las opciones posibles para incorporar información a la SPL. **3.** Se incorpora información a la SPL sobre el núcleo común y la variabilidad de la misma (por ejemplo: métodos y clases).
Flujos secundarios:
Ninguno
Flujos de excepción:
Ninguno

Tabla 4. Descripción detallada del caso de uso 4 "seleccionar decisiones"

NOMBRE	NÚMERO	PRIORIDAD
Solucionar conflictos	5	Alta
PROPÓSITO		**ACTOR O ROL**
Obtener el conjunto de soluciones válidas para la SPL.		Arquitecto de productos
DESCRIPCIÓN		
Pre-condición:		
Conflictos existentes entre requerimientos, decisiones o reglas y separación difusa entre componentes obtenidos en el caso de uso anterior (4).		
Post-condición:		
Conjunto reducido de soluciones posibles, de acuerdo al diseño de la SPL.		
Flujo Principal:		
1. Se aplican filtros de restricciones software al conjunto obtenido en el caso de uso anterior (4). **2.** Se eliminan conflictos entre posibles soluciones. **3.** Se obtiene un conjunto reducido o restringido de soluciones posibles para la SPL.		
Flujos secundarios:		
Ninguno		
Flujos de excepción:		
Error al obtener el conjunto de soluciones. Página de error: una página es visualizada indicando la presencia de características variantes del modelo que se encuentran en conflicto, o la ausencia de características necesarias para la construcción de algunos productos de la SPL.		

Tabla 5. Descripción detallada del caso de uso 5 "solucionar conflictos"

NOMBRE	NÚMERO	PRIORIDAD
Derivar	6	Alta
PROPÓSITO		**ACTOR O ROL**
Obtener un producto o instanciación de producto (puede ser más de uno) de todos los posibles que permite el diseño de la SPL.		Arquitecto de productos
DESCRIPCIÓN		
Pre-condición:		
Este caso de uso tiene una relación de inclusión con los casos de uso seleccionar decisión (4), solucionar conflictos (5) y crear diagrama (7), por tanto, "derivar" incluye las funcionalidades de estos		

casos de uso. Debe existir un conjunto de soluciones posibles para la SPL.
Post-condición:
Un conjunto final de soluciones válidas para la SPL.
Flujo Principal:
1. El usuario instancia unos o más productos. **2.** Se eliminan las opciones que no son aplicables. **3.** Se obtiene el conjunto de soluciones válidas o conjunto final de soluciones, que puede estar vacío, contener tan solo un elemento (producto) o varios elementos para la SPL.
Flujos secundarios:
Ninguno
Flujos de excepción:
Error al instanciar productos Página de error: una página es visualizada indicando que el conjunto de soluciones válidas se encuentra vacío y las razones por las cuales sucedió esto. Nota: si el conjunto está vacío porque las indicaciones introducidas por el usuario arrojaron ese resultado, no se considera un error.

Tabla 6. Descripción detallada del caso de uso 6 “derivar”

NOMBRE	NÚMERO	PRIORIDAD
Crear diagrama	7	Alta
PROPÓSITO		**ACTOR O ROL**
Crear un diagrama que contenga información específica a la SPL que se desea implementar (PIM detallado)		Arquitecto de productos
DESCRIPCIÓN		
Pre-condición:		
El conjunto final de soluciones válidas para la SPL, la posibilidad de elegir una única solución entre todas las que contiene dicho conjunto y la opción de guardar la solución elegida		
Post-condición:		
El diagrama que contiene la información de la solución seleccionada debe ser desplegado y el archivo .XMI que contiene información relacionada a éste, debe haber sido almacenado en una base de datos		
Flujo Principal:		
1. Se guarda la información resultante del caso de uso anterior (6) en un archivo .XMI **2.** Se crea un diagrama UML que refleja la información guardada en el archivo .XMI		
Flujos secundarios:		
Ninguno		
Flujos de excepción:		
Error al crear diagrama Página de error: una página es visualizada indicando que no es posible crear el diagrama porque el conjunto de soluciones válidas está vacío.		

Tabla 7. Descripción detallada del caso de uso 7 “crear diagrama”

3. Procesos que conforman el diagrama general de la aplicación

3.1. Lectura del modelo

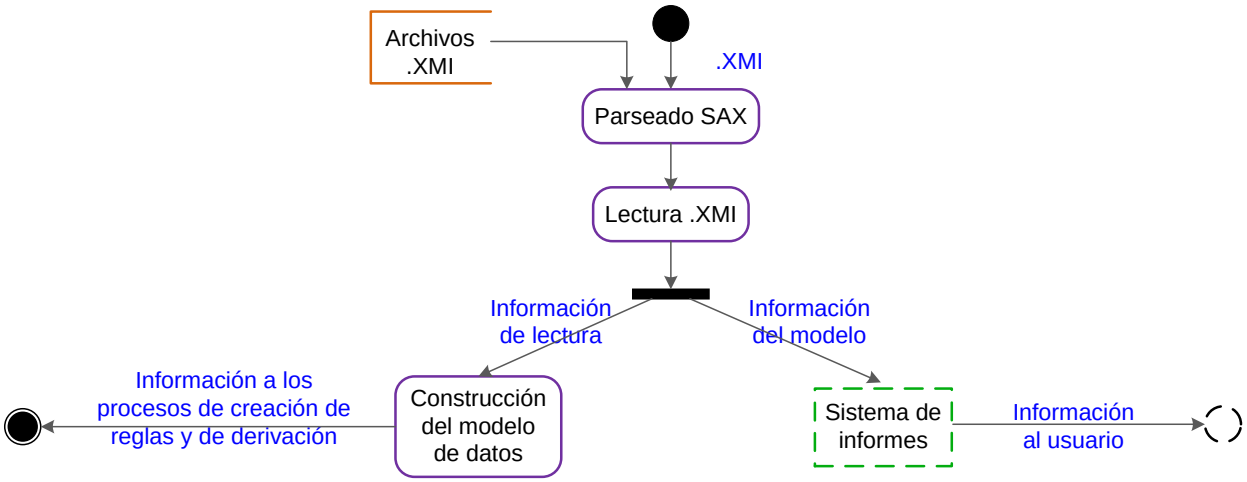

Figura 1. Diagrama del proceso de lectura de modelos

3.2. Construcción de las reglas de derivación

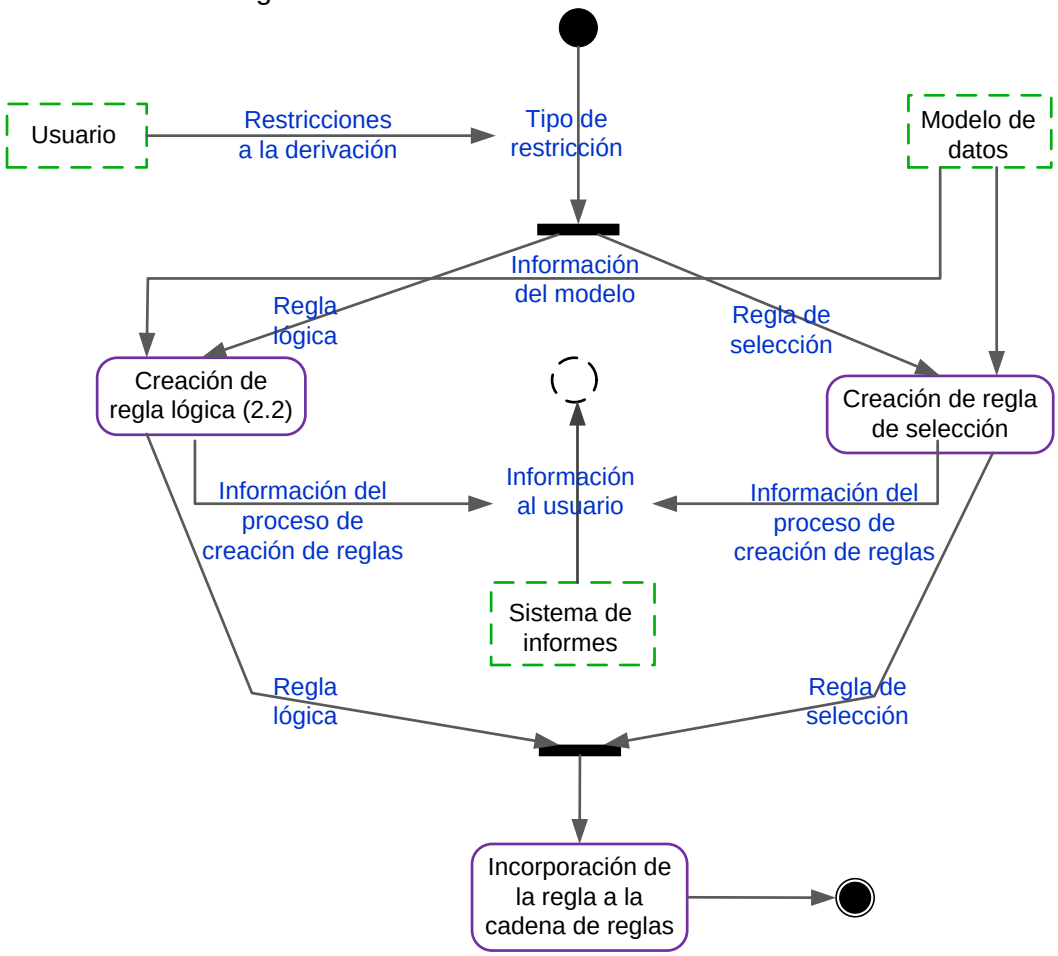

Figura 2. Diagrama del proceso de construcción de reglas

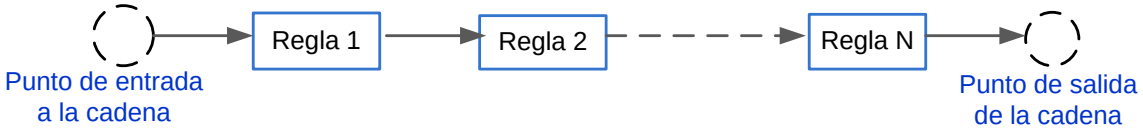

Figura 3. Diagrama de la cadena de reglas

Figura 4. Diagrama del proceso de construcción de reglas lógicas

La construcción de reglas lógicas pasa por 3 subprocesos principales:

- La canonización de la expresión: subproceso en el cual se sustituyen las referencias a elementos del diagrama por identificadores neutros, guardando la correspondencia en una tabla diccionario, se analizan los paréntesis y conectores que aparecen en la expresión y se construye una expresión canónica equivalente.
- El cálculo de la tabla de verdad: subproceso en el que se obtiene la tabla de verdad representativa del comportamiento de la expresión canónica calculada.
- La construcción de la regla lógica: subproceso en el que se relaciona la tabla de verdad obtenida con los elementos del modelo, a través del diccionario creado para ello en el primer subproceso.

La incorporación de reglas a la cadena es un subproceso en donde una regla es creada y repetida por cada una de las restricciones que el usuario desee incorporar, añadiendo en cada repetición, la regla creada a una cadena, en las que estas se suceden según su orden de creación.

3.3. Derivación del modelo

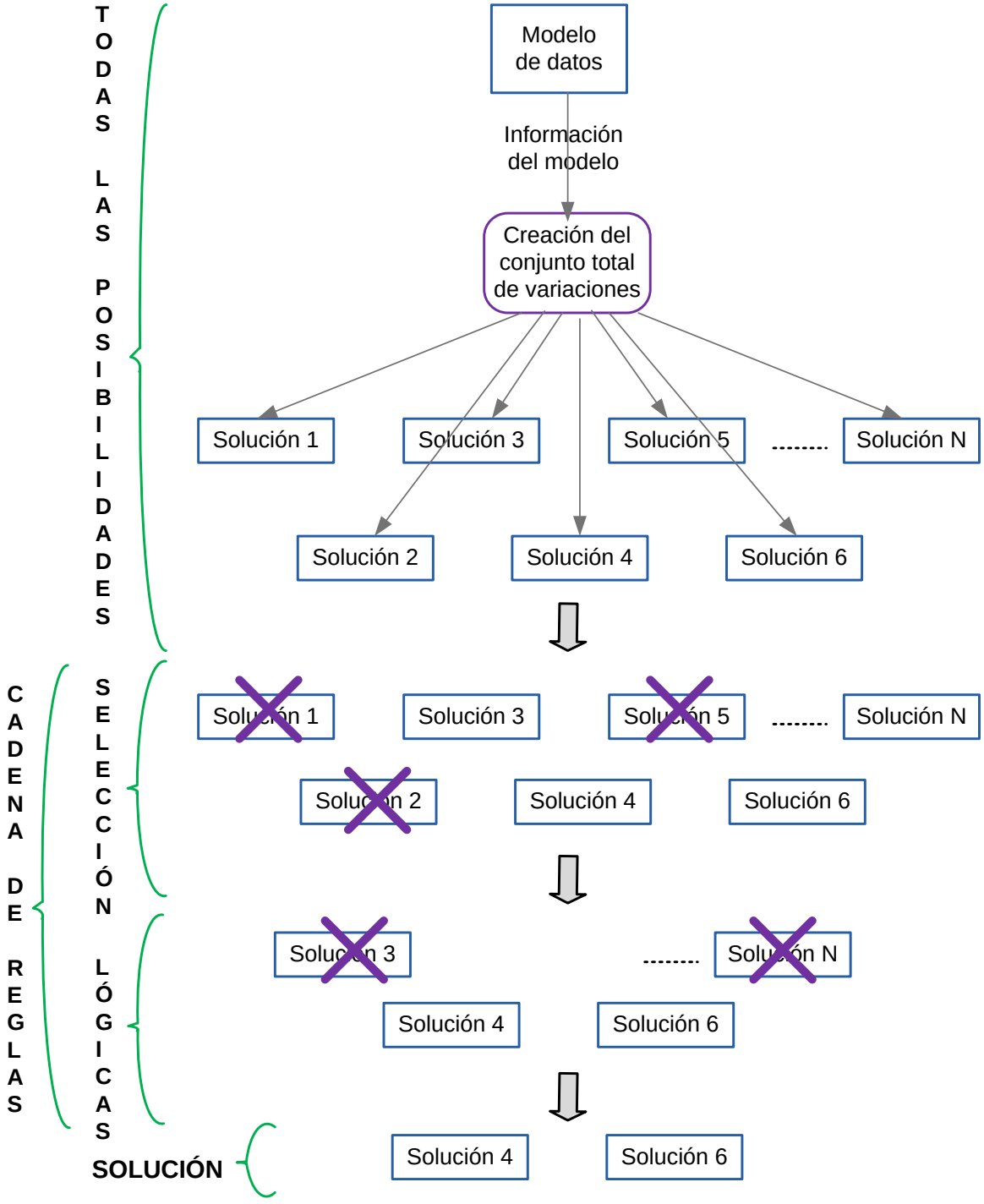

Figura 5. Ejemplo explicativo del proceso de derivación de un modelo

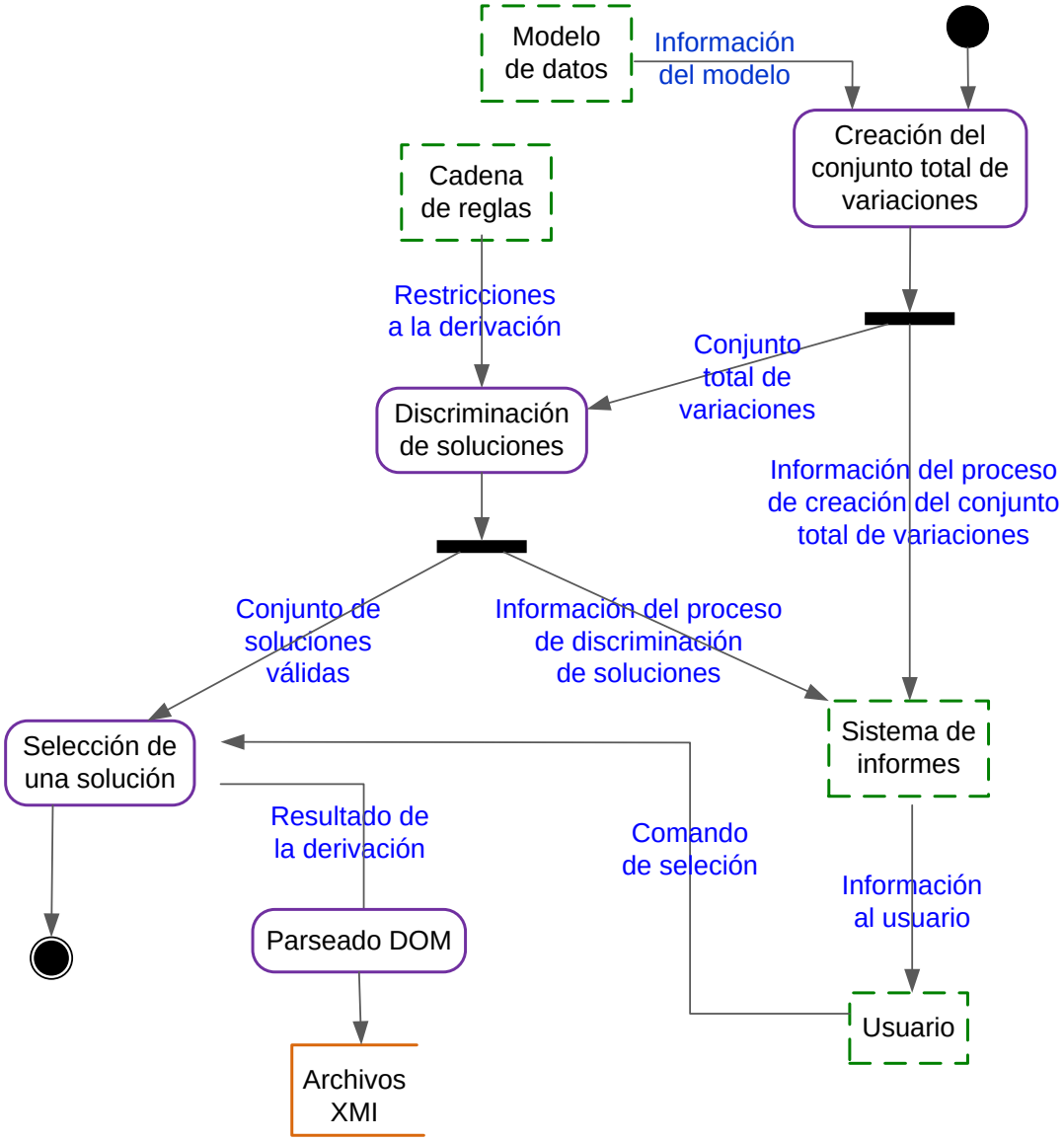

Figura 6. Diagrama del proceso de derivación del modelo

4. Descripción de las clases del modelo de entrada
Nota: el modelo de entrada es el diagrama de clases desarrollado para el caso de estudio

Clase	Objetivo	Variable
EstadísticasEjecución	Recoge y procesa las estadísticas (sistema de informes)	True
UtilTiempoReal	Utilidades de tiempo real, como relojes y contadores de tiempo, así como un sistema de activación de eventos.	False
TareaModelos	Ejecuta los modelos matemáticos que contiene la verdadera lógica del videojuego.	False
ControlCentral	Coordina la lógica del videojuego	False

Tabla 8. Clases del dominio del modelo

Clase (C)/ Interfaz (I)	Objetivo	Variable
SistemaControlReal (C)	Estas clases se encargan de implementar la interfaz de comunicación con cada uno de los componentes del simulador a través de un sistema de comunicación real, es decir, los componentes deben existir en su totalidad para que estas comunicaciones operen.	True
SistemaMovimientoReal (C)		True
SistemaSonidoReal (C)		True
SistemaVisualReal (C)		False
SistemaSonidoSimulado (C)	Implementa la interfaz de comunicación con el componente de sonido del videojuego, a través de un sistema de comunicación simulado, es decir, simula la existencia de este componente para posibilitar pruebas tempranas e integración gradual de los diferentes componentes.	True
InterfazControl (I)	Cada una de estas interfaces modela la forma particular de comunicación con cada uno de los componentes del videojuego. La interfaz de control está encargada especialmente del manejo de las comunicaciones enfocadas al control distribuido de los componentes.	False
InterfazMovimiento (I)		True
InterfazSonido (I)		True
InterfazVisual (I)		False
ComunicaciónControl (C)	Cada una de estas clases hereda de InterfazComunicación, especializándose en las comunicaciones de cada uno de los diferentes componentes del videojuego. Están encargadas de manejar la lógica particular de cada componente, así como sus tipos de datos asociados.	False
ComunicaciónMovimiento (C)		True
ComunicaciónSonido (C)		True
ComunicaciónVisual (C)		False
InterfazComunicación (C)	Encargada de la comunicación de una forma eficiente y genérica.	False
Tarea (I)	Modela la forma general de una interfaz o tarea de comunicación	False

Tabla 9. Clases del dominio del paquete comunicaciones

Clase (C)/ Interfaz (I)	Objetivo	Variable
Municiones (C)	Modelan la operativa y el comportamiento de diferentes tipos de municiones.	True
Blancos (C)	Modela la operativa y el comportamiento de diferentes tipos de objetivos	True
Armas (C)	Modela la operativa y el comportamiento de diferentes tipos de armas	True
Colisiones (C)	Modela los eventos de colisión y sus efectos sobre elementos del juego y el entorno.	False
Movimiento (C)	Modela los principios físicos del movimiento (velocidad, aceleración, fricción, resistencia, etc.)	False
Aerodinamico (C)	Modela las relaciones físicas impuestas por las leyes de la aerodinámica (Bernoulli, Venturi, etc.)	True
Metereologia (C)	Modela los efectos de los fenómenos meteorológicos (lluvia, niebla, granizo, nieve, etc.)	False
EntornoTerrestre (C)	Estas clases modelan las características físicas y	True

EntornoAereo (C)	ambientales apropiadas para cada entorno.	True
ModeloMatematico (I)	Define la lógica de comportamiento de los modelos matemáticos, así como su comunicación con el núcleo. Lo implementan todas las clases del paquete.	False

Tabla 10. Clases del dominio del paquete ModelosMatematicos

Clase	Objetivo	Variable
Camión	Representa la información y el comportamiento particulares de un camión. Hereda de la clase JugadorTierra.	True
Tanque	Representa la información y el comportamiento particulares de un tanque. Hereda de la clase JugadorTierra.	True
Carro	Representa la información y el comportamiento particulares de un carro. Hereda de la clase JugadorTierra.	True
Bombardero	Representa la información y el comportamiento particulares de un bombardero. Hereda de la clase Avión.	True
AvionCombate	Representa la información y el comportamiento particulares de un avión de combate. Hereda de la clase Avión.	True
Avion	Representa la información y el comportamiento general de un avión. Hereda de la clase JugardorAire	False
JugadorTierra	Representa la información y el comportamiento comunes a todos los elementos de tipo terrestre. Hereda de la clase Jugador.	True
JugadorAire	Representa la información y el comportamiento comunes a todos los elementos de tipo aéreo. Hereda de la clase Jugador.	True
Jugador	Representa la información y el comportamiento comunes a todos los elementos inteligentes de la simulación.	False

Tabla 11. Clases del dominio del paquete Elementos

Clase (C)/ Interfaz (I)	Objetivo	Variable
GuiAuxiliar	Proporciona las funciones auxiliares para la GUI	True
Gui	Proporciona la interfaz de usuario gráfica para las funciones de control.	True
InterfazLogica	Establece la comunicación de control con el núcleo del videojuego.	False
UnidadControl (C)	Maneja la lógica de señales que domina el comportamiento del juego.	False
ReconocimientoVoz	Proporciona la interfaz de usuario por reconocimiento de voz para las funciones de control.	True

Tabla 12. Clases del dominio del paquete Interacciones

5. Atributos y métodos de las clases de la aplicación
Nota: las clases corresponden al diagrama de clases de la vista lógica de la arquitectura

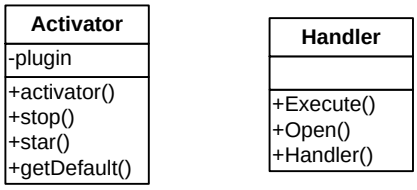

Figura 7. Diagrama de clases del paquete plugin, atributos y métodos

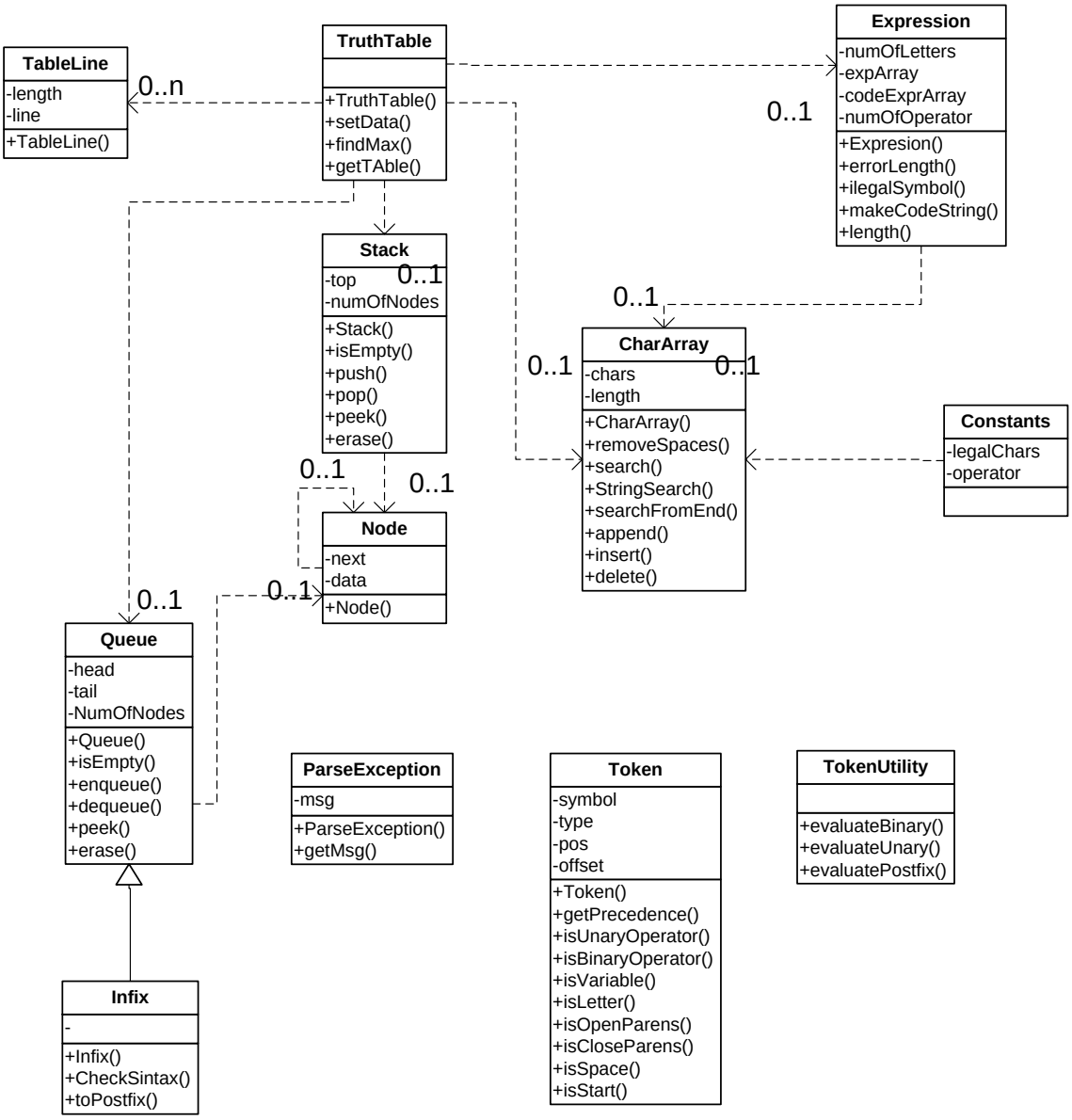

Figura 8. Diagrama de clases del paquete lógica, atributos y métodos

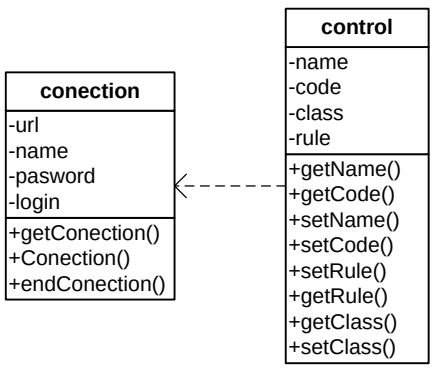

Figura 9. Diagrama de clases relacionado con la persistencia de la aplicación, atributos y métodos

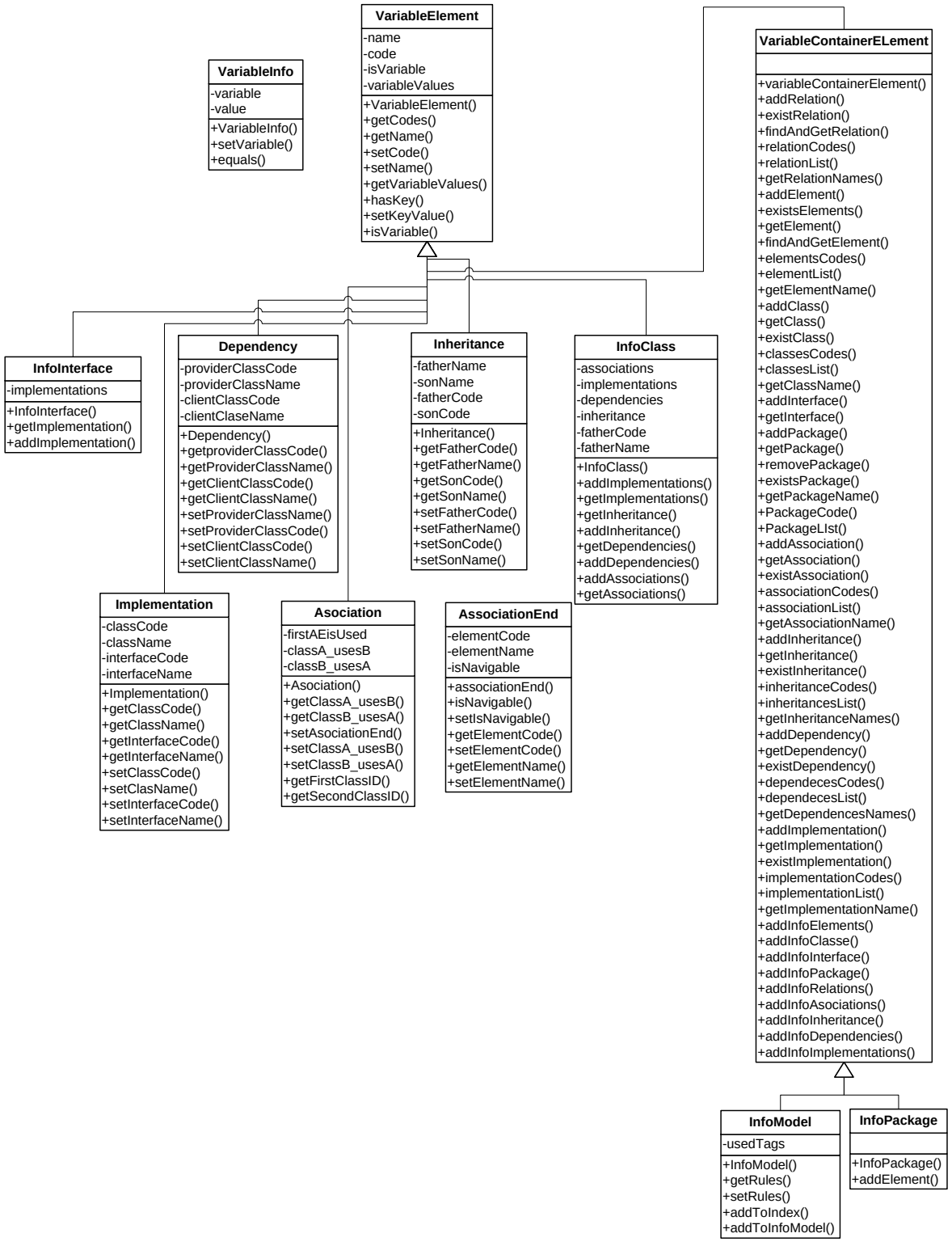

Figura 10. Diagrama de clases del paquete estructura de datos, atributos y métodos

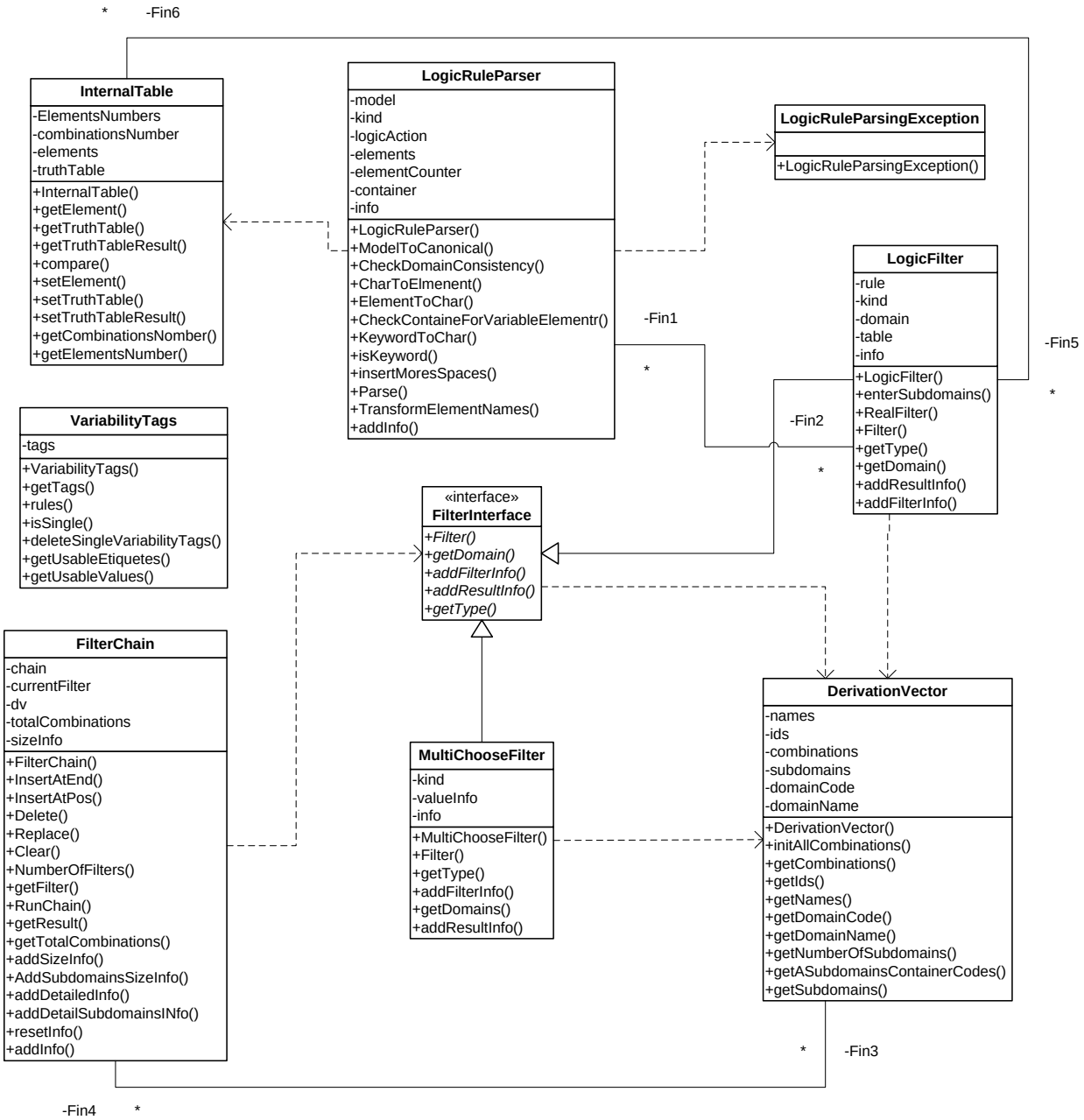

Figura 11. Diagrama de clases del paquete filtro, atributos y métodos

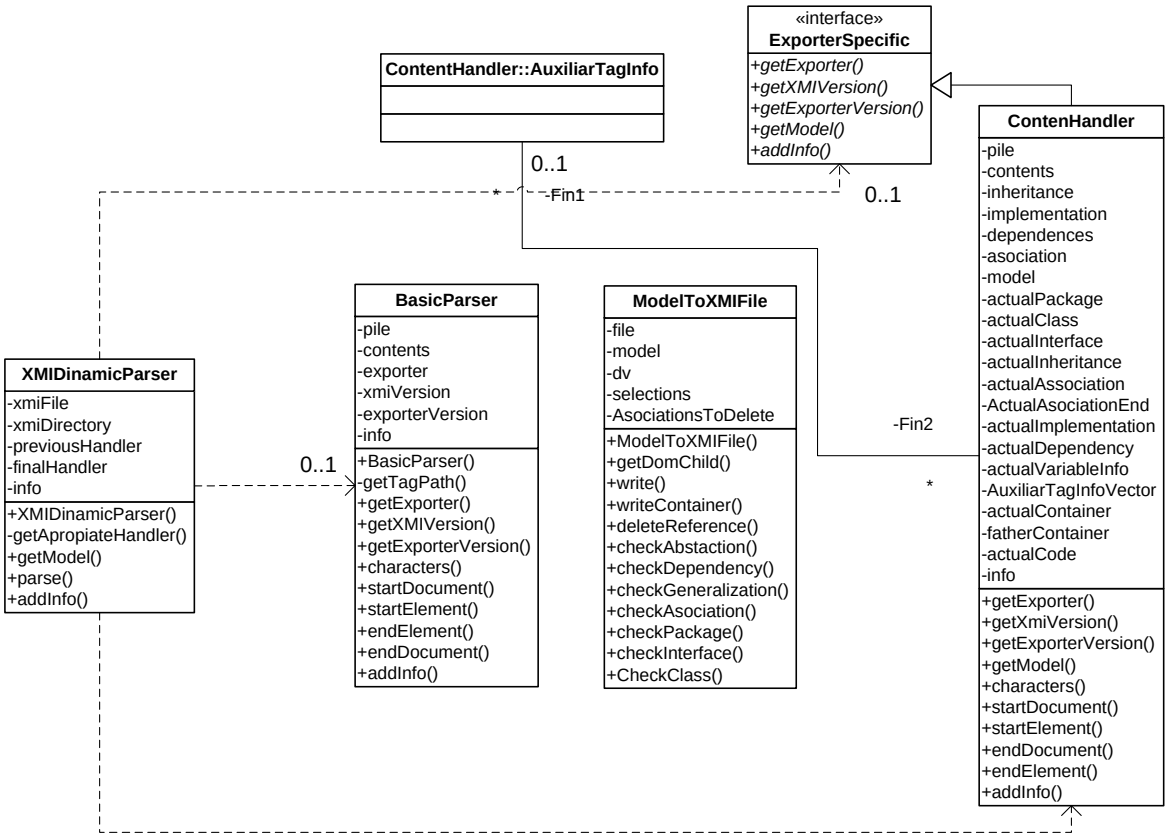

Figura 12. Diagrama de clases del paquete XMI, atributos y métodos

ANEXO B

Modelado de arquitecturas de Líneas de Productos Software

1. Desarrollo del núcleo de la arquitectura de referencia

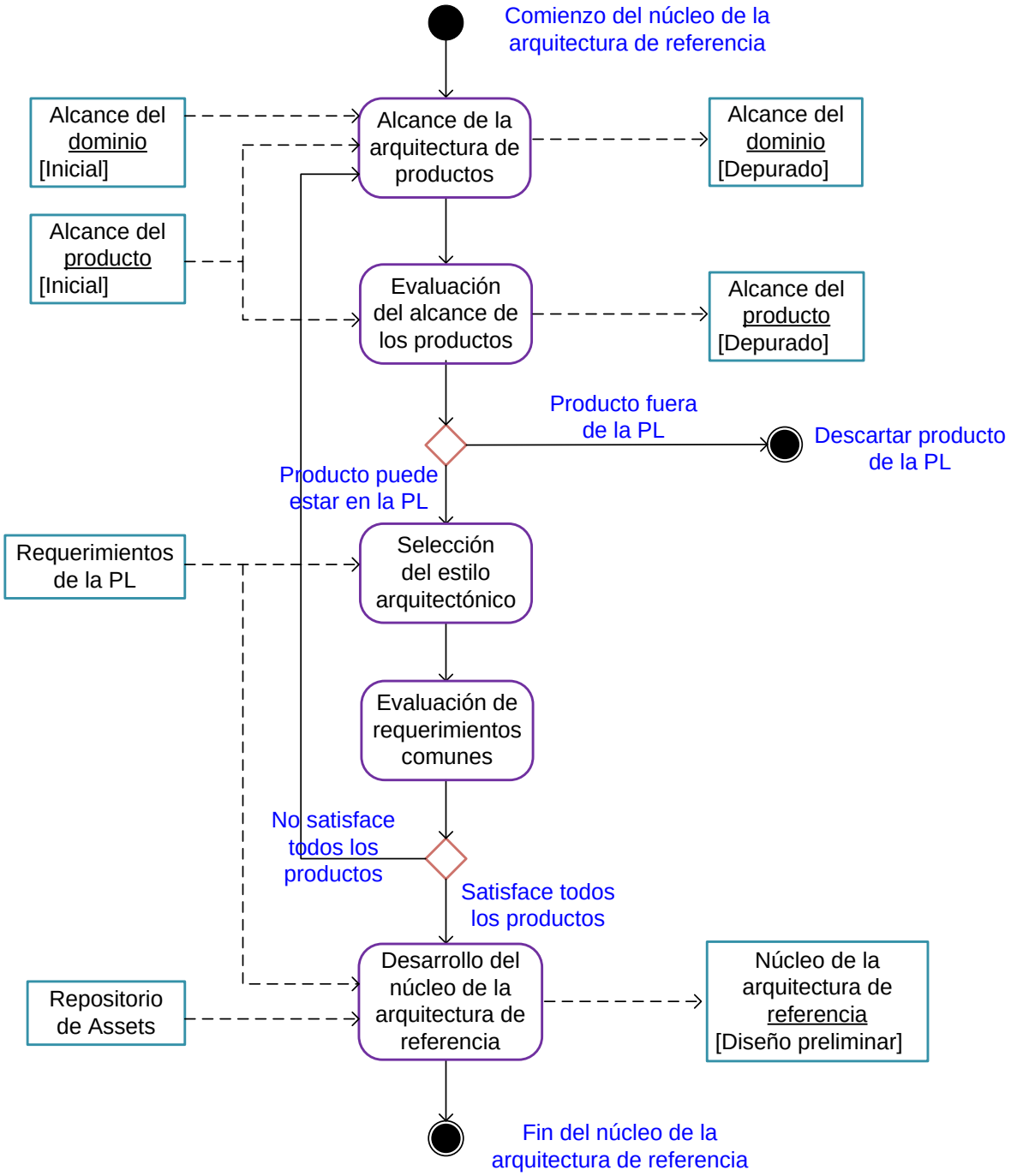

Figura 13. Diagrama del proceso de desarrollo del núcleo de la arquitectura de referencia

2. Refinamiento y finalización de la arquitectura

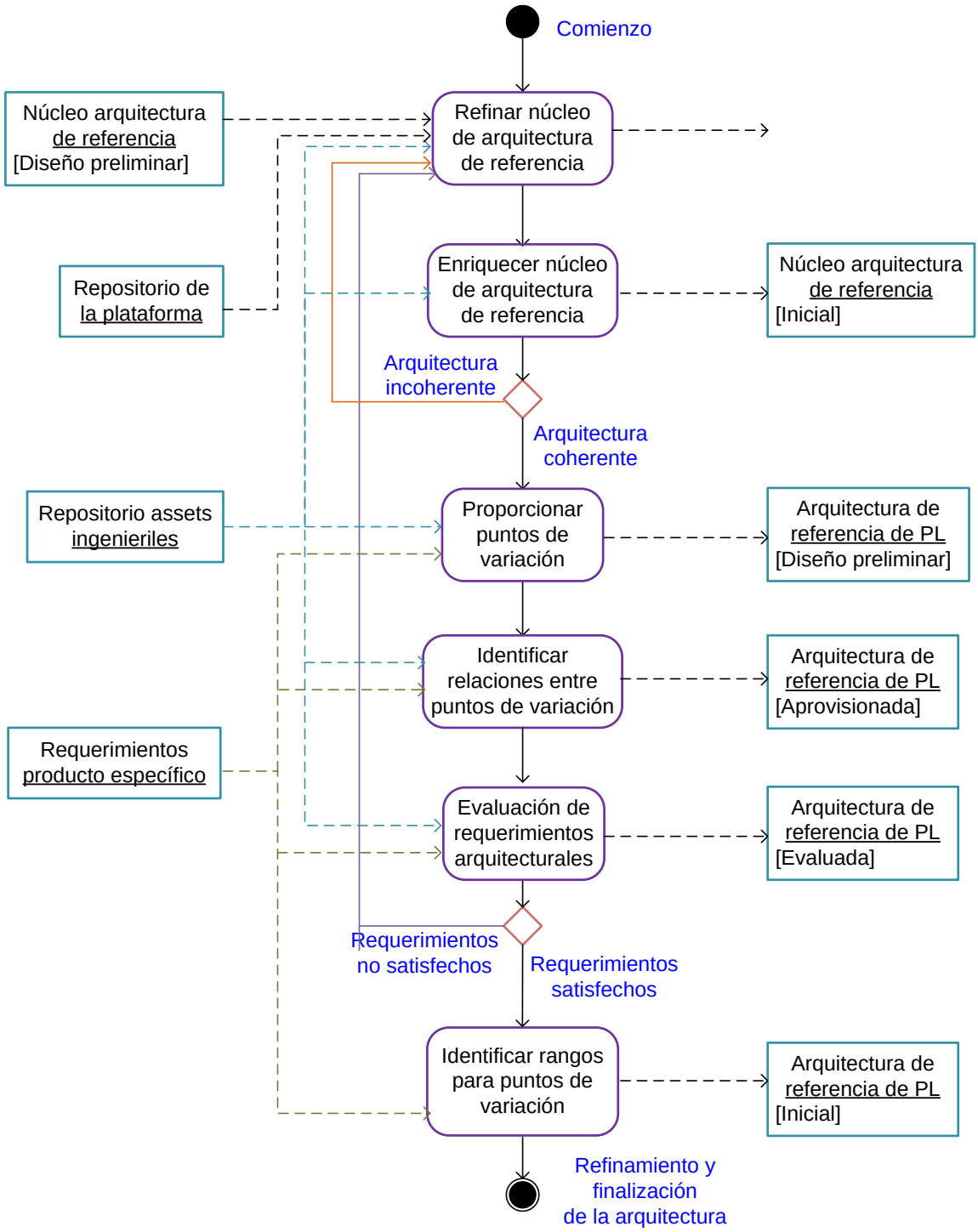

Figura 14. Diagrama del proceso de refinamiento y finalización de la arquitectura

ANEXO C
Manual de usuario

Mda.spl.architecture.project es una herramienta que sigue el enfoque MDA y permite al usuario hacer transformaciones horizontales de modelos PIM (PIM de alto nivel a PIM detallado), bajo el lineamiento de SPL. El plugin obtiene diagramas UML detallados a partir de diagramas de alto nivel teniendo en cuenta un conjunto de restricciones y selecciones definidas.

1. Requisitos de Usuario

Requisitos Hardware y software principales

- Computador Personal arquitectura requerida por el usuario, Sistema Operativo Windows 2000, XP, Vista 7.
- Maquina Virtual de Java 1.6, JRE 1.6.
- Eclipse Ganyemde 3.4
- Plugin eUML para Eclipse
- MySQL 5.0
- MySQL Front

2. Instalación de la aplicación

Para acceder a la aplicación el usuario debe tener instalada la máquina virtual de Java en su PC, para verificar la versión instalada ingrese en la consola MS DOS de su computador y digite el comando java –v que le informará la versión de Java, si no encuentra ninguna puede descargar la versión necesaria de la página web de Sun Microsystems *www.java.sun.com*.
Para realizar la instalación simplemente ejecute el install.exe dando doble clic sobre el ícono o clic derecho, abrir, aquí se ejecutará el plugin de eclipse.
Después de instalados los componentes y el MySQL importe la base de datos y está listo para trabajar.

3. Aspecto de la aplicación

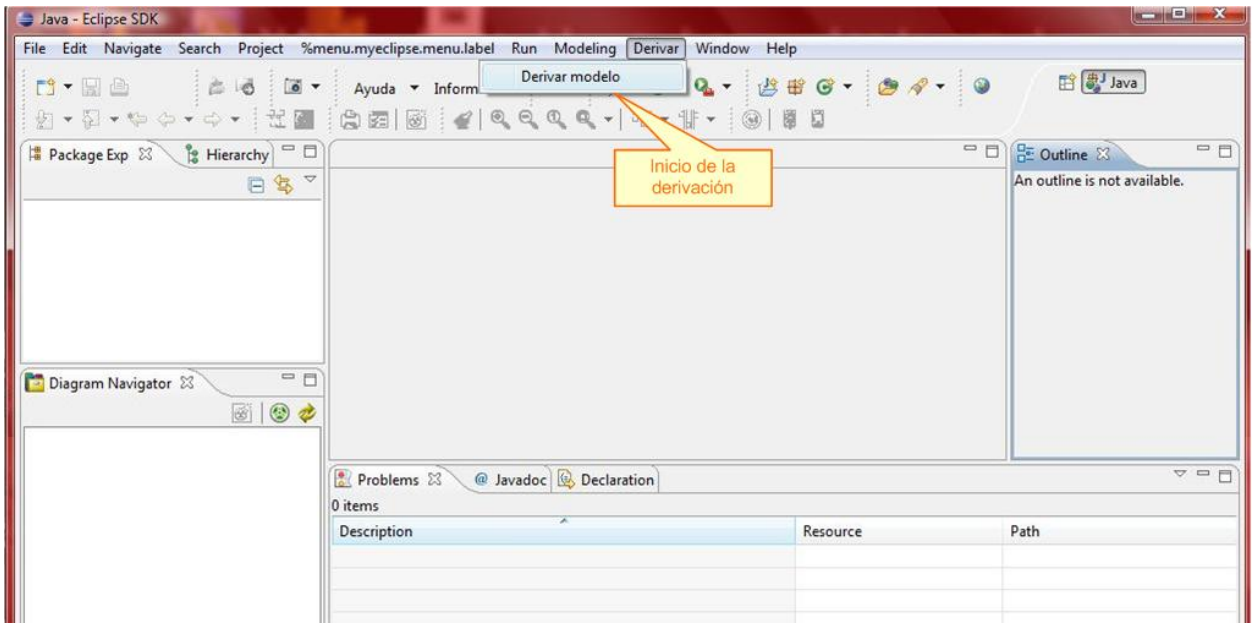

Figura 15. Aspecto general del plugin

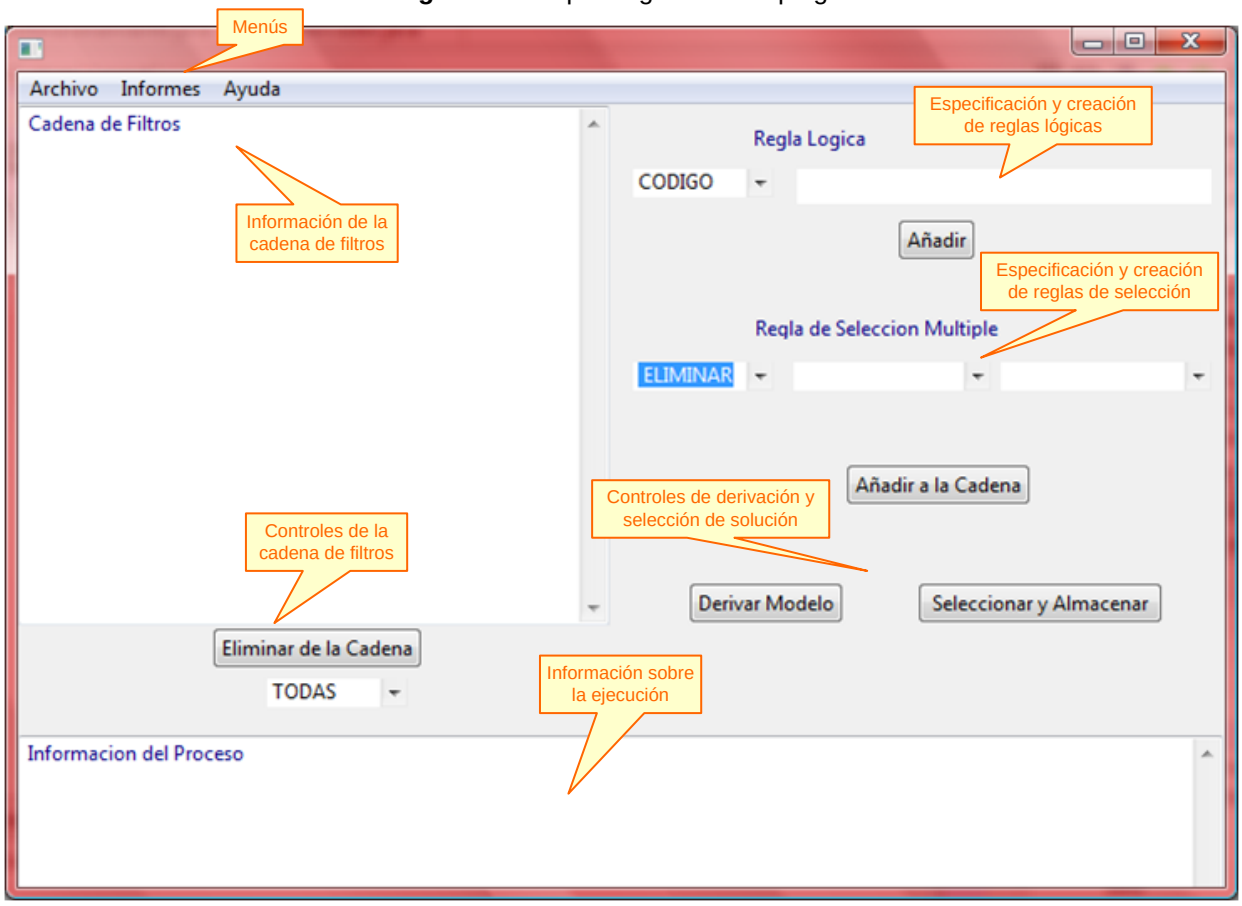

Figura 16. Aspecto general del derivador

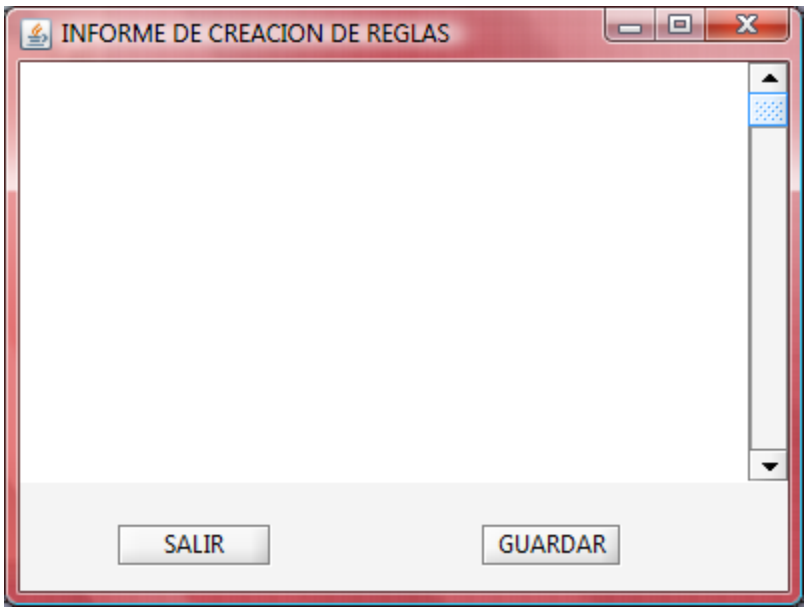

Figura 17. Aspecto general del Sistema de informes

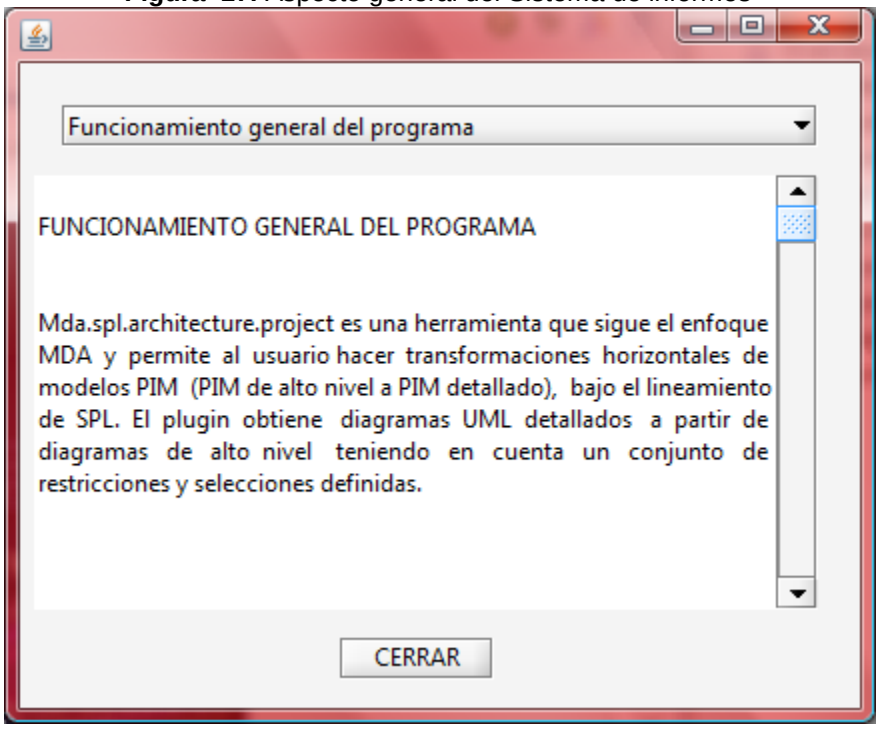

Figura 18. Aspecto general del sistema de ayuda

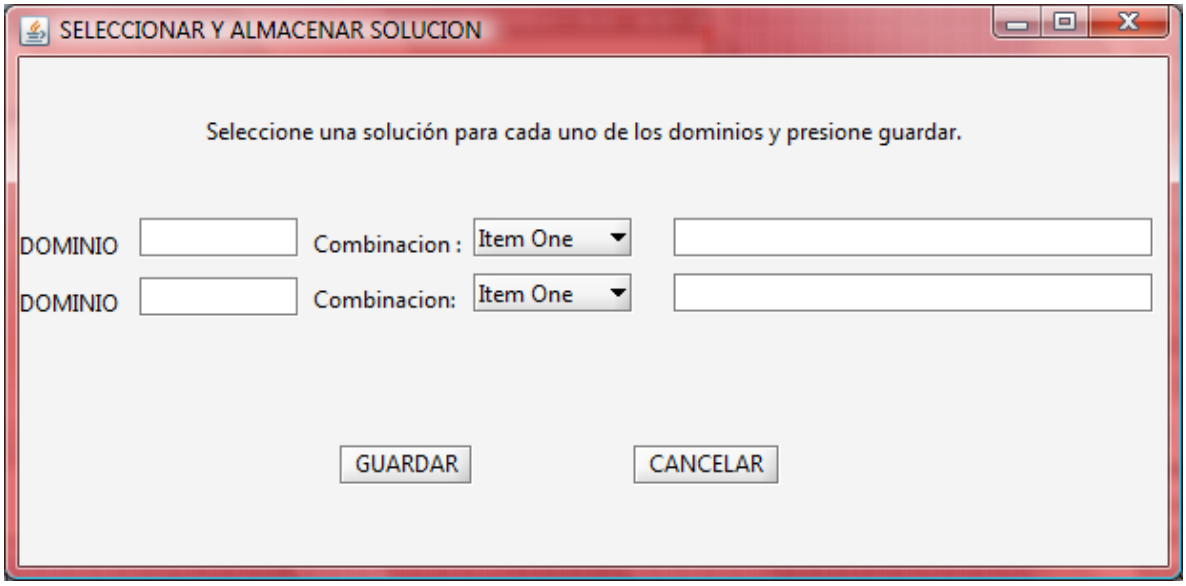

Figura 19. Aspecto de la pantalla de selección de soluciones

4. Tutorial

4.1. Cargar la aplicación

Ejecutado el plugin diríjase a la parte superior derecha en la barra de herramientas de eclipse y note que existe un menú llamado derivar, selecciónelo y seleccione la opción derivar modelo. A continuación el plugin carga una GUI que le muestra el derivador general.

4.2. Abrir el archivo

Para abrir el modelo seleccione el menú archivo Abrir modelo y elija el archivo XMI que desea derivar. A partir de este momento y durante cualquier punto de la ejecución de la aplicación usted puede acceder al informe de procesado del modelado. Estos informes presentan información detallada del proceso de modelado de la aplicación igual que las restricciones y filtros lógicos introducidos. Puede almacenar estos informes en un archivo si así se requiere.

4.3. Introducir filtros y restricciones

Para introducir los filtros y restricciones se debe tener en cuenta el tipo de regla de cadena, lógica o de selección.
Las reglas lógicas permiten introducir relaciones lógicas entre elementos variables del diagrama. Puede relacionar los elementos mediante su nombre en el modelo UML o con su código asignado XMI.
Las reglas de selección le permiten indicar que valores específicos deben tener ciertas características del diagrama, indicados en las etiquetas del modelo UML. SELECCIONAR indica que el elemento que contenga esta etiqueta debe pertenecer al diagrama y ELIMINAR indica que el elemento que contenga esta etiqueta debe ser eliminado mediante la derivación.

4.4. Derivando y almacenando los nuevos modelos

Una vez introducidas las reglas de restricción deseadas se procede a derivar el modelo. Cuando el proceso termine el panel le informa que la derivación a concluido y le presenta el numero de modelos solución que se han obtenido. Puede seleccionar una solución y almacenarla. A continuación se abre una ventana en la que se muestran los diferentes dominios, se selecciona la combinación se presiona el botón guardar y damos nombre al archivo.

5. Ayuda de la aplicación

El plugin contiene un sistema de ayuda que le permite tener conocimiento del funcionamiento general del programa. Diríjase a la parte del menú de herramientas de la GUI y seleccione el menú ayuda, elija la opción contenidos y seleccione el tipo de contenido que desea ver. Los temas disponibles son:

- Funcionamiento general del programa.
- Proceso de generación y visualización de informes.
- Funcionamiento de los filtros de selección múltiple.
- Funcionamiento de los filtros lógicos.