



ESCUELA SUPERIOR DE INGENIERÍA

Ingeniería Técnica en Informática de
Sistemas

Editor Gráfico para la Definición de
Patrones de Eventos Complejos sobre
Seguridad y Generador de Código EPL

Curso 2011-2012

Ismael Morales Alonso

Cádiz, 17 de septiembre de 2012



ESCUELA SUPERIOR DE INGENIERÍA

Ingeniería Técnica en Informática de Sistemas

Editor Gráfico para la Definición de
Patrones de Eventos Complejos sobre
Seguridad y Generador de Código EPL

DEPARTAMENTO: Ingeniería Informática.

DIRECTOR DEL PROYECTO: Juan Boubeta Puig.

AUTOR DEL PROYECTO: Ismael Morales Alonso.

Cádiz, 17 de septiembre de 2012

Fdo.: Ismael Morales Alonso

Índice general

1. Introducción y motivación	5
1.1. Objetivos	5
1.2. Requisitos	6
1.3. Alcance	6
1.4. Visión general	7
1.5. Glosario	7
1.5.1. Acrónimos	7
1.5.2. Definiciones	9
2. Estado del arte	11
2.1. Procesamiento de Eventos Complejos	11
2.1.1. Características	11
2.1.2. Ventajas	12
2.1.3. Patrones	12
2.2. Esper	13
2.3. Desarrollo de Software Dirigido por Modelos	17
2.3.1. Conceptos	17
2.3.2. Motivación	18
2.3.3. Arquitectura Dirigida por Modelos	19
2.3.4. Principios de Desarrollo Dirigido por Modelos	21
2.4. Lenguaje Específico del Dominio	22
2.5. Eclipse Modeling Framework	23
2.6. Graphical Modeling Framework	24
2.7. Epsilon	24
3. Desarrollo del calendario	29
3.1. Fases	29
3.1.1. 1 ^a fase: Conocimientos previos y elicitation de requisitos	29
3.1.2. 2 ^a fase: Búsqueda de información y asimilación de conceptos	29
3.1.3. 3 ^a fase: Diseño de metamodelo	30
3.1.4. 4 ^a fase: Familiarización de tecnologías	30
3.1.5. 5 ^a fase: Creación de metamodelo en EuGENia y personalización del editor	30

Índice general

3.1.6. 6 ^a fase: Validación de los modelos creados en el editor	30
3.1.7. 7 ^a fase: Transformación de los modelos definidos gráficamente a código Event Processing Language	30
3.1.8. 8 ^a fase: Validación de las transformaciones realizadas sobre los modelos	31
3.1.9. 9 ^a fase: Redacción de la memoria del Proyecto Fin de Carrera	31
3.2. Diagrama de Gantt	31
4. Descripción general del proyecto	35
4.1. Perspectiva del producto	35
4.1.1. Entorno del producto	35
4.1.2. Interfaz de usuario	35
4.2. Funciones	35
4.3. Características del usuario	40
4.4. Restricciones generales	40
4.4.1. Control de versiones	40
4.4.2. Lenguajes de programación y tecnologías	40
4.4.3. Herramientas	42
4.4.4. Sistemas operativos y hardware	42
5. Desarrollo del proyecto	43
5.1. Modelo de ciclo de vida	43
5.2. Requisitos	44
5.2.1. Funcionales	44
5.2.2. De información	45
5.2.3. Reglas de negocio	45
5.2.4. Interfaz	45
5.2.5. No funcionales	45
5.3. Análisis del sistema	46
5.3.1. Casos de uso	46
5.3.2. Metamodelo	48
5.4. Diseño del sistema	58
5.4.1. SecurityEPLdraw	58
5.4.2. Estructura	59
5.5. Implementación	59
5.5.1. Metamodelo	59
5.5.2. Validación de modelos	59
5.5.3. Transformación Model-to-Text	65
5.5.4. Personalización del editor	71
5.6. Pruebas	77

6. Conclusiones y trabajo futuro	79
6.1. Conclusiones	79
6.1.1. Valoración general	79
6.1.2. Valoración personal	80
6.2. Trabajo futuro	80
7. Agradecimientos	81
8. Manual de instalación	83
8.1. Instalación de Eclipse	83
8.2. Instalación del resto de software	83
8.3. Importación de código	83
9. Manual de usuario	85
9.1. Ejecución del editor	85
9.2. Diseño de patrones de eventos complejos	87
9.2.1. Primeros elementos	87
9.2.2. Ejemplos de patrones diseñados	87
9.2.3. Generación de código EPL	95
10. Manual del desarrollador	97
A. Subclipse	99
A.1. Instalación	99
A.2. Configuración	99
B. Código Fuente	101
B.1. Metamodelo	101
B.2. Validación de modelos	119
B.3. Transformación Model-to-Text	158
B.3.1. Fichero EGL	158
B.3.2. Plug-in de integración de la transformación M2T	167
B.4. Personalización del editor	170
B.4.1. Paleta gráfica	171
B.4.2. Entorno principal del editor	174
Bibliografía	177

Índice de figuras

2.1.	Procesamiento de flujos de eventos y correlación	14
2.2.	Arquitectura Esper	14
2.3.	Capas de la arquitectura del metamodelado OMG	18
2.4.	Desarrollo de software tradicional	19
2.5.	Desarrollo de software con MDE	20
2.6.	Proceso de desarrollo MDA	20
2.7.	Estructura GMF	24
3.1.	Diagrama de Gantt octubre-diciembre	32
3.2.	Diagrama de Gantt enero-abril	33
3.3.	Diagrama de Gantt mayo-agosto	34
4.1.	Ejemplo gráfico de definición de un patrón complejo	36
4.2.	Ejemplo gráfico de definición de un patrón sencillo	37
4.3.	Ejemplo de un error en la creación del modelo	38
4.4.	Generación de código EPL a partir del modelo definido con el editor .	39
4.5.	Lenguajes y herramientas de Epsilon utilizados en el PFC	41
4.6.	EMF - GMF - EuGENia	41
5.1.	Modelo en cascada	44
5.2.	Casos de uso con el Actor Usuario	46
5.3.	Casos de uso con el Actor Editor	46
5.4.	Metamodelo	49
5.5.	Root	50
5.6.	Node	51
5.7.	Link	51
5.8.	Clause	52
5.9.	Pattern	52
5.10.	Element	53
5.11.	Event	53
5.12.	Attribute	54
5.13.	Property	55
5.14.	Operator	55
5.15.	Repetition	56

Índice de figuras

5.16. Logical	56
5.17. FollowedBy	57
5.18. Guard	57
5.19. Observer	58
5.20. Relational	58
5.21. Estructura de la aplicación	59
5.22. <i>Plugin</i> y clase asociada	67
5.23. Package Explorer	68
5.24. <i>Plugin</i> y clase asociada	70
5.25. Visión general de la paleta	71
5.26. Clauses en la paleta	72
5.27. Security Events en la paleta	72
5.28. Temporal Operators en la paleta	73
5.29. Repetition Operators en la paleta	73
5.30. Logical Operators en la paleta	74
5.31. Relational Operators en la paleta	74
5.32. Clauses en el entorno principal	75
5.33. Security Events en el entorno principal	75
5.34. Temporal Operators en el entorno principal	76
5.35. Repetition Operators en el entorno principal	76
5.36. Logical Operators en el entorno principal	76
5.37. Relational Operators en el entorno principal	77
9.1. Entorno de ejecución	85
9.2. Entorno vacío	86
9.3. Primeros Elementos	87
9.4. Ataque DOS	88
9.5. Ataque DDOS	89
9.6. Ataque Smurf	90
9.7. Ataque Land	91
9.8. Ataque Supernuke	92
9.9. Ataque escaneo de puertos TCP	93
9.10. Ataque Flood al email	94
9.11. Ataque al FTP	95
B.1. Root	101
B.2. Node	102
B.3. Link	103
B.4. Clause	103
B.5. Select	104
B.6. InsertInto	105

Índice de figuras

B.7. Pattern	105
B.8. Element	106
B.9. Atom - Operator - Observer	107
B.10. Event - SecurityEvent	108
B.11. Event - SecurityEvent	109
B.12. IP - Port - MAC - Timestamp	110
B.13. Unary-Binary-Nary-Repetition-Logical-Temporal-Guard-Relational . .	111
B.14. Until - Every - EveryDistinct - Num - Range	113
B.15. And - Or - Not	114
B.16. FollowedBy	115
B.17. TimerWithin - TimerWithinMax - While	116
B.18. TimerInterval - TimerAt	117
B.19. Operadores Relacionales	118

Índice de figuras

1. Introducción y motivación

En los últimos años ha emergido una tecnología denominada procesamiento de eventos complejos o CEP (Complex Event Processing) [47], que permite procesar y analizar grandes cantidades de eventos, así como correlacionarlos para detectar y responder en tiempo real a situaciones críticas del negocio. Para ello, se utilizan unos patrones de eventos que inferirán nuevos eventos más complejos y con un mayor significado semántico, que ayudarán a tomar decisiones ante las situaciones acaecidas.

Este PFC (Proyecto Fin de Carrera) surge de la necesidad de disponer de una herramienta gráfica que permita definir a cualquier usuario no tecnólogo patrones de eventos complejos aplicados al ámbito de la seguridad informática.

Este PFC se ha realizado con la colaboración del Grupo de Investigación “Grupo UCASE de Ingeniería del Software” (TIC-025) dentro de las áreas de Desarrollo del Software Dirigido por Modelos y Arquitecturas Dirigidas por Eventos. Además de haber trabajado dentro del Grupo de Investigación UCASE, he sido alumno colaborador de Juan Boubeta Puig (tutor del presente PFC) durante el curso académico 2011/2012.

El lenguaje utilizado en el grupo de investigación para el diseño de patrones de eventos complejos es EPL (Event Processing Language) de Esper [17]. Sin embargo, en la actualidad, cualquier persona que tenga la necesidad de definir patrones de eventos complejos en EPL, tendrá que aprender previamente este lenguaje (similar a SQL (Structured Query Language) pero extendido con nuevas funcionalidades que permiten procesar eventos en tiempo real).

Para el desarrollo de dicha herramienta, denominada SecurityEPLdraw, se han utilizado técnicas de desarrollo dirigido por modelos o MDD (Model Driven Development) [57].

La herramienta desarrollada servirá como punto de partida para el diseño de otros patrones de eventos complejos aplicados a diversas áreas, tales como la medicina, la banca, etc.

1.1. Objetivos

El objetivo principal del proyecto es la creación de una herramienta gráfica para el diseño de patrones de eventos complejos sobre seguridad informática. Para lograr

1. Introducción y motivación

este objetivo principal se deben de cumplir los siguientes sub-objetivos:

- Diseñar un metamodelo capaz de representar todos los modelos posibles para la definición de patrones de eventos complejos sobre seguridad informática.
- Personalizar la herramienta gráfica para la obtención de una interfaz amigable e intuitiva.
- Validar los modelos creados por el usuario para garantizar que los patrones diseñados gráficamente son correctos.
- Transformar los modelos definidos gráficamente por los usuarios a código EPL.
- Validar las transformaciones realizadas sobre los modelos.

1.2. Requisitos

Aparte de dichos objetivos, se deben de cumplir los siguientes requisitos:

- Facilidad de uso para que usuarios sin conocimientos de lenguajes de procesamiento de eventos sean capaces de diseñar patrones de manera sencilla.
- La herramienta debe de ser independiente del sistema operativo en el que se utilice.

1.3. Alcance

La herramienta creada pretende ser utilizada por usuarios no tecnólogos con algunos conocimientos sobre la seguridad informática que no conozcan el lenguaje EPL, ya que un usuario avanzado de EPL podría exprimir al máximo la totalidad del lenguaje. SecurityEPLdraw sólo pretende ser un subconjunto de EPL, ofreciendo la creación de los patrones más utilizados de una manera gráfica e intuitiva sin tener ningún conocimiento sobre EPL.

Evidentemente, al ser un DSL (Domain-Specific Language) [49], la herramienta se centra en la definición de patrones sobre ataques a través la red y no puede abarcar otras ramas, pero a partir de esta herramienta, otros desarrolladores podrían utilizarla como punto de partida para el desarrollo de DSL sobre otras ramas de diversa índole.

La herramienta es multiplataforma, por lo que podrá ser ejecutada y desarrollada en diversos sistemas operativos, lo que facilita su difusión.

1.4. Visión general

A continuación trataremos el estado del arte en el que se encuentran CEP, Esper, MDE (Model Driven Engineering), MDA (Model Driven Architecture), MDD (Model Driven Development), DSL, EMF (Eclipse Modeling Framework), GMF (Graphical Modeling Framework) y Epsilon.

Posteriormente nos centraremos en el desarrollo del calendario, donde se detallan las fases y en el se incluye un diagrama de Gantt para facilitar la visualización de éstas.

Luego se incluye la descripción general del proyecto junto con las tecnologías utilizadas.

En el capítulo posterior se detalla todo el proceso de desarrollo de la herramienta.

Luego están las conclusiones, el trabajo futuro y los agradecimientos.

Los últimos capítulos son los manuales de instalación, de usuario y de desarrollador.

También se incluyen al final dos apéndices: código fuente y Subclipse.

Y por último se encuentra la bibliografía.

1.5. Glosario

1.5.1. Acrónimos

CEP Complex Event Processing

CIM Computational Independent Model

CSV Concurrent Versions System

DSL Domain-Specific Language

ECL Epsilon Comparison Language

EGL Epsilon Generation Language

EMF Eclipse Modeling Framework

EML Epsilon Merging Language

EOL Epsilon Object Language

EPL Event Processing Language

ETL Epsilon Transformation Language

1. Introducción y motivación

ETL Epsilon Transformation Language

EVL Epsilon Validation Language

EWL Epsilon Wizard Language

GMF Graphical Modeling Framework

GNU GNU is Not Unix

GPL GNU General Public License

HUTN Human Usable Textual Notation

IDE Integrated Development Environment

M2T Model to Text

MDA Model Driven Architecture

MDD Model Driven Development

MDE Model Driven Engineering

MOF Meta Object Facility

OCL Object Constraint Language

OMG Object Management Group

PFC Proyecto Fin de Carrera

PIM Platform Independent Model

PM Platform Model

PSM Platform Specific Model

SQL Structured Query Language

UCA Universidad de Cádiz

XMI XML Metadata Interchange

1.5.2. Definiciones

Eclipse Es un IDE (Integrated Development Environment) multiplataforma desarrollado inicialmente por IBM y que actualmente es desarrollado por la Fundación Eclipse

Emfatic Es un lenguaje diseñado para representar modelos EMF Ecore de una forma textual.

Epsilon Familia de lenguajes de programación que se usan para interactuar con modelos EMF para realizar tareas de MDE tales como generación de código, transformaciones modelo a modelo, etc.

EuGENia Herramienta que genera los modelos necesarios para crear un editor GMF a partir de un único metamodelo anotado en Ecore.

EOL Lenguaje de programación imperativo que sirve para crear consultar y modificar los modelos EMF. Es el lenguaje principal de Epsilon, y en el que se basan el resto de lenguajes para interactuar con los modelos EMF.

EVL Lenguaje de validación de modelos similar a OCL pero con algunas funcionalidades añadidas.

EGL Lenguaje adaptado para las transformaciones modelo a código (M2T).

EUnit Es un *framework* de pruebas unitarias para validar modelos.

Evento Suceso en el sistema, ya sea un mensaje que se envía o una interacción con el usuario. También se puede considerar un evento algo que se espera que ocurra pero no llega a suceder.

Evento complejo Un evento que es una abstracción de otros eventos, el resultado de la suma de varios eventos simples que están correlacionados entre sí.

Ingeniería Dirigida por Modelos Metodología de desarrollo de software que se centra en la creación y explotación de modelos de dominio más que en los conceptos informáticos.

Patrón de evento Condiciones que han de cumplirse para detectar un evento complejo a partir de eventos simples.

1. Introducción y motivación

2. Estado del arte

2.1. Procesamiento de Eventos Complejos

El Procesamiento de Eventos Complejos o *Complex Event Processing* (CEP) es una tecnología emergente que está orientada a las aplicaciones dirigidas por eventos y está caracterizada por poseer las características necesarias para utilizarse en sistemas que precisen de respuestas en tiempo real [29, 39, 53, 30]. Como ya hemos comentado anteriormente, para conseguirlo, se utilizan unos patrones de eventos que inferirán nuevos eventos más complejos y con un mayor significado semántico, que ayudarán a tomar decisiones ante las situaciones acontecidas. En nuestro proyecto nos centraremos en el lenguaje de diseño de eventos complejos de Esper: EPL.

2.1.1. Características

Un evento es algo que ocurre, ha ocurrido o se espera que ocurra ya sea dentro del sistema o fuera de él. Los eventos tienen asociados una serie de datos, de operaciones y de relaciones entre ellos. Pueden estar basadas en tres tipos de relaciones:

- Temporalidad: El evento contiene información sobre la fecha y hora en el que se ha producido.
- Causalidad: Cuando un evento sucede debido a que otro evento ha ocurrido con anterioridad.
- Agregación: Consiste en la unión de varios eventos.

Un flujo de eventos es una secuencia de eventos ordenados en el tiempo. Existen dos tipos:

- Homogéneo: Eventos del mismo tipo.
- Heterogéneo: Eventos de distinto tipo.

El objetivo principal es filtrar los eventos que nos interesan y desechar los que no. Para realizar ese filtro, se definen los patrones de eventos complejos. De esta manera observamos las relaciones que existen en los eventos y obtenemos las situaciones complejas. Este filtro se hace en tiempo real, ya que los eventos pueden ser detectados

2. Estado del arte

mediante esta tecnología de manera inmediata. En nuestro caso, en el ámbito de la seguridad informática, es esencial que los eventos puedan ser detectados en tiempo real debido a la importancia de responder de la manera más rápida posible a los ataques recibidos.

2.1.2. Ventajas

Las ventajas del uso de CEP se pueden resumir en los siguientes puntos:

- Escalabilidad, ya que se pueden extender aplicaciones de una manera flexible.
- Separación de lógica de procesamiento de eventos con respecto a la aplicación.
- Capacidad de recibir grandes cantidades de datos.
- Capacidad de trabajo en tiempo real.
- Disminución en costes de desarrollo y mantenimiento debido a la abstracción en el manejo de los eventos.

2.1.3. Patrones

Los patrones (que están definidos dentro del motor) detectan eventos complejos que envían al *listener* correspondiente.

Se propone la siguiente clasificación de patrones proporcionados en [53]. Está compuesta por los siguientes puntos:

- Buenas y malas soluciones: Se distinguen entre los patrones CEP, que son los que ofrecen soluciones y los anti-patrones CEP que son los patrones que producen consecuencias negativas.
- Niveles de abstracción:
 - Pautas y buenas prácticas.
 - Patrones de gestión.
 - Patrones de diseño.
 - Patrones de mapeo.
 - Patrones de ejecución.
 - Patrones de refactorización.
- Objetivos a alcanzar: Distingue los patrones acorde al objetivo a alcanzar.
 - Patrones de adopción.

- Patrones de negocios.
 - Patrones de integración.
 - Patrones compuestos.
 - Patrones de flujos de trabajo.
 - Patrones de coordinación.
 - Patrones personalizados.
 - Patrones de aplicación.
- Niveles de gestión:
- Patrones estratégicos.
 - Patrones tácticos.
 - Patrones operacionales.

2.2. Esper

El motor CEP utilizado en este PFC es Esper, que provee el lenguaje EPL para el procesamiento de eventos complejos. Como hemos comentado anteriormente, EPL es el lenguaje que se utiliza actualmente en el Grupo de Investigación UCASE. Este motor CEP es desarrollado por *EsperTech Inc.* [18] y distribuido mediante la licencia GPL (GNU General Public License) v2 [1]. Esper proporciona EPL, que es un lenguaje de procesamiento de eventos complejos para expresar filtrado, agregación, y unión, a través de ventanas deslizantes de múltiples flujos de eventos. También incluye la semántica del patrón para expresar causalidad temporal compleja entre eventos (relaciones *followed-by*) [18]. En la Figura 2.1 (extraída de [18]) se observa el procesamiento de flujos de eventos y la correlación.

Las tres partes principales de la arquitectura de Esper son los eventos, los patrones de eventos y las interfaces que notifican eventos (*listeners*). En la Figura 2.2 (extraída de [18]) se puede observar la arquitectura.

El motor se encarga de analizar un flujo entrante de eventos y de filtrar los eventos que nos sean de utilidad y puedan ser relacionados con los patrones de eventos complejos diseñados previamente. La entrada de eventos puede estar implementada en diferentes formatos como pueden ser objetos planos de Java o *Plain Old Java Object* (POJO), documentos XML y *map* de Java.

Esper trabaja en un flujo continuo, en el que el flujo de eventos de entrada está siendo constantemente evaluado por las sentencias EPL diseñadas para encontrar un patrón en concreto, intentando identificar el patrón en el flujo de eventos de entrada del motor. Cuando llega un evento a través del flujo, se compara con todos

2. Estado del arte

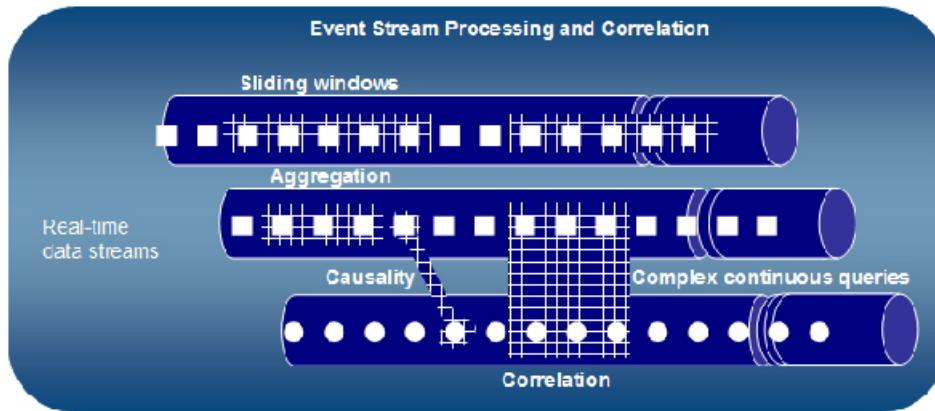


Figura 2.1.: Procesamiento de flujos de eventos y correlación

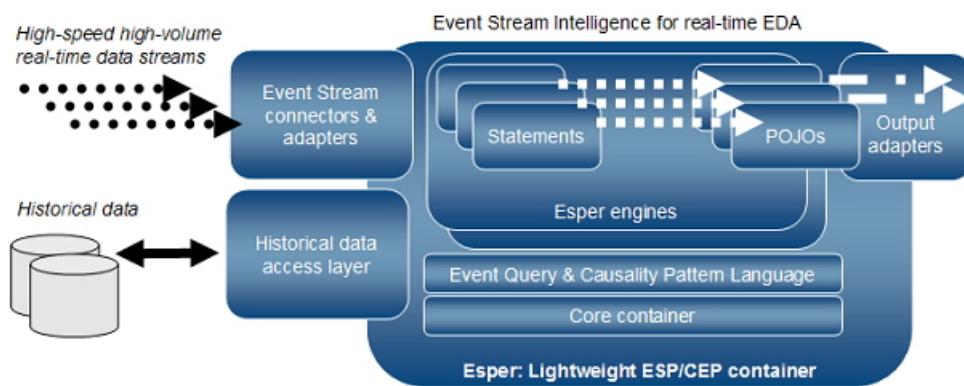


Figura 2.2.: Arquitectura Esper

2.2. Esper

los patrones activos y se notifica a los *listener* cuando se obtiene una relación entre un evento y un patrón.

Una expresión de un patrón en EPL está formada por átomos de patrones y operadores. Los átomos se dividen en las siguientes categorías [17]:

- Expresiones de filtro: Son expresiones que se le añaden a un evento para encontrar un patrón. Su sintaxis es poner primero el nombre del evento y luego entre paréntesis la expresión con las condiciones del filtrado (e.g. en el Listado 2.1).

Listado 2.1: Ejemplo de expresión de filtro

```
every A(PortSource != 80)
```

- Observadores de eventos basados en tiempo: Especifican el tiempo en intervalos o un tiempo determinado. Los observadores son **timer:interval**, que sirve para especificar el tiempo en un intervalo y **timer:at**, que sirve para especificar un tiempo determinado (e.g. en el Listado 2.2).

Listado 2.2: Ejemplo de observador de eventos basados en tiempo

```
A -> timer:interval(10 seconds)
```

Los operadores existentes en EPL son los siguientes:

- Operadores temporales: Operan sobre el orden de los eventos. La expresión de la izquierda del operador tiene que ser verdadera para que se pueda evaluar la expresión de la derecha. Operador *followed-by* (->) (e.g. en el Listado 2.3).

Listado 2.3: Ejemplo de operador temporal

```
A -> B
```

- Operadores de repetición: Controlan las repeticiones de las subexpresiones de los patrones. Los diferentes operadores de repetición son **every**, **every-distinct**, **[num]** y **until**. Al operador **every** le sigue una expresión. Cuando la expresión sea evaluada, se reinicia la subexpresión del patrón. Al utilizar **every-distinct** se consiguen eliminar los resultados duplicados. El operador **[num]** obliga a que la expresión que le sigue debe de ser verdadera las veces que indica el operador. El operador **until** busca eventos del tipo especificado a la izquierda del operador hasta que haya eventos del tipo indicado a la derecha del operador. Ejemplos en el Listado 2.4

2. Estado del arte

Listado 2.4: Ejemplos de operadores de repetición

```
every A  
every-distinct (s.sensor) s = sample  
[5] A  
A until B
```

- Operadores lógicos: Los operadores lógicos que ofrece el lenguaje son **and**, **or** y **not**. El operador **and** es utilizado para que para que la expresión sea verdadera, ambas expresiones deben de ser verdaderas. En el caso de **or** si una de las expresiones es verdadera, la expresión completa es verdadera. El operador **not** niega el valor de la expresión indicada posteriormente del operador, es decir, si la expresión es verdadera, se convierte en falsa y viceversa. Ejemplos en el Listado 2.5.

Listado 2.5: Ejemplos de operadores lógicos

```
A and B  
C or D  
A -> B and not C
```

- Guardas: Son las condiciones **where** las que controlan el ciclo de vida de las subexpresiones. Los operadores son **timer:within**, **timer:withinmax** y **while**. El operador **timer-within** indica un intervalo de tiempo en el que debe de evaluarse la expresión como verdadera. El operador **timer-withinmax** indica el mismo intervalo de tiempo que **timer:within** pero con la diferencia de que **timer:withinmax** también indica un contador. La expresión finaliza cuando se alcanza el tiempo especificado o cuando se llega a las repeticiones indicadas en el contador. El operador **while** va seguido de una expresión que debe de ser evaluada para cada relación. Cuando la expresión es falsa, la subexpresión del patrón termina. Ejemplos en el Listado 2.6.

Listado 2.6: Ejemplos de guardas

```
A where timer:within(5 seconds)  
B where timer:within(5 seconds , 6)
```

2.3. Desarrollo de Software Dirigido por Modelos

La ingeniería dirigida por modelos o *Model Driven Engineering* (MDE) es una metodología de desarrollo de software que se centra en la creación y explotación de modelos de dominio más que en los conceptos informáticos [57]. Este enfoque surge debido a que hasta ahora, todo el desarrollo software está enfocado en la programación, mientras que en otras ingenierías se utilizan los modelos como referencia, es decir, el nivel de abstracción es mayor. Por lo tanto, las ventajas que se plantean son las siguientes:

- Mayor nivel de abstracción.
- Mayor nivel de reutilización.
- Mayor interoperabilidad.

MDE es la manera genérica de definir MDD y MDA.

2.3.1. Conceptos

Definamos qué es un modelo:

- Una descripción de (parte de) un sistema escrito en un lenguaje bien definido [44].
- Una representación de una parte de la función, estructura y/o comportamiento de un sistema [52].
- Una descripción o especificación del sistema y de su entorno para un propósito determinado. Un modelo que es presentado a menudo como una combinación de dibujos y texto [50].
- Un conjunto de afirmaciones sobre el sistema [55].

Definamos ahora qué es un metamodelo:

- Un modelo de un lenguaje bien definido [44].
- Un modelo de modelos [52].
- Un modelo de un lenguaje de modelado [55].

Las cuatro capas de la arquitectura del metamodelado OMG se pueden visualizar en la Figura 2.3 (extraída de [37]):

2. Estado del arte

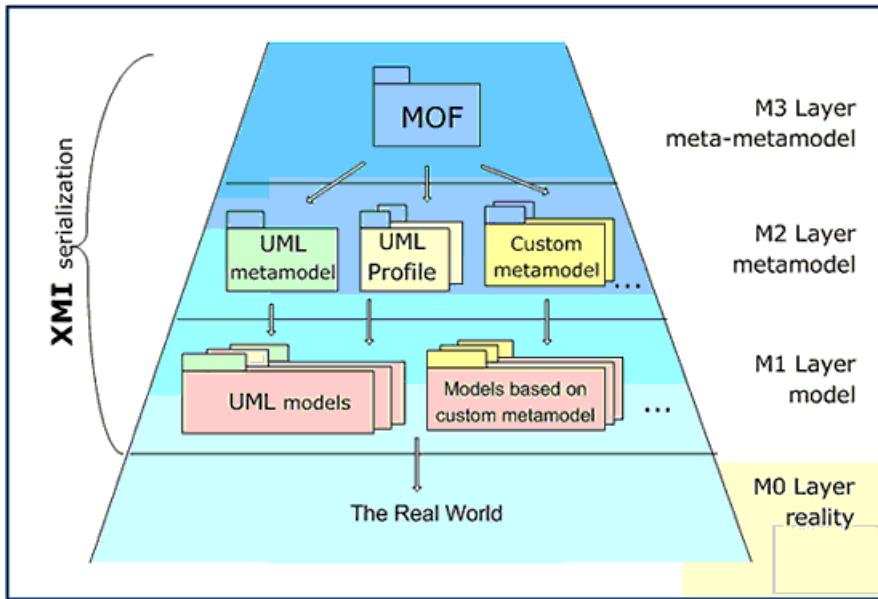


Figura 2.3.: Capas de la arquitectura del metamodelado OMG

2.3.2. Motivación

En [41] los autores afirman que aún no hay una industria del software, proporcionando los siguientes datos de [58]:

- En Estados Unidos hay un gasto de unos 250 mil millones anualmente en el desarrollo de software, con un coste medio de 430 mil dólares a 2 millones 300 mil dólares por empresa.
- Sólo el 16 % de los proyectos son completados dentro del presupuesto y a tiempo.
- Un 31 % de los proyectos son cancelados, debido principalmente a problemas de calidad, creando pérdidas sobre 81 mil millones de dólares anualmente.
- Otro 53 % tienen más coste que los planeados, excediendo el presupuesto en una media de un 189 %, creando pérdidas de unos 59 mil millones de dólares anualmente.
- Los proyectos que llegan a ser finalizados, sólo contienen un 42 % de las características originales.

En estos años, la complejidad de las aplicaciones demandadas por las empresas aumenta, debido a los cambios frecuentes de tecnologías y debido a la extensión

2.3. Desarrollo de Software Dirigido por Modelos

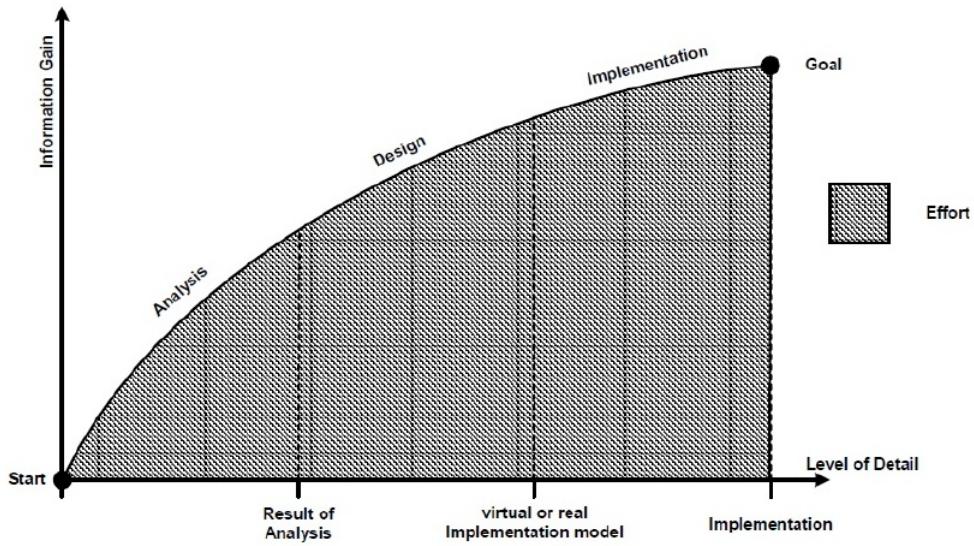


Figura 2.4.: Desarrollo de software tradicional

de los sistemas distribuidos y heterogéneos. Según los autores en [41], la manera de poder afrontar este tipo de problemas es con una industria del software que abandone las prácticas artesanales. Como respaldo, el autor de [42] afirma que la productividad en la industria del software está disminuyendo. Explica cómo las empresas se están dedicando en automatizar los procesos, que hacen que los problemas puedan ser detectados con más facilidad por los desarrolladores.

En la Figura 2.4 (extraída de [57]) se muestra una gráfica sobre el desarrollo del software tradicional y en la Figura 2.5 (extraída también de [57]) se muestra una gráfica con el desarrollo de software con MDE.

2.3.3. Arquitectura Dirigida por Modelos

La arquitectura dirigida por modelos o MDA comienza con la idea bien conocida y bien establecida de separar la especificación de la operación de un sistema, de los detalles de implementación. Sus tres objetivos principales son la portabilidad, interoperabilidad y reusabilidad [50].

Los niveles MDA que se proponen son:

- CIM (Computational Independent Model): Es una visión de un sistema desde un punto de vista independiente de la computación. También es llamado modelo de dominio. Ejemplo: Modelos de procesos de negocio (por ejemplo BPMN).

2. Estado del arte

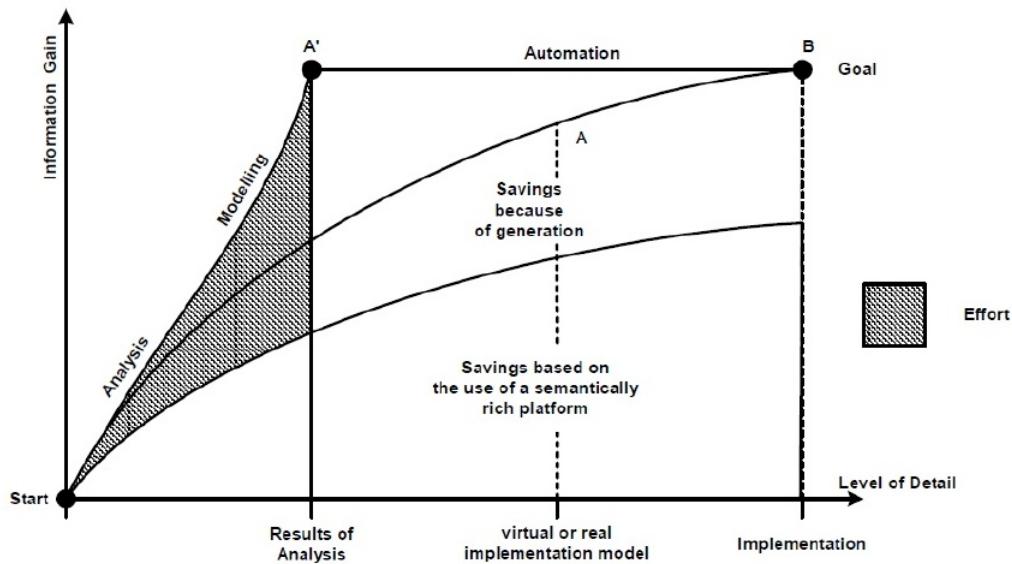


Figura 2.5.: Desarrollo de software con MDE

- PIM (Platform Independent Model): Es una visión de un sistema desde un punto de vista independiente de la plataforma. Ejemplos: Modelo de casos de uso, Modelo entidad/relación.
- PSM (Platform Specific Model): Es una visión de un sistema desde un punto de vista de la plataforma específica. Ejemplo: Diagrama de clases de diseño.

En la Figura 2.6 (extraída de [54]) se observa el proceso de desarrollo de MDA.

Cuando se desarrolla una aplicación mediante MDA, se especifica la lógica del negocio con modelos PIM que están expresados en un lenguaje de modelado. A partir de los modelos PIM especificados, se podrán generar modelos específicos de

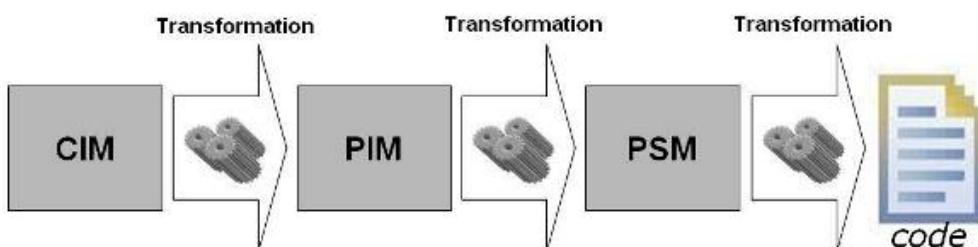


Figura 2.6.: Proceso de desarrollo MDA

2.3. Desarrollo de Software Dirigido por Modelos

las plataformas y el código necesario para el funcionamiento de la aplicación.

El lenguaje de metamodelado MOF (Meta Object Facility) es uno de los más utilizados y es uno de los estándares de MDA. MOF por tanto es un lenguaje que tiene la capacidad de crear metamodelos. Podemos distinguir dentro de MOF [51]:

- EMOF Model (Essential MOF Model): Es un subconjunto de MOF. El objetivo principal de EMOF es permitir que simples metamodelos puedan ser definidos usando conceptos simples mientras que soportan extensiones para el metamodelado más sofisticado usando CMOF.
- CMOF Model (Complete MOF Model): Es el metamodelo usado para especificar otros metamodelos tales como UML 2.

En Eclipse se utiliza el lenguaje de metamodelado Ecore [7] que es cercano a MOF 2.0 EMOF [8, 40].

El *framework* que hemos utilizado en este proyecto para crear editores específicos para metamodelos es GMF [23], que se basa en crear editores gráficos para EMF [59].

2.3.4. Principios de Desarrollo Dirigido por Modelos

Los lenguajes utilizados en el desarrollo dirigido por modelos o *Model Driven Development* (MDD) son los lenguajes denominados lenguajes específicos del dominio (DSL) o lenguajes de modelado. Son lenguajes orientados a un problema específico, por lo que son opuestos a los lenguajes de propósito general. Los DSL siguen la teoría descrita en [60], donde existen los siguientes conceptos:

- Lenguajes de metamodelado: Son los lenguajes que se utilizan para describir otros lenguajes.
- Sintaxis abstracta: Define los conceptos del lenguaje y las relaciones. Además, también se definen un conjunto de restricciones para poder diferenciar entre un modelo que se puede expresar con el lenguaje de metamodelado y cuál no.
- Sintaxis concreta: Es una notación para expresar los modelos que forman parte de la sintaxis abstracta.
- Semántica: Es el significado de los conceptos y relaciones del lenguaje.
- Transformaciones: El modelo creado en el DSL es necesario que se transforme posteriormente a código. Estas transformaciones pueden realizarse en uno o más pasos en los que se pueden transformar en otro modelo o directamente a código, por lo que existen las transformaciones de modelo a modelo (M2M) y las transformaciones de modelo a código (M2T), respectivamente. En nuestro proyecto nos centramos en la transformación de M2T y no de M2M, ya que transformamos los modelos creados directamente en código EPL.

2. Estado del arte

2.4. Lenguaje Específico del Dominio

Un lenguaje específico del dominio o DSL es un pequeño, por lo general declarativo lenguaje que ofrece una potencia expresiva centrada en el dominio de un determinado problema [36].

En todas las ramas de la ingeniería, se pueden distinguir entre los enfoques genéricos y los enfoques específicos. El enfoque genérico provee una solución general para muchos problemas en un área determinada pero esa solución puede no ser óptima completamente. Un enfoque específico provee una solución más óptima para un pequeño conjunto de problemas.

Esto no es un tema nuevo, ya que los lenguajes de programación más antiguos (Cobol, Fortran, Lisp) nacieron como lenguajes especializados para resolver problemas en un área determinada, en estos casos respectivamente fueron el procesamiento de los negocios, cálculo numérico y procesamiento simbólico. Sin embargo, gradualmente se han ido convirtiendo en lenguajes de propósito general, por lo que han surgido de nuevo los lenguajes con un enfoque específico para resolver problemas determinados. [36].

Las ventajas que se proponen en [36] son las siguientes:

- Permiten soluciones expresadas en el lenguaje y en el nivel de abstracción del dominio del problema.
- Los programas DSL son concisos, con gran documentación y pueden ser reutilizados con diferentes propósitos [32].
- Mejoran la productividad, fiabilidad, mantenibilidad y portabilidad [35, 43].
- Incorporan conocimiento del dominio y permiten así la conservación y reutilización de este conocimiento.
- Permiten la validación y optimización del nivel del dominio [31].
- Mejora las pruebas de los siguientes enfoques tales como [56].

Las desventajas propuestas son:

- Costes de diseño, implementación y mantenimiento.
- Coste de la enseñanza del DSL a los usuarios.
- Limitada disponibilidad [46].
- Dificultad de encontrar un ámbito adecuado.

2.5. Eclipse Modeling Framework

- Dificultad para encontrar el balance entre la especificidad del dominio y el propósito general.
- La posible pérdida de eficiencia en comparación con software hecho a mano.

Varios ejemplos de DSL se mencionan en [36] por temas y se pueden acceder a sus citas en el artículo mencionado:

- Ingeniería del software: Productos financieros, control de comportamiento y coordinación, arquitecturas software y bases de datos.
- Sistemas software: Descripción y análisis de árboles sintácticos abstractos, especificaciones de drivers de dispositivos de vídeo, protocolos de coherencia de caché, estructuras de datos en C y especialización de sistemas operativos.
- Multimedia: Computación Web, manipulación de imágenes, animación 3D y dibujo.
- Telecomunicaciones: Protocolos de comunicación.
- Misceláneo: Simulación, agentes móviles, control y robótica, y diseño de hardware digital.

En nuestro caso, hemos construido un DSL basado en el ámbito de la seguridad informática que es el encargado de proporcionar al usuario la construcción de patrones de eventos complejos en una interfaz gráfica y su posterior transformación a código EPL.

2.5. Eclipse Modeling Framework

Eclipse Modeling Framework o EMF es un framework de modelado y generación de código la construcción de herramientas y otras aplicaciones basadas en un modelo de datos estructurado. A partir de la especificación de un modelo descrito en XMI (XML Metadata Interchange) [27], EMF proporciona herramientas y soporte de ejecución para producir un conjunto de clases Java para el modelo, junto con un conjunto de clases adaptadoras que permiten la visualización y edición del modelo, y un editor básico [8].

El modelo usado para representar modelos en EMF es llamado Ecore. Ecore en sí mismo es un modelo EMF y es su propio metamodelo. Podríamos decir que construye también su meta-metamodelo. Un metamodelo es simplemente el modelo de un modelo, y si ese modelo es en sí mismo un metamodelo, entonces el metamodelo es de hecho un meta-metamodelo [59].

2. Estado del arte

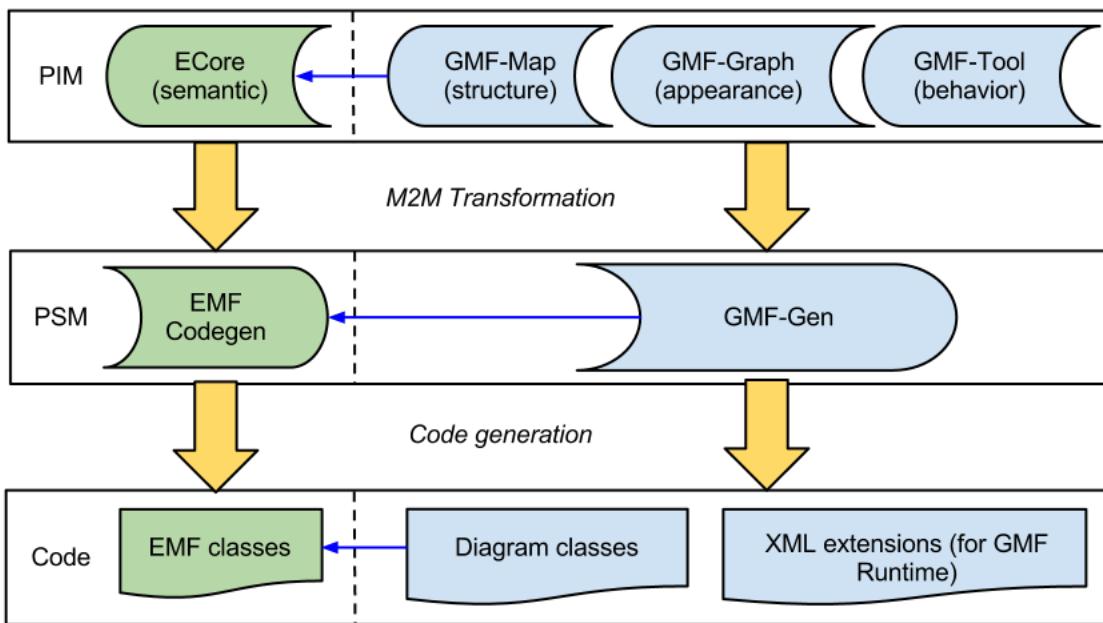


Figura 2.7.: Estructura GMF

2.6. Graphical Modeling Framework

El proyecto Eclipse Modeling Framework o GMF es un conjunto de componentes generativos e infraestructuras en tiempo de ejecución para el desarrollo de editores gráficos basados en EMF y GEF [23]. En la Figura 2.7 (extraída de [23]) se puede observar la estructura de GMF.

En este PFC se ha utilizado GMF mediante EuGENia, que será explicado en el siguiente capítulo.

2.7. Epsilon

Epsilon es una plataforma que proporciona una familia de lenguajes consistentes e interoperables, específicos de tareas para la gestión de modelos tales como la transformación de modelos, generación de código, comparación de modelos, fusión, refactorización y validación [45].

Epsilon provee los siguientes lenguajes:

- EOL (Epsilon Object Language)
- EVL (Epsilon Validation Language)

2.7. Epsilon

- ETL (Epsilon Transformation Language)
- ECL (Epsilon Comparison Language)
- EML (Epsilon Merging Language)
- EWL (Epsilon Wizard Language)
- EGL (Epsilon Generation Language)

Para cada lenguaje, Epsilon provee herramientas basadas en Eclipse y un intérprete que puede ejecutar programas escritos en el lenguaje.

Epsilon además contiene varias herramientas y utilidades que complementan la experiencia en EMF/GMF tales como un asistente para generar editores GMF (EuGENIA), una implementación de HUTN (Human Usable Textual Notation), una vista del registro EPackage, un editor personalizable de modelos EMF basado en árboles, un editor multi-vista para establecer referencias cruzadas de modelos (Modelink) y un framework para la gestión de tareas (EUnit).

En el PFC nos hemos centrado en los siguientes lenguajes y herramientas:

- **Epsilon Object Language.** Es un lenguaje de programación imperativo para crear, consultar y modificar modelos EMF. Provee todas las características imperativas usuales de Javascript (e.g. secuencia de declaraciones, variables, bucles for y while, if, etc) y todas las buenas características de OCL tales como funciones prácticas sobre consultas en colecciones (e.g. `Sequence{1..5}.select(x|x>3)`) [14]. Características:
 - Soporte para acceso y modificación simultáneo de modelos o de diferentes metamodelos.
 - Todas las construcciones normales de programación.
 - Soporte para las operaciones lógicas de primer orden de OCL.
 - Capacidad para crear y llamar a objetos y métodos Java.
 - Soporte para fijar de forma dinámica las operaciones existentes en metaclasses y tipos en tiempo de ejecución.
 - Soporte para almacenar operaciones.
 - Soporte para propiedades extendidas.
 - Soporte para la interacción con el usuario.
 - Capacidad para reutilizar bibliotecas de operaciones e importarlas de diferentes módulos de Epsilon (no sólo de EOL).

2. Estado del arte

EOL ha sido utilizado en este PFC para poder realizar *scripts* que permitan personalizar el editor gráfico final.

- **Epsilon Validation Language.** Es un lenguaje construido por encima de EOL. En su forma simple, las restricciones EVL son bastante similares a las restricciones en OCL. Sin embargo, EVL también soporta dependencias entre restricciones (e.g. si la restricción A falla, no se evalua la restricción B), mensajes de errores personalizables para ser mostrados al usuario y especificación de los arreglos de los errores (en EOL). También, como EVL está construido sobre EOL, se pueden evaluar restricciones entre modelos (a diferencia de OCL) [16]. Características:

- Distinción entre errores y avisos durante la validación.
- Arreglos específicos para restricciones que no se cumplen.
- Restricciones guarda.
- Dependencias específicas de restricciones.
- Evaluación automática de restricciones.
- Integración con la validación de EMF y GMF.

También hereda todas las características del lenguaje EOL, comentadas anteriormente. La sintaxis concreta del lenguaje se puede observar en el Listado 2.7 y en el Listado 2.8 la sintaxis concreta para el arreglo de un error.

Listado 2.7: Sintaxis concreta EVL

```
(@lazy)?
(constraint | critique) name {
    (guard (:expression)|({statementBlock}))?
    (check (:expression)|({statementBlock}))?
    (message (:expression)|({statementBlock}))?
    (fix)*
}
```

Listado 2.8: Sintaxis concreta para el arreglo de un error en EVL

```
fix {
    (guard (:expression)|({statementBlock}))?
```

2.7. Epsilon

```
( title (:expression)|({statementBlock}))?  
do {  
    statementBlock  
}  
}
```

Este lenguaje ha sido utilizado en el PFC para poder validar los modelos diseñados por el usuario en el editor gráfico.

- **Epsilon Generation Language.** Es un lenguaje basado en plantillas M2T para la generación de código, documentación y otros artefactos textuales de los modelos. Soporta la mezcla de regiones generadas automáticamente con otras escritas a mano [12]. Hereda también todas las características de EOL al igual que EVL. En el Listado 2.9 se puede observar una plantilla básica en EGL. En el Listado 2.10 se encuentra el texto generado a partir de la plantilla EGL del Listado 2.9.

Listado 2.9: Plantilla básica en EGL

```
[% for (i in Sequence{1..5}) { %]  
i is [%=i %]  
[% } %]
```

Listado 2.10: Texto generado

```
i is 1  
i is 2  
i is 3  
i is 4  
i is 5
```

EGL se ha utilizado en el PFC para realizar las transformaciones de modelo a código.

- **EuGENia.** Es una herramienta que genera automáticamente los modelos .gmfgraph, .gmftool y .gmfmap necesarios para implementar un editor GMF a partir de un único metamodelo Ecore. Provee anotaciones de alto nivel que abstraen de la complejidad de GMF y baja la barrera de dificultad para crear un editor GMF sin previa experiencia. Mientras EuGENia es muy útil para empezar con GMF, no se detiene ahí y puede ser usado para pulir la versión final del editor [19]. En este PFC se ha utilizado EuGENia para poder generar el editor GMF a partir del metamodelo Ecore diseñado previamente y anotado textualmente mediante Emfatic.

2. Estado del arte

- **Epsilon Unit Testing Framework (EUnit).** Es un *framework* de pruebas unitarias construido sobre la plataforma Epsilon. Está especialmente diseñado para probar las tareas de gestión de modelado, tales como transformaciones modelo a modelo, transformaciones M2T y validaciones de modelos, entre otros. EUnit puede ser usado para probar cualquier tarea de gestión de modelado que se expone a través de una tarea Ant, incluso si no es parte de la plataforma Epsilon [22]. Características:

- Reutiliza pruebas a través de diferentes conjuntos de datos o modelos.
- Configura pruebas y tareas de modelado que son realizadas a través de tareas Ant.
- El desmontaje de la prueba es implícita: los modelos son recargados automáticamente.
- Compara modelos, ficheros y directorios de forma transparente con afirmaciones incluidas.
- Genera reportes de las pruebas en el formato XML.
- Proporciona una vista de los resultados de la prueba y compara las diferencias gráficamente en Eclipse.
- Generación de los modelos dentro de la prueba usando EOL.
- Carga los modelos para ser usados en la prueba a partir de los fragmentos HUTN.

Hereda todas las características de EOL al estar construido sobre él. EUnit se ha utilizado para validar las transformaciones realizadas por EGL en el PFC.

3. Desarrollo del calendario

A continuación se detallan las fases bien diferenciadas en las que está constituido el proyecto.

3.1. Fases

A continuación se exponen las fases bien diferenciadas en las que se ha dividido el proyecto. Se ha seguido el modelo de ciclo de vida lineal secuencial (explicado con detalle en el Capítulo 5).

3.1.1. 1^a fase: Conocimientos previos ylicitación de requisitos

En esta fase se tuvo un primer contacto con el grupo de investigación UCASE, en el que los miembros del grupo describieron las distintas líneas de investigación y en el que se ubicó el presente PFC.

Posteriormente se realizaron varias reuniones con el tutor, donde se tuvo una idea general del proyecto, se pudieron definir algunos objetivos, tecnologías y se proporcionó bibliografía de utilidad.

3.1.2. 2^a fase: Búsqueda de información y asimilación de conceptos

Esta fase se basó principalmente en la búsqueda bibliográfica. Al principio, la búsqueda bibliográfica se centró en los conceptos de CEP, Esper y MDD para poder enmarcar el proyecto. Posteriormente se hizo una búsqueda exhaustiva de las diferentes tecnologías que se podían emplear para llevar a cabo el PFC, donde nos decantamos por el uso completo de los lenguajes que provee Epsilon. Cabe destacar que esta fase se podría incluir en la primera también, ya que estas dos fases se han realizado en paralelo debido a que las dos fases han sido para poder enmarcar el proyecto y afrontarlo con garantías en las fases sucesivas.

3. Desarrollo del calendario

3.1.3. 3^a fase: Diseño de metamodelo

Para poder llevar a cabo el PFC, era necesario diseñar un metamodelo que incluya una gran parte de los modelos que pueden ser creados a partir del lenguaje EPL. Esta fase era decisiva en el proyecto, porque sin un metamodelo bien diseñado no se podría haber realizado el resto de las fases del PFC.

3.1.4. 4^a fase: Familiarización de tecnologías

En esta fase se elaboraron varios editores basados en EuGENia [19], intentando aprovechar al máximo las herramientas que ofrece y así aplicarlo a nuestro problema en concreto. También se utilizaron las herramientas de validación y de generación de código que ofrece Epsilon tales como EVL y EGL.

3.1.5. 5^a fase: Creación de metamodelo en EuGENia y personalización del editor

Esta fase se basa en la integración del metamodelo con el editor a partir de la herramienta EuGENia, que utiliza Emfatic [9] para realizar la “traducción” del metamodelo Ecore a GMF. Además, se han personalizado las herramientas del editor, tales como la paleta gráfica, el entorno principal, las etiquetas de los elementos, los mensajes en el editor u otras funciones que la herramienta EuGENia no soporta de base pero son configurables a través de *scripts*.

3.1.6. 6^a fase: Validación de los modelos creados en el editor

Para que los modelos creados en el editor sean correctos, necesitan de un proceso de validación. Esta validación se ha conseguido gracias a la herramienta EVL, que es un lenguaje similar a OCL [4] pero que añade más funcionalidad.

3.1.7. 7^a fase: Transformación de los modelos definidos gráficamente a código Event Processing Language

Una vez creados los modelos y verificados que son correctos, se procede a la transformación M2T de los modelos creados a código EPL gracias al lenguaje EGL. Al principio de conseguir la transformación, la transformación se llevaba a cabo sin la integración de la transformación en el editor. Posteriormente se desarrolló un *plugin* de Eclipse que integraba la generación de código a EPL en el propio editor gráfico.

3.2. Diagrama de Gantt

3.1.8. 8^a fase: Validación de las transformaciones realizadas sobre los modelos

Para validar las transformaciones realizadas mediante pruebas unitarias, se ha utilizado EUnit. Esta validación se ha realizado en la última etapa del proyecto cuando ya se tenía la versión definitiva de la aplicación. De esta manera se comprueba que la generación de código se ha realizado correctamente a partir de los modelos diseñados.

3.1.9. 9^a fase: Redacción de la memoria del Proyecto Fin de Carrera

La redacción de la memoria del PFC se ha realizado durante todo el desarrollo del proyecto, ya que al seguir el modelo de ciclo de vida lineal secuencial hay que documentar las fases realizadas una vez que llegan a su fin.

3.2. Diagrama de Gantt

A continuación se incluye un diagrama de Gantt Figura 3.1, Figura 3.2 y Figura 3.3 donde se reflejan las fases descritas anteriormente. Para la elaboración del diagrama se ha empleado la herramienta de código abierto GNOME Planner disponible en <https://live.gnome.org/Planner>.

3. Desarrollo del calendario

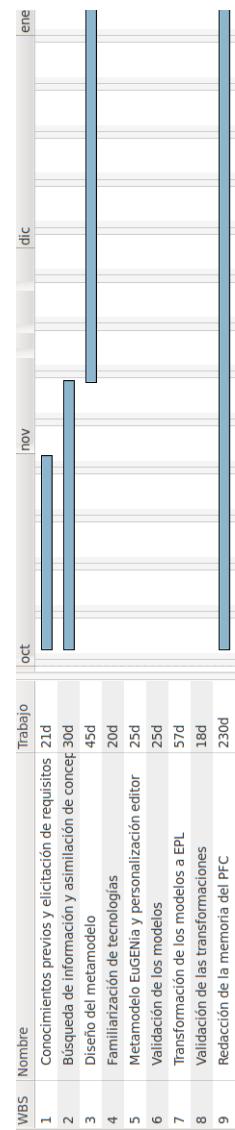


Figura 3.1.: Diagrama de Gantt octubre-diciembre

3.2. Diagrama de Gantt

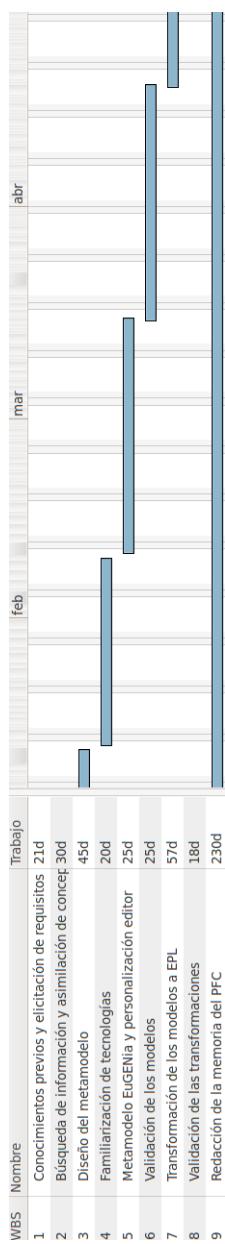


Figura 3.2.: Diagrama de Gantt enero-abril

3. Desarrollo del calendario

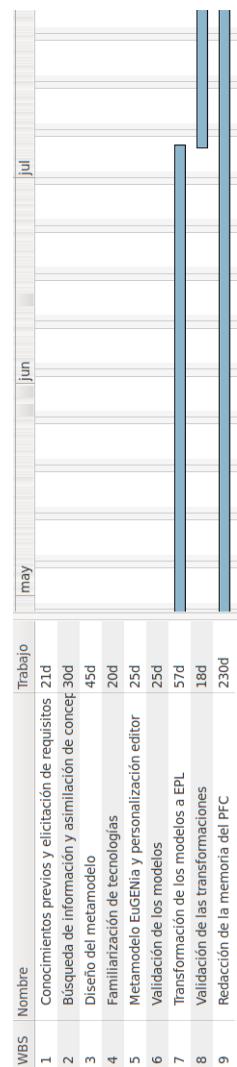


Figura 3.3.: Diagrama de Gantt mayo-agosto

4. Descripción general del proyecto

4.1. Perspectiva del producto

A continuación se detallará información de la herramienta desarrollada como el entorno, la interfaz de usuario, funciones, características generales y restricciones generales.

4.1.1. Entorno del producto

La herramienta SecurityEPLdraw se puede enmarcar como un DSL orientado a la seguridad informática que utiliza técnicas de MDE para la creación del editor gráfico, validación de modelos, generación de código y pruebas unitarias. Es una herramienta desarrollada desde el inicio sin ser continuación de ningún otro proyecto.

Para llevar a la práctica el MDE, SecurityEPLdraw utiliza la familia de lenguajes de programación Epsilon para conseguir el editor GMF final con las validaciones y generación de código correspondientes.

4.1.2. Interfaz de usuario

En las figuras Figura 4.1, Figura 4.2, Figura 4.3 y Figura 4.4 podemos visualizar la interfaz de usuario de la herramienta SecurityEPLdraw.

4.2. Funciones

Las funciones de SecurityEPLdraw son las siguientes:

- Creación de modelos para diseñar patrones de eventos complejos.
- Validación de los modelos creados.
- Transformación de los modelos diseñados a código EPL.

4. Descripción general del proyecto

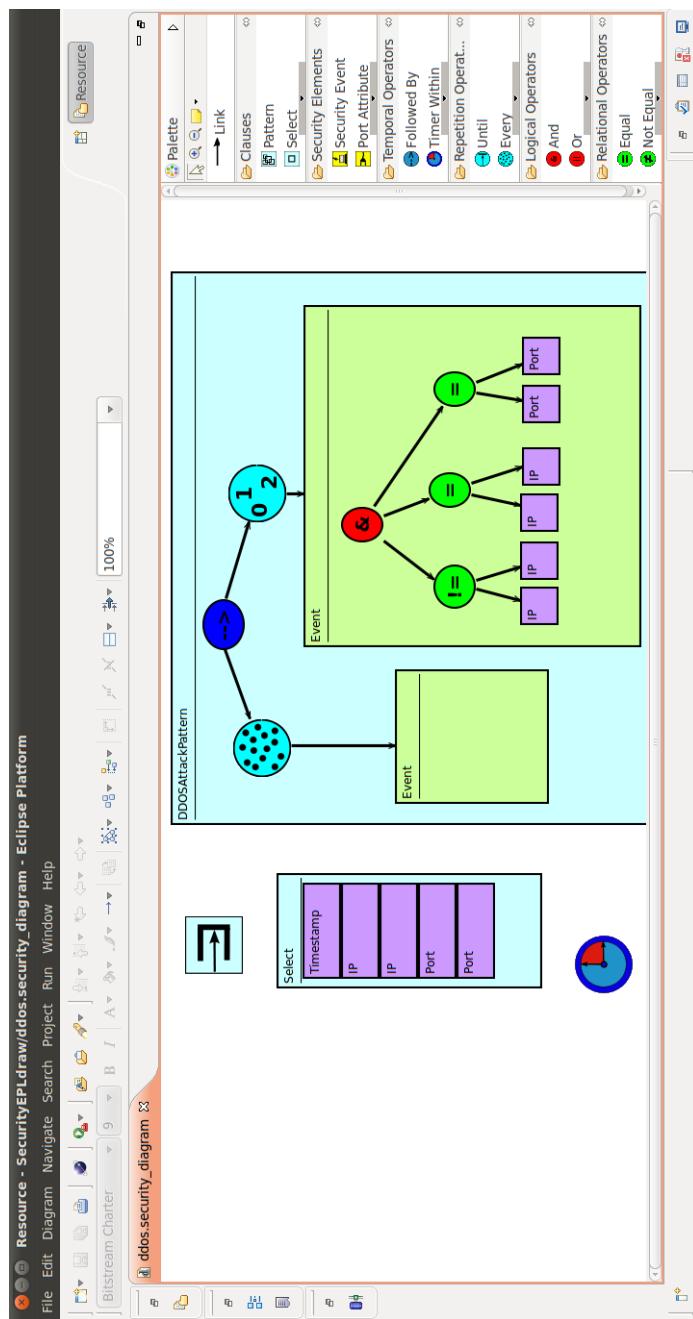


Figura 4.1.: Ejemplo gráfico de definición de un patrón complejo

4.2. Funciones

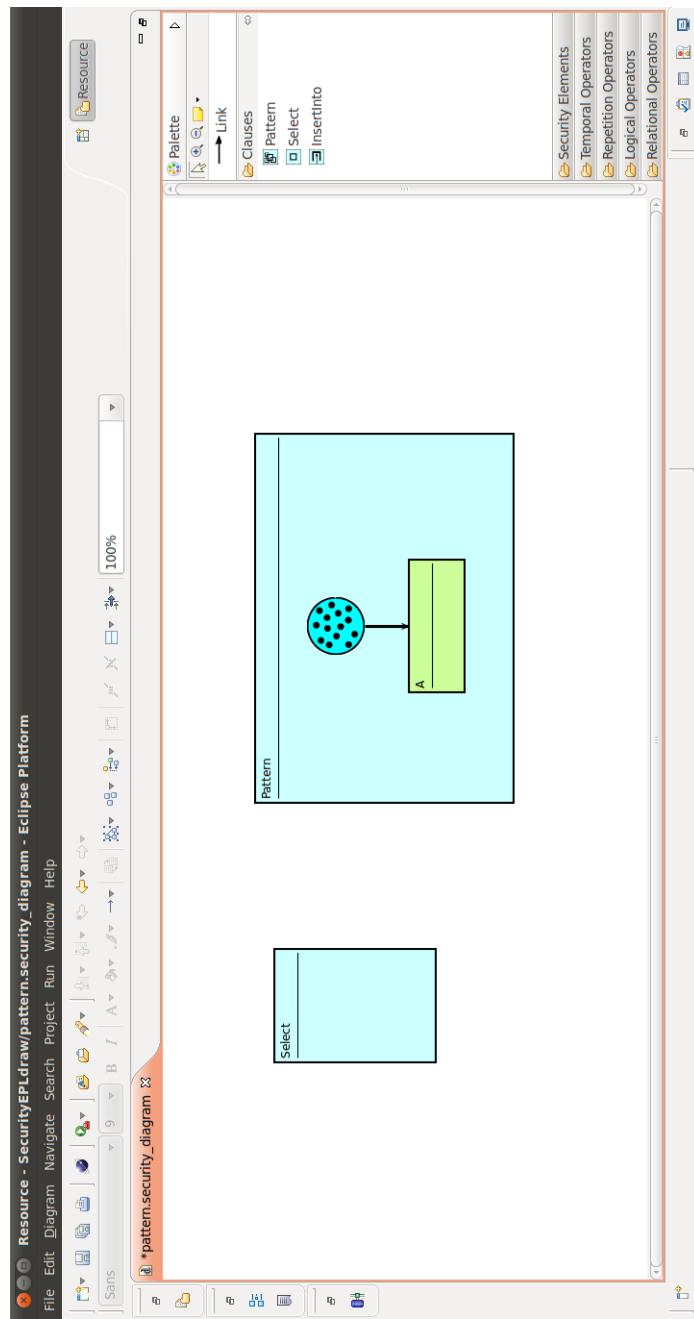


Figura 4.2.: Ejemplo gráfico de definición de un patrón sencillo

4. Descripción general del proyecto

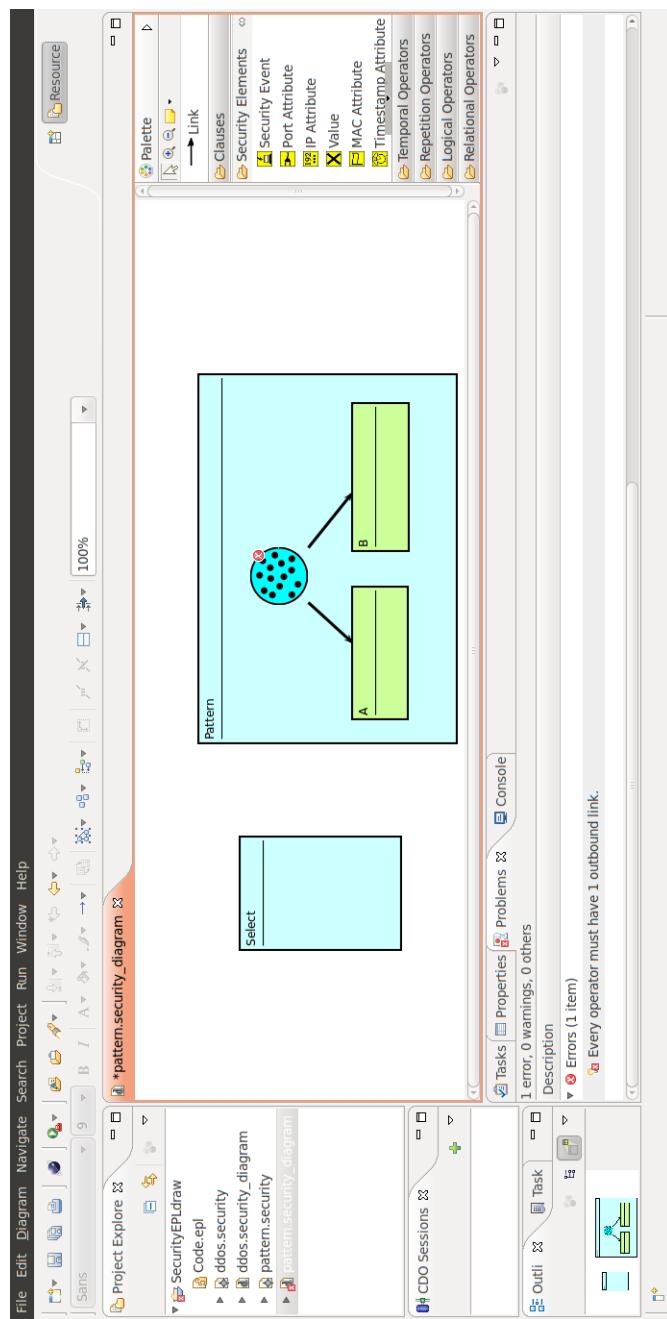


Figura 4.3.: Ejemplo de un error en la creación del modelo

4.2. Funciones

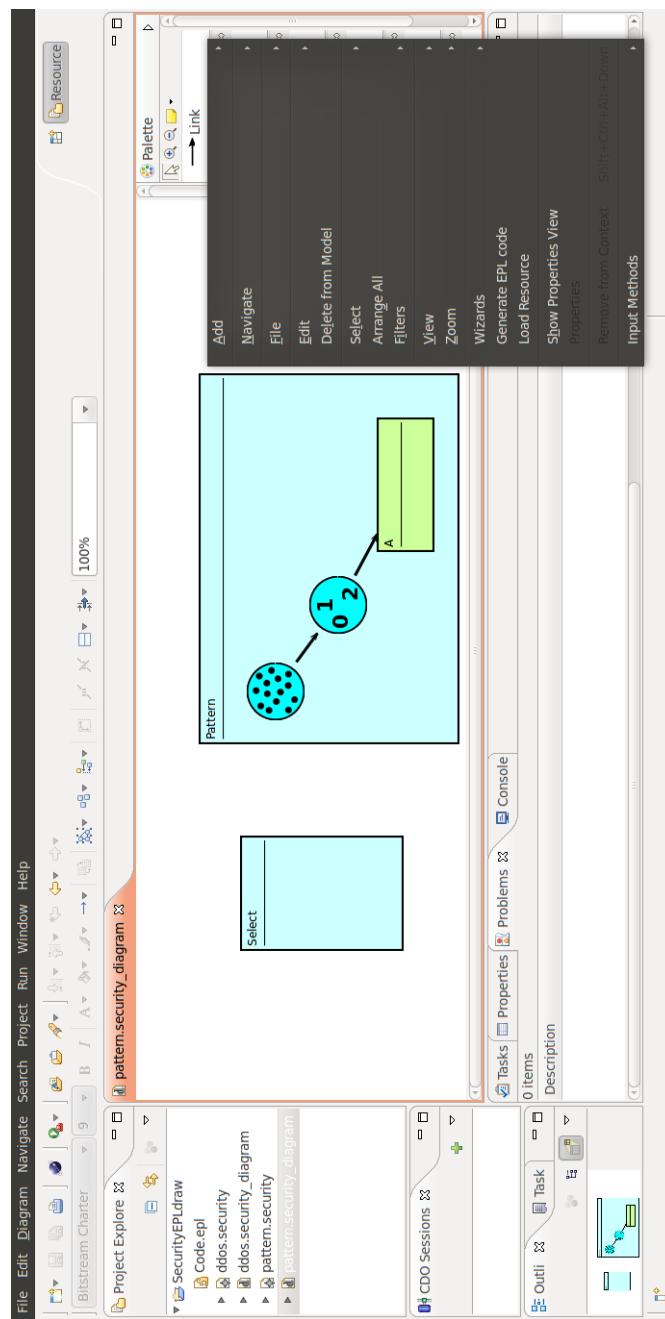


Figura 4.4.: Generación de código EPL a partir del modelo definido con el editor

4. Descripción general del proyecto

4.3. Características del usuario

La aplicación va destinada a usuarios que deseen diseñar patrones de eventos complejos sin tener conocimientos de lenguajes de programación. No obstante, los usuarios tienen que conocer algunos conceptos básicos sobre la seguridad informática, ya que la aplicación está orientada en este ámbito.

A medida que el usuario diseña el patrón, puede ir aprendiendo cómo puede ir diseñando otros patrones, debido a que si se van teniendo errores, la aplicación provee información sobre el error que se comete y los posibles modelos que se pueden ir diseñando una vez subsanado el error.

4.4. Restricciones generales

4.4.1. Control de versiones

En los sistemas de control de versiones se almacenan todas las versiones de los ficheros utilizados en el desarrollo.

Dos de las múltiples ventajas de estos sistemas son que se pueden revertir cambios y que no hay pérdida de información debido a que mientras se va desarrollando la aplicación se van almacenando las versiones de los ficheros. Otra de las grandes ventajas es que se puede desarrollar una aplicación en conjunto con otros desarrolladores, pero este no es el caso en el que se puede aprovechar esa ventaja.

En resumen, es una manera de incrementar la seguridad de nuestra aplicación durante el desarrollo y la agilización debido a la estructuración del almacenamiento del sistema de control de versiones.

El sistema de control de versiones utilizado en la aplicación ha sido Subversion, uno de los sistemas más utilizados en el mundo y fue diseñado para reemplazar a CSV (Concurrent Versions System). Posee gran documentación, por ejemplo en los libros [33] y [48].

Además, para integrar Subversion en Eclipse, se ha utilizado Subclipse [28]. Subclipse es un *plugin* creado especialmente para la integración de Subversion en el entorno Eclipse. Debido a que todo el desarrollo de la aplicación se ha realizado en Eclipse, la utilización de este *plugin* es una manera de agilizar el desarrollo debido a que no hace falta salir del entorno para utilizar el control de versiones (véase Anexo A para los detalles de instalación de Subclipse).

4.4.2. Lenguajes de programación y tecnologías

Los lenguajes de programación y tecnologías utilizadas son las siguientes:

4.4. Restricciones generales

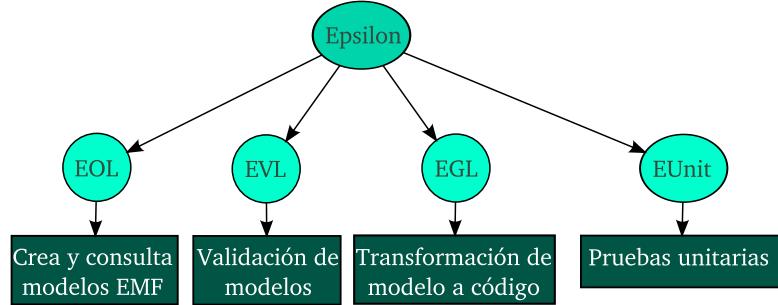


Figura 4.5.: Lenguajes y herramientas de Epsilon utilizados en el PFC

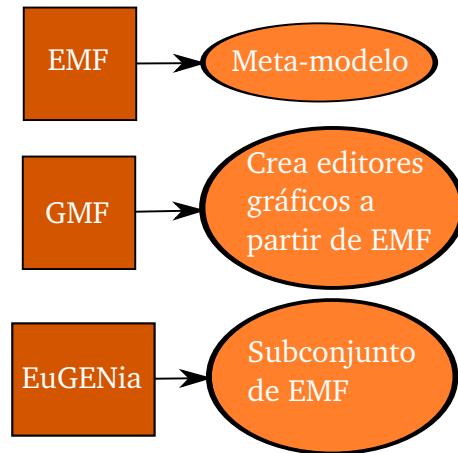


Figura 4.6.: EMF - GMF - EuGENia

- Epsilon: Familia de lenguajes de programación que se usan para interactuar con modelos EMF para realizar tareas de MDE tales como generación de código, transformaciones modelo a modelo, etc [45]. En la Figura 4.5 se muestra con qué lenguajes y herramientas de Epsilon se ha desarrollado el PFC.
- EuGENia: Herramienta que genera los modelos necesarios para crear un editor GMF. En la Figura 4.6 se muestra dónde está ubicada la herramienta EuGENia.
- EOL: Lenguaje de programación imperativo que sirve para crear consultar y modificar los modelos EMF.
- EVL: Lenguaje de validación de modelos similar a OCL pero con algunas funcionalidades añadidas.

4. Descripción general del proyecto

- EGL: Lenguaje adaptado para las transformaciones modelo a código (M2T).
- EUnit: *Framework* de pruebas unitarias para validar las transformaciones.
- Java: Lenguaje de alto nivel orientado a objetos, utilizados en este proyecto para crear el *plugin* necesario para integrar la transformación M2T del modelo a código EPL en el propio editor de la aplicación.

4.4.3. Herramientas

Las herramientas utilizadas han sido:

- Eclipse: Es un IDE multiplataforma desarrollado inicialmente por *IBM* y que actualmente es desarrollado por la Fundación Eclipse. Se distribuye bajo la licencia libre *Eclipse Public License*. La página web de Eclipse es [5].
- Inkscape: Es un editor de gráficos vectoriales de código abierto, utilizado en este proyecto para crear todas las imágenes utilizadas en la personalización del editor. Se ha utilizado tanto para las imágenes de la paleta de la aplicación, como para las imágenes del entorno principal. Toda la información sobre la herramienta puede encontrarse en [25].

4.4.4. Sistemas operativos y hardware

Esta aplicación ha sido desarrollada en el sistema operativo *GNU/Linux Ubuntu 10.04* utilizando *Eclipse Modeling Indigo linux-gtk-x86_64* (más información sobre la versión de Eclipse utilizada en el Capítulo 8). A pesar de haber sido desarrollada en *GNU/Linux*, el desarrollo se podría haber realizado en otros sistemas operativos debido a que Eclipse es un IDE multiplataforma. Además la aplicación puede ser ejecutada igualmente en distintos sistemas operativos.

Para el desarrollo y prueba de la aplicación se recomienda tener al menos 1024 MB de memoria *RAM* debido a los recursos que consume Eclipse.

5. Desarrollo del proyecto

En este capítulo hablaremos sobre el modelo de ciclo de vida utilizado en el PFC, los requisitos necesarios para el correcto funcionamiento del sistema y las fases análisis, diseño, implementación y pruebas.

5.1. Modelo de ciclo de vida

El modelo de ciclo de vida elegido en el proyecto ha sido el modelo lineal secuencial o modelo en cascada.

Este modelo se basa en la idea de empezar una fase una vez que se haya terminado la fase previa [3]. Las fases del modelo son las siguientes [3]:

1. **Análisis y definición de requerimientos.** Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.
2. **Diseño del sistema y del software.** El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.
3. **Implementación y prueba de unidades.** Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.
4. **Integración y prueba del sistema.** Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.
5. **Funcionamiento y mantenimiento.** Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos.

5. Desarrollo del proyecto

En la Figura 5.1 (extraída de [26]) se puede observar un esquema del modelo.

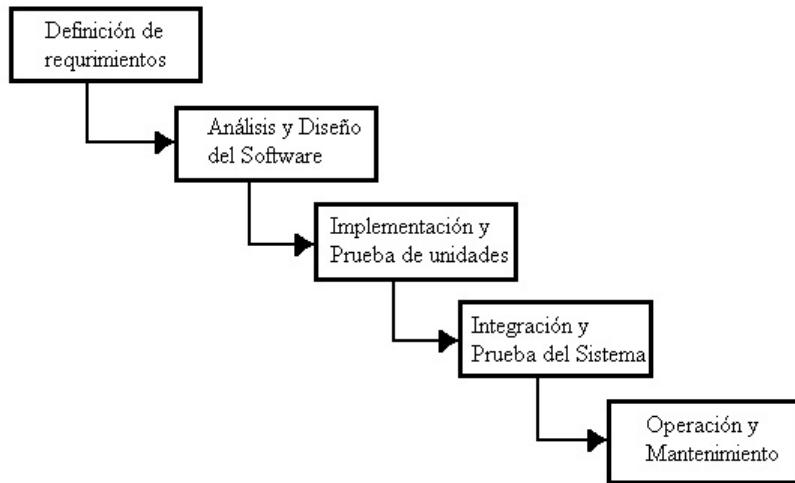


Figura 5.1.: Modelo en cascada

Las ventajas de este modelo son que la documentación se produce en cada fase y que éste cuadra con otros modelos del proceso de ingeniería. Su principal problema es su inflexibilidad al dividir el proyecto en distintas etapas [3].

Por lo tanto, el modelo en cascada sólo se debe utilizar cuando los requerimientos se comprendan bien y sea improbable que cambien radicalmente durante el desarrollo del sistema [3].

Este modelo de ciclo de vida es adecuado en este PFC, debido a que cumple con los requisitos indicados anteriormente para que se pueda llevar a cabo.

5.2. Requisitos

5.2.1. Funcionales

Los requisitos que debe de cumplir la aplicación son los siguientes:

- Definir el patrón gráfico: Permitir al usuario diseñar patrones de eventos complejos mediante el editor GMF que se proporciona.
- Validación de modelos creados: El editor deberá comprobar que los modelos diseñados por el usuario son correctos.
- Transformar a código EPL: Una vez diseñado el modelo del patrón de eventos complejos mediante el editor, deberá existir una función para poder transformar dicho modelo en código EPL.

5.2.2. De información

A continuación se detalla toda la información que deberá ser almacenada por la aplicación para que se puedan diseñar los modelos:

- Un patrón será “representado” por nodos y enlaces.
- Los nodos se deberán de clasificar en cláusulas, patrones y elementos.
- Los elementos se clasificarán en átomos de patrones y operadores.
- Los átomos de patrones se pueden clasificar en observadores, eventos y atributos. En todos los casos se deberá almacenar su nombre. Los observadores almacenarán información sobre tiempo (año, mes, día...), los eventos deberán almacenar un alias, información sobre las condiciones que deben de cumplir y los atributos que relaciona. Los atributos deberán almacenar un alias y el evento que relaciona.
- Los operadores podrán ser clasificados en relacionales, de repetición, lógicos, temporales y guardas, junto a otro tipo de clasificación: unarios, binarios y n-arios. Se deberá almacenar su nombre.
- Los enlaces permitirán unir los distintos nodos y se almacenarán los nodos que relaciona.

5.2.3. Reglas de negocio

La generación de código deberá realizarse siempre que hayan sido validados los modelos satisfactoriamente.

5.2.4. Interfaz

El sistema deberá integrarse por completo con los lenguajes EOL, EVL, EGL y la herramienta EuGENia, para que en conjunto, creen un editor GMF completamente personalizado. Además, el editor deberá proporcionar la generación de código desde el mismo editor, integrando EGL con el editor GMF final.

5.2.5. No funcionales

Los requisitos no funcionales que deberá cumplir la aplicación son los siguientes:

- Portabilidad: La herramienta debe poder ser desarrollada y ejecutada en distintos sistemas operativos.

5. Desarrollo del proyecto

- Usabilidad: Se debe de ofrecer una interfaz gráfica intuitiva, para que el usuario se centre en diseñar patrones y no en entender el funcionamiento de la interfaz gráfica, ni tampoco tiene por qué conocer/comprender el lenguaje EPL.
- Mantenibilidad: La aplicación debe de estar bien documentada y con un código suficientemente claro para que la herramienta pueda ser entendible por otros desarrolladores y así poder ser mejorada en un futuro.
- Fiabilidad: Todo el código generado por los modelos debe de ser correcto, debido a que está pensado para usarse directamente en el motor de Esper.

5.3. Análisis del sistema

5.3.1. Casos de uso

Un caso de uso describe un conjunto de interacciones entre actores externos y el sistema en consideración. Los actores son partes externas del sistema que interactúan con el sistema. Un actor puede ser una clase de usuarios, usuarios que pueden desempeñar un rol u otros sistemas [34]. A continuación se detallan los casos de uso de nuestro sistema. En la Figura 5.2 y en la Figura 5.3 se muestran los diagramas con los casos de uso necesarios en el sistema.

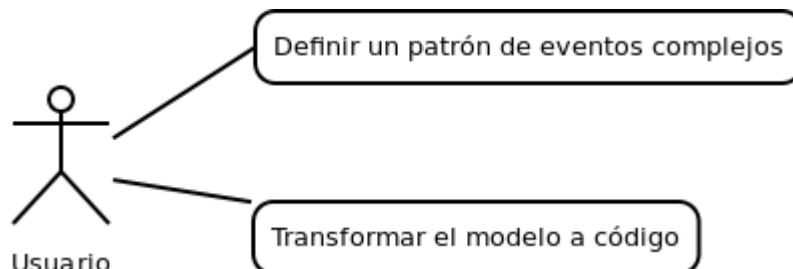


Figura 5.2.: Casos de uso con el Actor Usuario

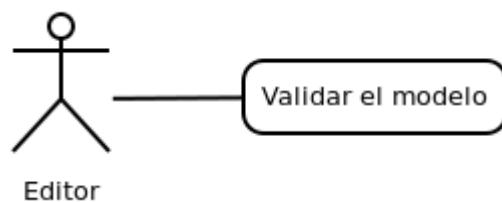


Figura 5.3.: Casos de uso con el Actor Editor

Caso de uso: Definir un patrón de eventos complejos

Descripción El usuario define un patrón de eventos complejos.

Actor El usuario que desea definir un patrón de eventos complejos.

Precondiciones Ninguna.

Postcondiciones Se define un patrón de eventos complejos.

Escenario principal

1. El editor (sistema) muestra un entorno donde se pueden seleccionar los elementos que conforman un patrón de eventos complejos.
2. El usuario (actor) selecciona un elemento.
3. El usuario posiciona el elemento en el editor.
4. El editor comprueba que la posición del elemento es correcta. (Repetir pasos 2-4 hasta que el usuario deje de seleccionar elementos).
5. El patrón de eventos complejos se define con éxito.

Caso de uso: Validar el modelo

Descripción El editor (sistema) valida un modelo definido por el usuario.

Actor El editor.

Precondiciones Debe de existir un patrón de eventos complejos definido.

Postcondiciones Se valida el modelo definido por el usuario.

Escenario principal

1. El editor (sistema) comprueba que todos los elementos son correctos.
2. El editor valida el modelo con éxito.

Escenario alternativo

1a. Algún elemento no es correcto:

1. El editor muestra un mensaje indicando al usuario el tipo de error producido.
2. El usuario soluciona el error.

5. Desarrollo del proyecto

Caso de uso: Transformar el modelo a código

Descripción El editor (sistema) transforma un modelo definido por el usuario a código.

Actor El editor.

Precondiciones Debe existir un modelo correctamente validado.

Postcondiciones Se genera el código EPL a partir del modelo definido.

Escenario principal

1. El editor provee un botón para que el usuario pueda generar el código EPL a partir del modelo definido.
2. El usuario pulsa el botón que le proporciona el editor para generar el código.
3. El editor genera satisfactoriamente el código EPL.

5.3.2. Metamodelo

Para realizar el metamodelo, se ha utilizado el metamodelo Ecore, que como ya hemos comentado anteriormente, es muy similar a EMOF. Para conseguir el metamodelo Ecore, se ha utilizado Emfatic. El metamodelo completo se puede visualizar en la Figura 5.4. Este metamodelo define la sintaxis abstracta de EPL.

Root es el elemento raíz del metamodelo. Tiene cuatro atributos: `clauses`, un contenedor de `0..*` Clause (cláusula); `pattern`, `0 ó 1` Pattern (patrón); `links`, un contenedor de `0..*` Link (enlace); y `nodes`, un contenedor de `0..*` Node (nodo). A nivel conceptual deberíamos de tener `1` Pattern, pero debido a detalles de implementación utilizamos `0..1` Pattern (véase Figura 5.5).

5.3. Análisis del sistema

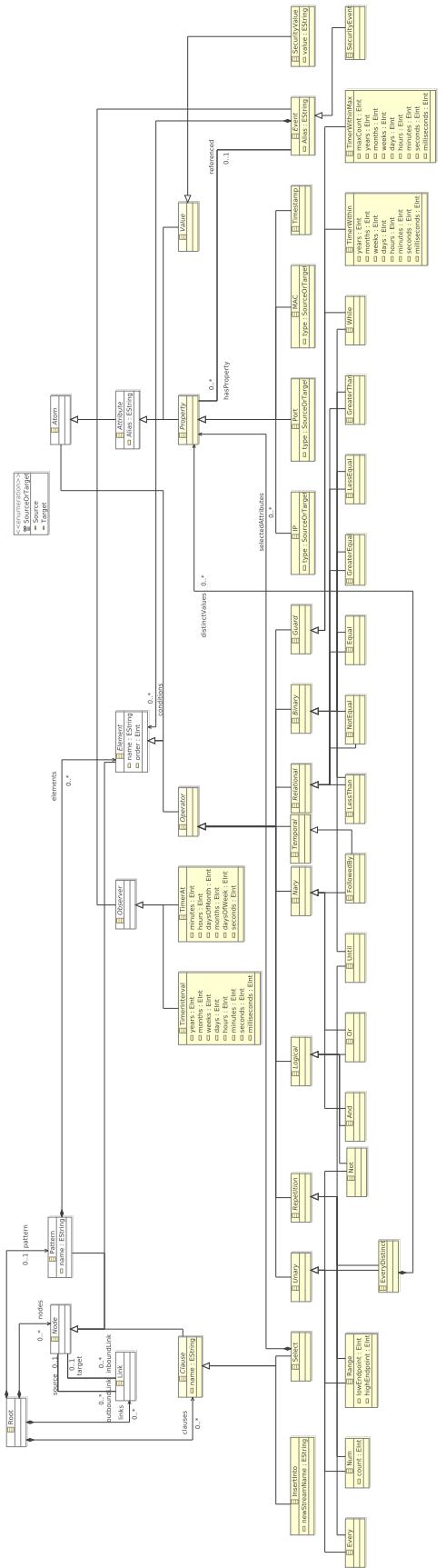


Figura 5.4: Metamodelo

5. Desarrollo del proyecto

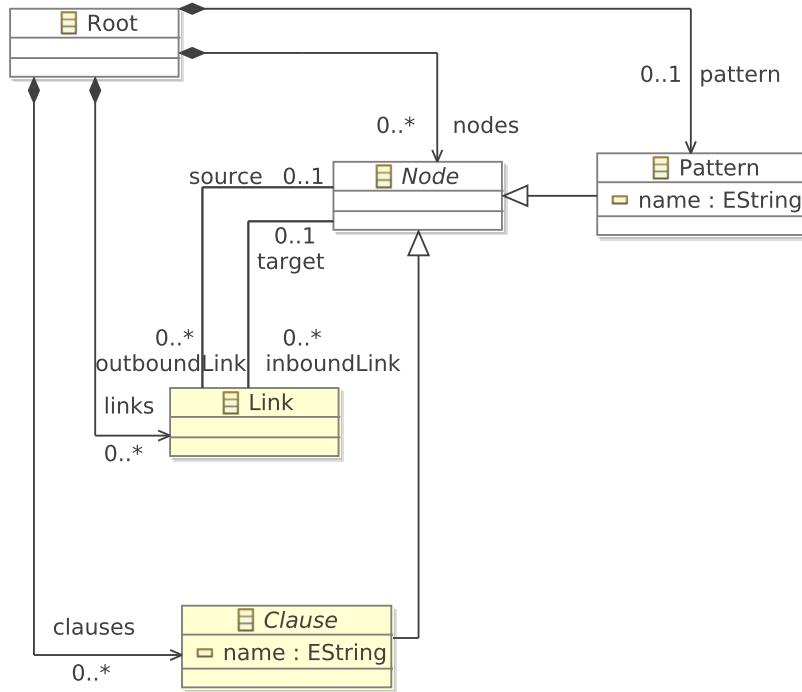


Figura 5.5.: Root

Node son todos los elementos del editor excepto Link. Contiene dos atributos: **outboundLink**, una referencia a un contenedor de 0..* Link que son los Link que van dirigidos a los Node que apuntan e **inboundLink**, una referencia a un contenedor de 0..* Link que son los Link que apuntan al Node en cuestión (véase Figura 5.6).

5.3. Análisis del sistema

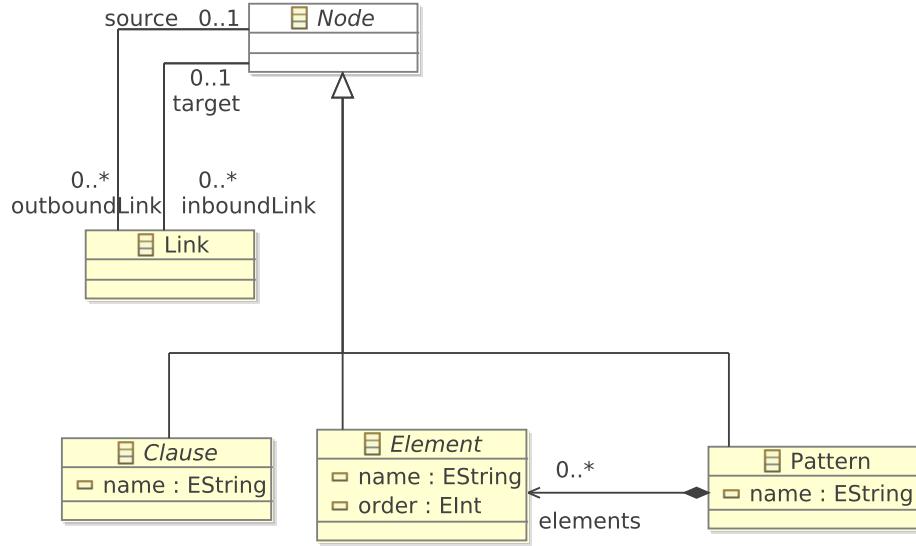


Figura 5.6.: Node

Link son los enlaces que unen los distintos Node del editor. Sus atributos son **source** y **target**, que quiere decir que contiene el Node que le apunta y el Node al que apunta (véase Figura 5.7).

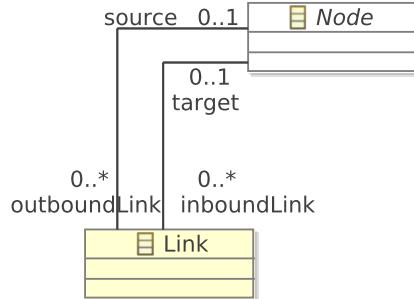


Figura 5.7.: Link

Clause (cláusula) es la superclase de las cláusulas Select e InsertInto y hereda de Node. Su atributo es **name** (el nombre) e InsertInto tiene un atributo llamado **newStreamName** que es el nombre del flujo de eventos (véase Figura 5.8).

5. Desarrollo del proyecto

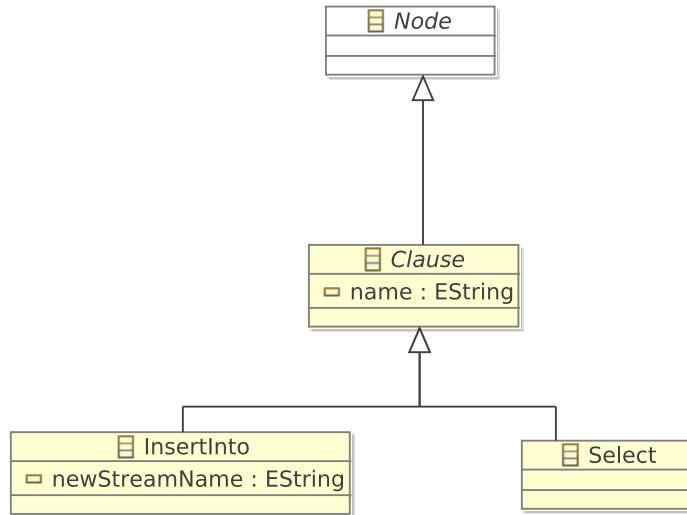


Figura 5.8.: Clause

Pattern es toda la lógica del patrón que se desea diseñar. Es una subclase de Node, tiene un atributo **name** y tiene una composición de Element (elementos) denominado **elements** que son los encargados de dar la lógica al patrón que se diseña. (véase Figura 5.9).

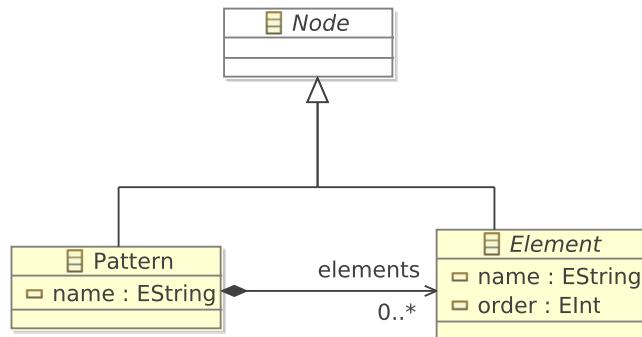


Figura 5.9.: Pattern

Element hereda de Node y es la superclase de Operator (operadores) y Atom (átomos de patrones).

Los atributos de Element son **name** y **order**. El atributo **order** es un atributo que sirve para indicar la posición del Element cuando es apuntado por un operador. Si el operador que le apunta es Unary (unario), **order** debe de ser siempre 1. Si el operador es Binary (binario), el atributo **order** podrá ser 1 (si se encuentra a la

5.3. Análisis del sistema

izquierda del operador) ó 2 (si se encuentra a la derecha del operador). Si el operador es Nary (n-ario) el atributo **order** podrá tener el valor de 1 a N (véase Figura 5.10).

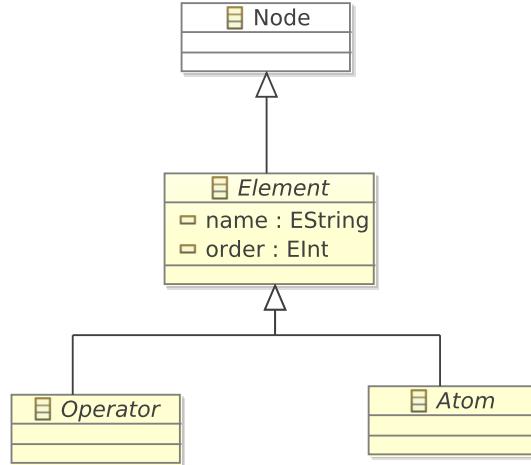


Figura 5.10.: Element

Event (eventos) son los eventos del sistema. Hereda de Atom, su atributo **Alias** es el alias del Event y además tiene una relación de 0..* de Property (propiedades) que son los Property a los que referencia. Es la superclase de SecurityEvent (véase Figura 5.11).

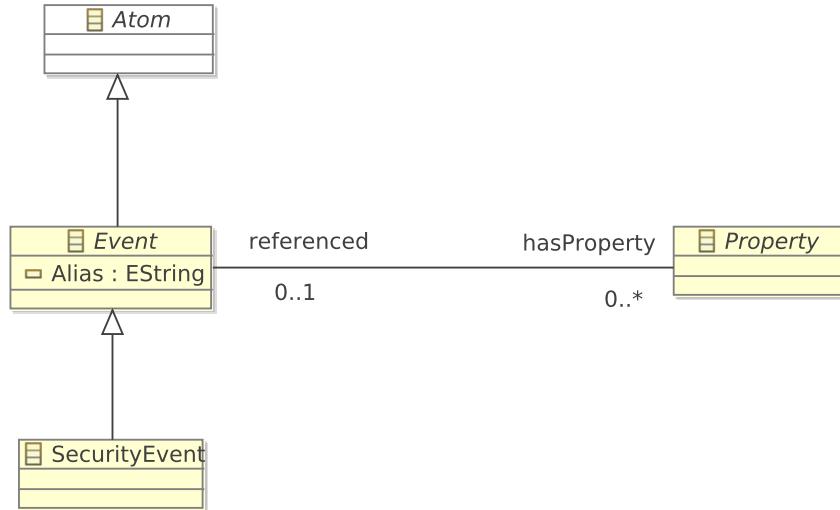


Figura 5.11.: Event

5. Desarrollo del proyecto

Attribute (atributos) son los atributos que pueden tener los eventos. Attribute hereda de Atom al igual que Event y las subclases de Attribute son Value (valor), que es el valor de un Attribute; y Property, que son las propiedades de los eventos y que pueden ser referenciados o no con un Event (relación 0..1). SecurityValue hereda de Value y tiene el atributo `value` (su valor) (véase Figura 5.12).

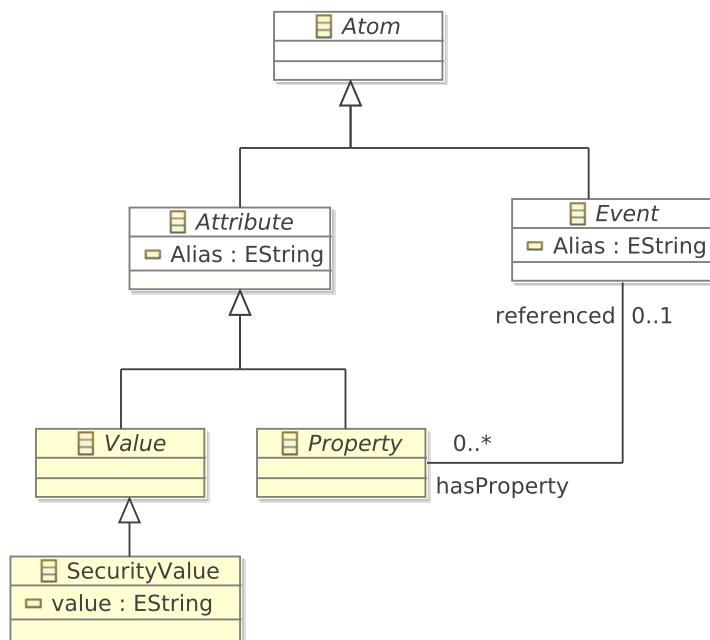


Figura 5.12.: Attribute

Las subclases de Property son IP, Port, MAC y Timestamp. Todas estas subclases (excepto Timestamp) tienen un atributo enumerado llamado `type` y que puede valer `source` o `target`. Esto quiere decir que pueden ser origen o destino (e.g IP origen o IP destino) (véase Figura 5.13).

5.3. Análisis del sistema

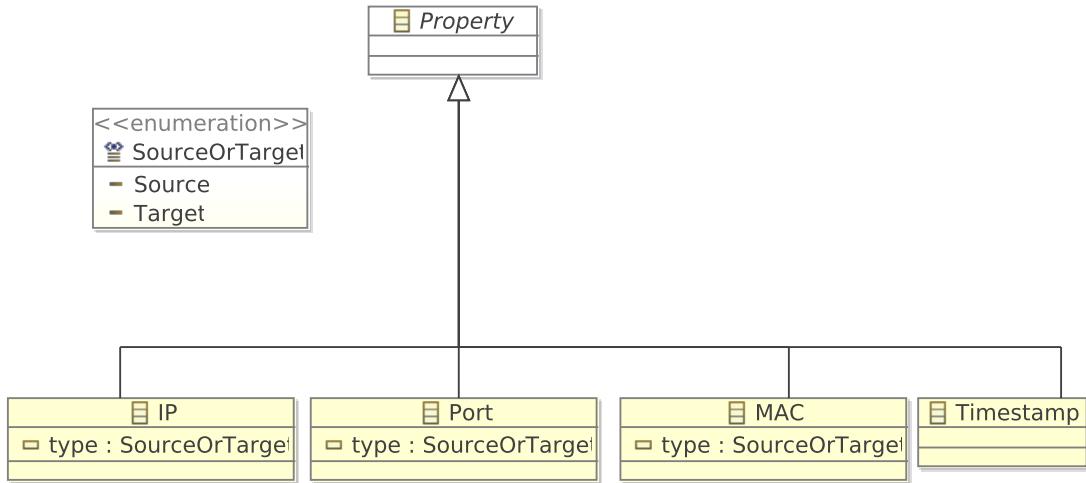


Figura 5.13.: Property

Las subclases de Operator se dividen en dos clasificaciones. La primera clasificación son los operadores Unary, Binary y Nary, y la otra clasificación son los operadores Logical (lógicos), Repetition (de repetición), Temporal (temporales), Relational (relacionales) y Guard (guardas) (véase Figura 5.14).

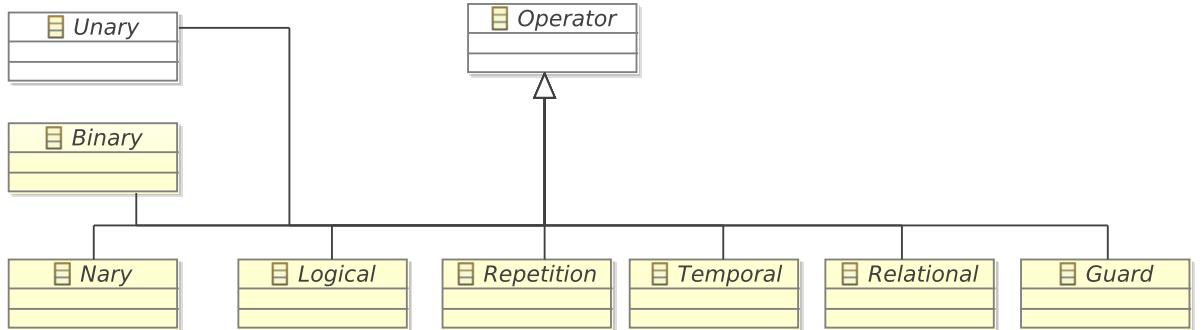


Figura 5.14.: Operator

Los operadores Repetition se dividen en las siguientes subclases: Every, EveryDistinct, Range, Until y Num. Every, Range, EveryDistinct y Num son operadores Unary mientras que Until es un operador Binary. EveryDistinct está compuesto de 0..* Property, que son los Property de los que no interesan resultados duplicados. Range tiene los atributos lowEndpoint y highEndpoint que son el valor máximo y el valor mínimo del rango (véase Figura 5.15).

5. Desarrollo del proyecto

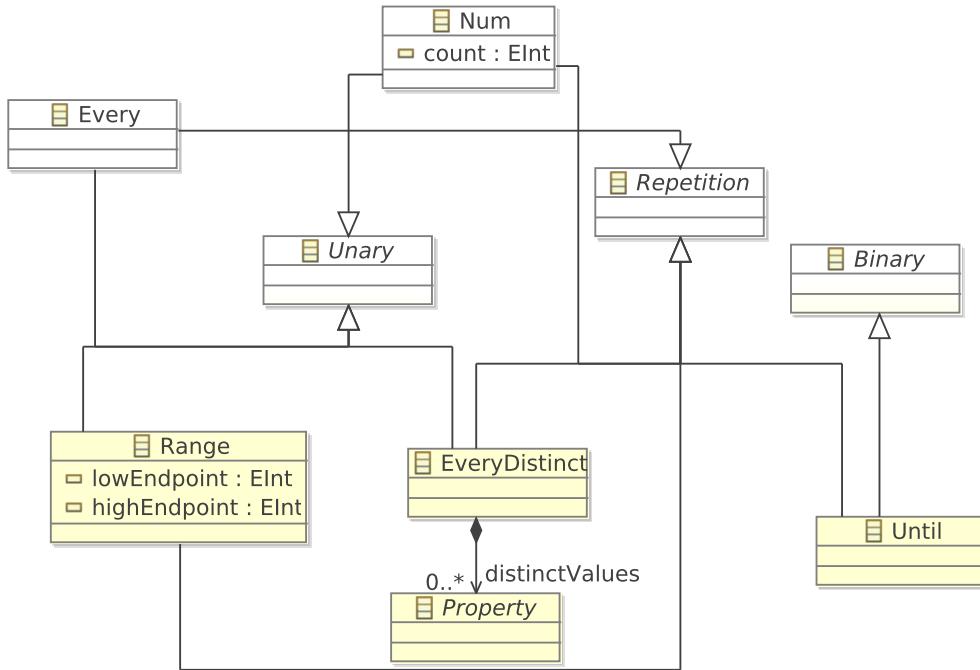


Figura 5.15.: Repetition

Los operadores Logical son And, Or, y Not, sobradamente conocidos en el mundo de la lógica y la programación (véase Figura 5.16).

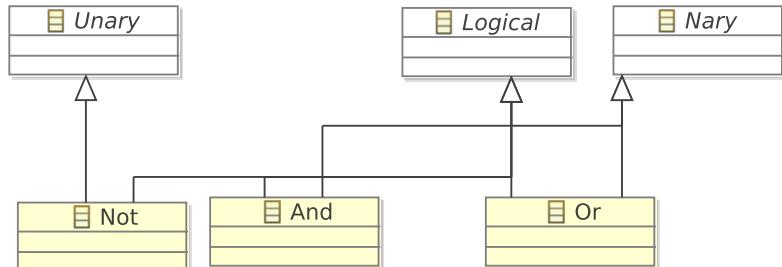


Figura 5.16.: Logical

El operador FollowedBy es un operador Nary y Temporal, y es el que se encarga de evaluar un evento por su derecha cuando el evento de la expresión de su izquierda devuelve true (véase Figura 5.17).

5.3. Análisis del sistema

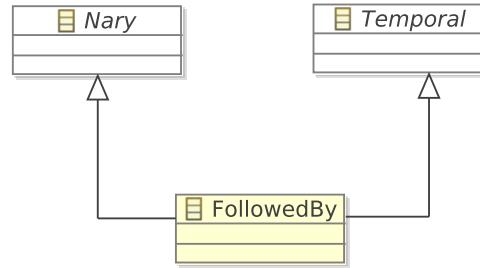


Figura 5.17.: FollowedBy

Guard son las condiciones where que controlan el ciclo de vida de las subexpresiones. Tiene las siguientes subclases: TimerWithin, TimerWithinMax y While. TimerWithin y TimerWithinMax tienen los siguientes atributos comunes: **years**, **months**, **weeks**, **days**, **hours**, **minutes**, **seconds**, **milliseconds** (información sobre tiempo). Además, TimerWithinMax tiene un atributo **maxCount**. While es un operador Binary (véase Figura 5.18).

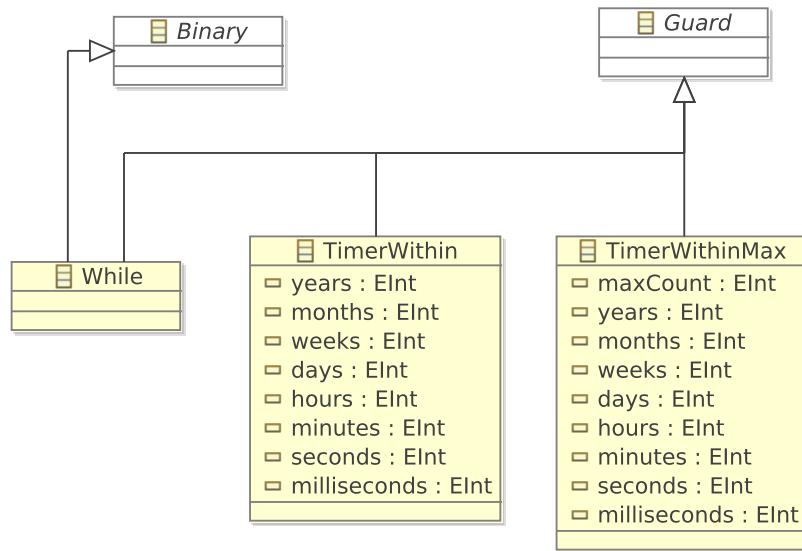


Figura 5.18.: Guard

Los Observer observan los eventos basados en tiempo. Son los siguientes: **TimeInterval** y **TimerAt**. Los atributos de **TimeInterval** son: **years**, **months**, **weeks**, **days**, **hours**, **minutes**, **seconds**, **milliseconds** y los atributos de **TimerAt** son: **minutes**, **hours**, **daysOfMonth**, **months**, **daysOfWeek**, **seconds** (véase Figura 5.19).

5. Desarrollo del proyecto

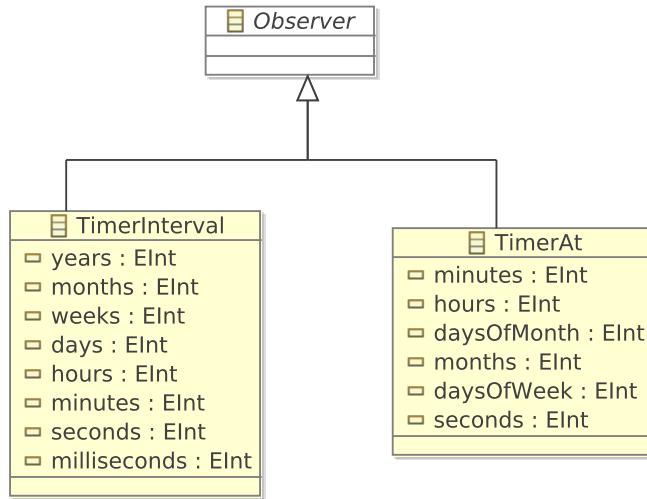


Figura 5.19.: Observer

Los operadores Relational son los siguientes: Equal, NotEqual, LessThan, GreaterThan, LessEqual, GreaterEqual ($=, !=, <, >, \leq, \geq$). Todos estos operadores son Binary (véase Figura 5.20).

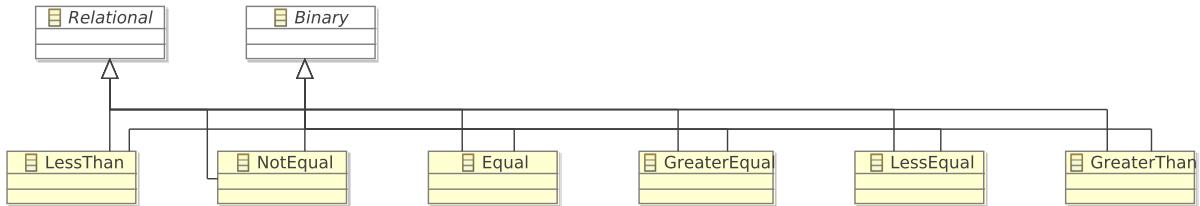


Figura 5.20.: Relational

5.4. Diseño del sistema

La aplicación diseñada para cumplir todos los objetivos del PFC se denomina SecurityEPLdraw. A continuación se detalla su funcionamiento.

5.4.1. SecurityEPLdraw

La aplicación desarrollada es capaz de realizar:

- Diseño de modelos de patrones complejos: El usuario es el que se encarga de diseñar un patrón.

- Validación de los modelos diseñados: Se valida el patrón diseñado por el usuario.
- Transformación a código EPL: El modelo es transformado a código EPL.

5.4.2. Estructura

Como se observa en la Figura 5.21, una vez definido el metamodelo, se utiliza EuGENia para crear un editor gráfico GMF que permite poder diseñar todos los modelos posibles a partir del metamodelo. A continuación, cuando el usuario crea un modelo, se valida mediante EVL. Para finalizar, para transformar el modelo a código EPL se utiliza EGL. Para validar las transformaciones a código EPL se ha utilizado EUnit.

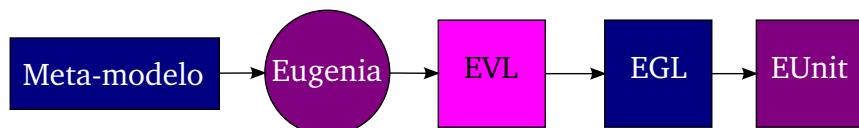


Figura 5.21.: Estructura de la aplicación

5.5. Implementación

5.5.1. Metamodelo

El metamodelo completo puede visualizarse en la Figura 5.4 y la explicación de cada una de sus partes está detallada en la Sección 5.3.2. El código fuente del metamodelo completo se encuentra en el Anexo B.1.

5.5.2. Validación de modelos

Para que los modelos diseñados sean correctos, tienen que pasar un proceso de validación. Como ya se ha comentado anteriormente, esta validación se ha conseguido gracias al lenguaje EVL, que provee las herramientas necesarias para la validación de modelos. El fichero EVL encargado de la validación se denomina *security.evl*. Las restricciones que se han definido para la validación de los modelos han sido las siguientes:

- **Restricciones generales.** Estas restricciones afectan a todos o casi todos los elementos del editor o es necesario mirar el resto de elementos para conseguir definir la restricción. Las restricciones son las siguientes:

5. Desarrollo del proyecto

- Origen y destino de Link deben ser diferentes. El código en EVL se puede visualizar en el Listado B.20.
 - Como máximo un nodo puede ser apuntado por un Link. El código en EVL se puede visualizar en el Listado B.21.
 - Nodos permitidos fuera de Pattern. El código en EVL se puede visualizar en el Listado B.22.
 - Timestamp sólo puede estar dentro de Select. El código en EVL se puede visualizar en el Listado B.23.
 - Al menos debe de haber un Pattern en el editor, no podemos trabajar con un editor vacío. El código en EVL se puede visualizar en el Listado B.24.
- **Restricciones que afectan a Pattern.** Son todas las restricciones que se tienen que cumplir en los Pattern:
- Cada Pattern debe de tener un nombre. El código en EVL se puede visualizar en el Listado B.25.
 - Sólo un Node dentro de Pattern puede estar sin ningún Node que le apunte, es decir, que en el árbol que se forma dentro de Pattern para diseñar el patrón, sólo puede haber un nodo raíz. El código en EVL se puede visualizar en el Listado B.26.
- **Restricciones que afectan a Select.** Son todas las restricciones que se tienen que cumplir en los Select:
- Un Select debe ser definido en el editor. El código en EVL se puede visualizar en el Listado B.27.
 - Los Attribute dentro de Select deben tener distintos alias. El código en EVL se puede visualizar en el Listado B.28.
 - Sólo un Select está permitido en el editor. El código en EVL se puede visualizar en el Listado B.29.
- **Restricciones que afectan a Property.** Son todas las restricciones que se tienen que cumplir en los Property:
- Los Property que estén fuera de Select o EveryDistinct tienen que tener un Link que le apunte. El código en EVL se puede visualizar en el Listado B.30.
 - Sólo un Link que le apunte está permitido como máximo. El código en EVL se puede visualizar en el Listado B.31.

5.5. Implementación

- Comprobación de Node que apuntan a Property. El código en EVL se puede visualizar en el Listado B.32.
 - Comprobación del Event referenciado por Property. El código en EVL se puede visualizar en el Listado B.33.
 - Property con order = 2 debe de tener un Event referenciado. El código en EVL se puede visualizar en el Listado B.34.
- **Restricciones que afectan a SecurityValue.** Son todas las restricciones que se tienen que cumplir en los SecurityValue:
- El atributo Value tiene que tener un valor. El código en EVL se puede visualizar en el Listado B.35.
 - Tiene que tener el atributo order = 2. El código en EVL se puede visualizar en el Listado B.36.
- **Restricciones que afectan a Binary.** Son todas las restricciones que se tienen que cumplir en los operadores Binary:
- Los operadores Binary tienen que apuntar a 2 Node destino. El código en EVL se puede visualizar en el Listado B.37.
 - Los Node a los que apuntan los operadores Binary tienen que tener los atributos order = 1 y order = 2. El código en EVL se puede visualizar en el Listado B.38.
- **Restricciones que afectan a Nary.** Son todas las restricciones que se tienen que cumplir en los operadores Nary:
- Los Node a los que apuntan los operadores Nary tienen que tener los atributos order = 1 ... order = N. El código en EVL se puede visualizar en el Listado B.39.
 - Los operadores Nary tienen que apuntar al menos 2 Node destino. El código en EVL se puede visualizar en el Listado B.40.
- **Restricciones que afectan a los operadores Unary.** Son todas las restricciones que se tienen que cumplir en los operadores Nary:
- Los operadores Unary tienen que apuntar a 1 Node destino. El código en EVL se puede visualizar en el Listado B.41.
- **Restricciones que afectan a los operadores Relational.** Son todas las restricciones que se tienen que cumplir en los operadores Relational:

5. Desarrollo del proyecto

- Los operadores Relational tienen que apuntar a Attribute, no Element. El código en EVL se puede visualizar en el Listado B.42.
 - Sólo se puede apuntar como máximo a un SecurityValue. El código en EVL se puede visualizar en el Listado B.43.
 - Los Node a los que apuntan tienen que ser del mismo tipo o con un SecurityValue. El código en EVL se puede visualizar en el Listado B.44.
 - Los operadores Relational tienen que estar dentro de SecurityEvent si no le apunta un While. El código en EVL se puede visualizar en el Listado B.45.
- **Restricciones que afectan a InsertInto.** Son todas las restricciones que se tienen que cumplir en InsertInto:
- Su atributo NewStreamName tiene que tener un valor. El código en EVL se puede visualizar en el Listado B.46.
 - Sólo un InsertInto está permitido en el editor. El código en EVL se puede visualizar en el Listado B.47.
- **Restricciones que afectan a operadores Logical.** Son todas las restricciones que se tienen que cumplir en Logical:
- Si es un operador And o un operador Or y está fuera de un SecurityEvent, tienen que apuntar a 2 Node. El código en EVL se puede visualizar en el Listado B.48.
 - Los Node a los que apuntan And y Or fuera de SecurityEvent tienen que ser FollowedBy, SecurityEvent, operadores Logical o TimerInterval. El código en EVL se puede visualizar en el Listado B.49.
 - Los Node a los que apuntan And y Or dentro de SecurityEvent tienen que ser operadores Logical u operadores Relational. El código en EVL se puede visualizar en el Listado B.50.
 - Los Node a los que apunta Not fuera de SecurityEvent tienen que ser FollowedBy, SecurityEvent, operadores Logical o TimerInterval. El código en EVL se puede visualizar en el Listado B.51.
 - Los Node a los que apunta Not dentro de SecurityEvent tienen que ser FollowedBy, SecurityEvent, operadores Logical o TimerInterval. El código en EVL se puede visualizar en el Listado B.52.
- **Restricciones que afectan a SecurityEvent.** Son todas las restricciones que se tienen que cumplir en SecurityEvent:

5.5. Implementación

- Los elementos permitidos dentro de SecurityEvent son operadores Relational, operadores Logical y Attribute. El código en EVL se puede visualizar en el Listado B.53.
 - El atributo name tiene que tener un valor. El código en EVL se puede visualizar en el Listado B.54.
 - Sólo un Node sin que le apunte ningún nodo está permitido dentro de SecurityEvent (único elemento raíz). El código en EVL se puede visualizar en el Listado B.55.
- **Restricciones que afectan a TimerInterval.** Son todas las restricciones que se tienen que cumplir en TimerInterval:
- Los Node que le apuntan tienen que ser o Every u operadores Logical. El código en EVL se puede visualizar en el Listado B.56.
 - Sólo un atributo tiene que ser mayor que 0. El código en EVL se puede visualizar en el Listado B.57.
- **Restricciones que afectan a TimerAt.** Son todas las restricciones que se tienen que cumplir en TimerAt:
- Los Node que le apuntan tienen que ser Every. El código en EVL se puede visualizar en el Listado B.58.
- **Restricciones que afectan a TimerWithin.** Son todas las restricciones que se tienen que cumplir en TimerWithin:
- Debe de estar fuera de Pattern. El código en EVL se puede visualizar en el Listado B.59.
 - Sólo un atributo tiene que ser mayor que 0. El código en EVL se puede visualizar en el Listado B.60.
- **Restricciones que afectan a TimerWithinMax.** Son todas las restricciones que se tienen que cumplir en TimerWithinMax:
- Debe de estar fuera de Pattern. El código en EVL se puede visualizar en el Listado B.61.
 - Sólo un atributo tiene que ser mayor que 0. El código en EVL se puede visualizar en el Listado B.62.
 - El atributo maxCount tiene que ser mayor que 0. El código en EVL se puede visualizar en el Listado B.63.
 - Sólo un TimerWithin o un TimerWithinMax está permitido en el editor. El código en EVL se puede visualizar en el Listado B.64.

5. Desarrollo del proyecto

- **Restricciones que afectan a FollowedBy.** Son todas las restricciones que se tienen que cumplir en FollowedBy:
 - Los Node a los que apunta FollowedBy tienen que ser SecurityEvent, Every, EveryDistinct, Num o operadores Logical. El código en EVL se puede visualizar en el Listado B.65.
- **Restricciones que afectan a Every.** Son todas las restricciones que se tienen que cumplir en Every:
 - Los Node a los que apunta Every tienen que ser SecurityEvent, FollowedBy, TimerInterval, TimerAt o Num. El código en EVL se puede visualizar en el Listado B.66.
- **Restricciones que afectan a EveryDistinct.** Son todas las restricciones que se tienen que cumplir en EveryDistinct:
 - Los Node a los que apunta EveryDistinct tienen que ser SecurityEvent, FollowedBy, TimerInterval, TimerAt o Num. El código en EVL se puede visualizar en el Listado B.67.
 - Al menos debe contener a un Attribute. El código en EVL se puede visualizar en el Listado B.68.
- **Restricciones que afectan a Num.** Son todas las restricciones que se tienen que cumplir en Num:
 - Los Node a los que apunta Num tienen que ser SecurityEvent, FollowedBy, And u Or. El código en EVL se puede visualizar en el Listado B.69.
- **Restricciones que afectan a Range.** Son todas las restricciones que se tienen que cumplir en Range:
 - Los Node a los que apunta Range tienen que ser SecurityEvent, FollowedBy, And u Or. El código en EVL se puede visualizar en el Listado B.70.
 - El Node que apunta a Range tiene que ser Until. El código en EVL se puede visualizar en el Listado B.71.
 - Los atributos lowEndPoint y/o highEndPoint tienen que ser mayores que 0. El código en EVL se puede visualizar en el Listado B.72.
 - El atributo highEndpoint tiene que ser mayor o igual que lowEndpoint. El código en EVL se puede visualizar en el Listado B.73.

- **Restricciones que afectan a While.** Son todas las restricciones que se tienen que cumplir en While:

- Los Node a los que apunta While en la expresión de la izquierda tienen que ser SecurityEvent, Every o EveryDistinct. El código en EVL se puede visualizar en el Listado B.74.
- Los Node a los que apunta While en la expresión de la derecha tienen que ser operadores Relational. El código en EVL se puede visualizar en el Listado B.75.

- **Restricciones que afectan a Until.** Son todas las restricciones que se tienen que cumplir en Until:

- Los Node a los que apunta Until en la expresión de la izquierda tienen que ser Range. El código en EVL se puede visualizar en el Listado B.76.
- Los Node a los que apunta Until en la expresión de la derecha tienen que ser Event, And, Or o TimerInterval. El código en EVL se puede visualizar en el Listado B.77.

5.5.3. Transformación Model-to-Text

Fichero EGL

Para realizar la transformación M2T, se ha utilizado EGL. El fichero completo se puede visualizar en el Listado B.78.

A continuación se explica de forma general el contenido del Listado B.78:

- Las líneas 1-8 especifican la parte del comienzo de un fichero EPL, indicando nombre de Pattern e InsertInto en el caso de que exista.
- Las líneas de 9-55 extraen toda la información relativa al Select del modelo.
- En la línea 56 se especifica todo lo que hay dentro de Pattern.
- En las líneas 57-60, extraemos la información del modelo sobre TimerWithin y TimerWithinMax.
- Las líneas 61-419 se han utilizado para la definición de operaciones que ayudan a modularizar la funcionalidad completa de la transformación.
- La mayor dificultad en todo el desarrollo de la transformación M2T se ha concentrado en la operación `traversal` (líneas 122-304). Esta operación consiste en hacer un recorrido en inorden por los árboles de expresión que se forman dentro de Pattern cuando diseñamos los patrones.

5. Desarrollo del proyecto

Plug-in de integración de la transformación M2T

Se ha desarrollado un *plugin* que integra la transformación M2T en el propio editor gráfico. Para conseguir desarrollar el *plugin* se siguieron los siguientes pasos:

- Creación del proyecto:

- Creamos un *Plug-in project*.
- *File / New / Plug-in project*

- Asignación de nombres:

- Asignamos un nombre al proyecto. En nuestro caso *org.eclipse.epsilon.eugenia.examples.security.m2t*
- En las propiedades del proyecto asignar nombre al *plugin*.
- Desmarcar la casilla de generar un activador.
- Desmarcar la casilla de creación del *plugin* a partir de plantillas.

- Identificación de *plugin* y clase para el botón que realizará la acción:

- Identificamos con las teclas *Alt+Shift+F1* en el editor, los campos *The contributing plug-in* (en nuestro caso era: *org.eclipse.epsilon.eugenia.examples.security.diagram*) y el campo de *The type of the selected element* (en nuestro caso *RootEditPart*). En la Figura 5.22 se puede visualizar lo explicado anteriormente.

5.5. Implementación

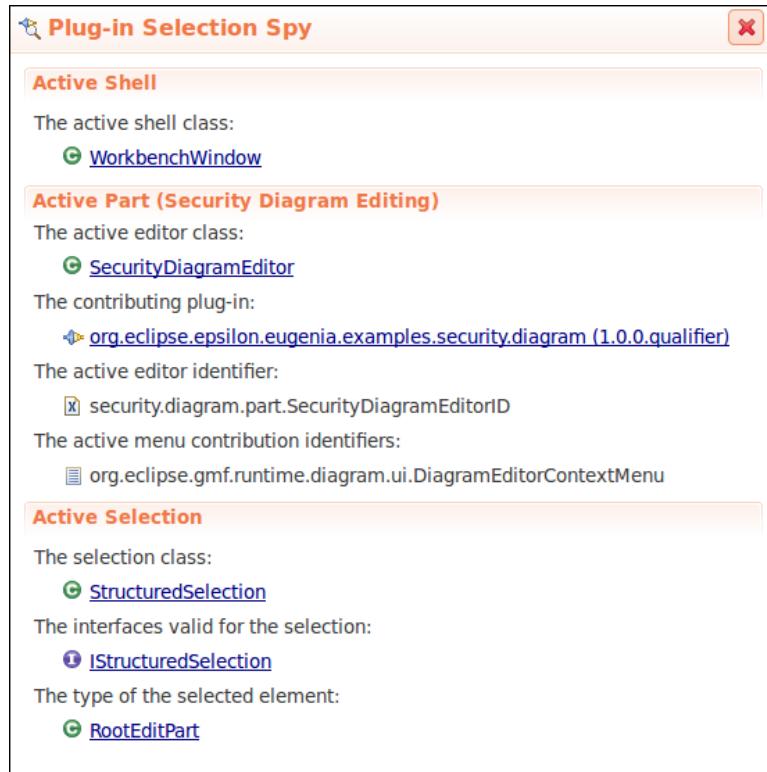


Figura 5.22.: *Plugin* y clase asociada

- Observamos en el *Package Explorer* que la clase *RootEditParts.java* se encuentra en el proyecto: *org.eclipse.epsilon.eugenia.examples.security.diagram*. En la Figura 5.23 se muestra este paso.

5. Desarrollo del proyecto

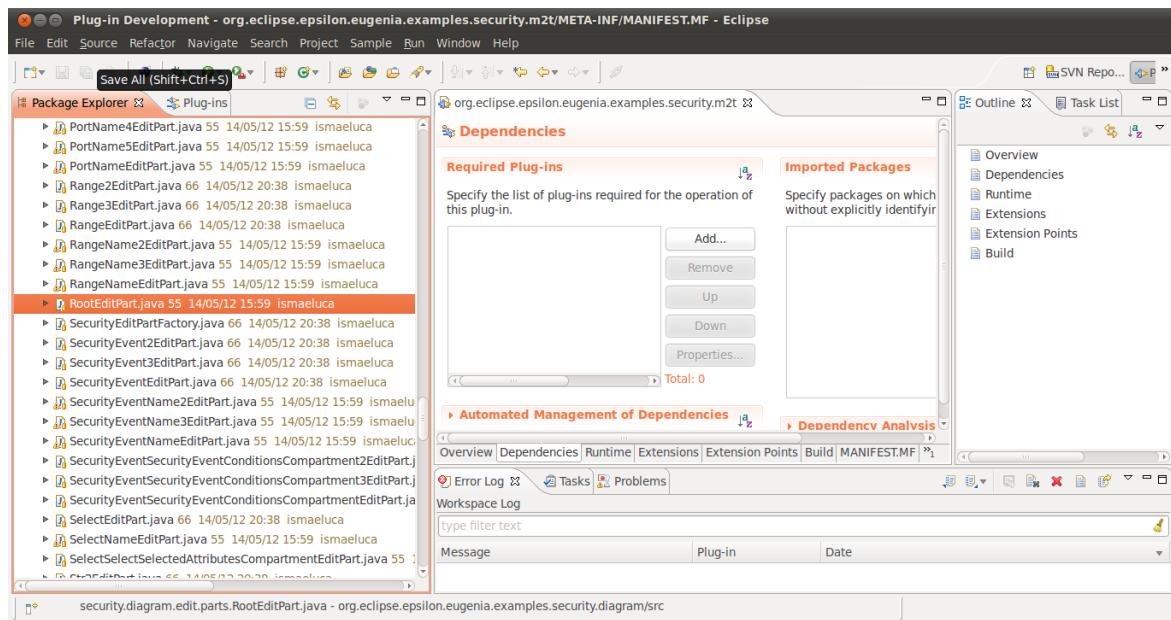


Figura 5.23.: Package Explorer

■ Añadimos Dependencias:

- Las dependencias se añaden en el archivo *MANIFEST.MF* en la pestaña *Dependencies* pulsando sobre el botón *Add...*
- Añadimos:
 - *org.eclipse.epsilon.eugenia.examples.security.diagram*
 - *org.eclipse.ui*

■ Añadimos Extensiones:

- En la pestaña *Extensions* pulsamos sobre el botón *Add...*
- Añadimos: *org.eclipse.ui.popupMenus*

■ Creamos un *objectContribution*:

- Pulsamos botón derecho sobre *org.eclipse.ui.popupMenus / New / objectContribution*
- A la derecha aparecen unos campos (configuración de *objectContribution*)
- Rellenamos *objectClass* con: *security.diagram.edit.parts.RootEditPart*

■ Creamos un *action*:

- Pulsamos *botón derecho / New / action*
 - A la derecha aparecen unos campos (configuración de *action*)
 - Rellenamos label con la etiqueta que queramos que aparezca en el botón del editor (en nuestro caso *Generate EPL code*)
 - En style elegimos la opción *push*.
- Aparición del botón:
- Al ejecutar nuestro editor ya aparece al pulsar el botón derecho del ratón la opción de *Generate EPL code*.
 - Por ahora no hace nada, hay que añadirle funcionalidad.
- Se añaden los ficheros necesarios para añadir funcionalidad:
- Creamos un paquete en src que se denomine *org.eclipse.epsilon.eugenia.examples.security.m2t*
 - Añadimos el fichero *EglTransformModelToEPL.java* (véase Listado B.79).
 - Explico de manera general el Listado B.79:
 - El método *execute()* (líneas 42-54) es el encargado de añadirle funcionalidad al botón creado previamente en el *plugin*.
 - El método *createEmfModel()* (líneas 56-66) se encarga de crear un modelo EMF.
 - El método *getFile()* (líneas 69-76) extrae el fichero .egl de manera local y lo convierte en un objeto en Java.
 - Añadimos una plantilla EGL, en nuestro caso *eglExample.egl*.
 - Comprobar que en la pestaña *Build* se encuentra la carpeta *src/* que es la que contiene nuestro ejemplo en EGL.
- Rellenamos el campo *class*:
- Nos vamos de nuevo a la pestaña *Extensions* y rellenamos el campo *class*.
 - Insertamos la ruta de nuestro fichero *EglTransformModelToEPL.java*
 - Ruta:
org.eclipse.epsilon.eugenia.examples.security.m2t.EglTransformModelToEPL
- Importamos paquetes:
- Importamos los paquetes necesarios para que el fichero *EglTransformModelToEPL.java* funcione.

5. Desarrollo del proyecto

- Añadimos el resto de dependencias:
 - Añadir el resto de dependencias necesarias para el proyecto.
 - Recordar que se añaden en el archivo *MANIFEST.MF* del proyecto *org.eclipse.eugenia.examples.security.m2t* en la pestaña *Dependencies*.
 - Muestro una captura de todos los *Required Plug-ins* e *Imported Packages* necesarios en la Figura 5.24.

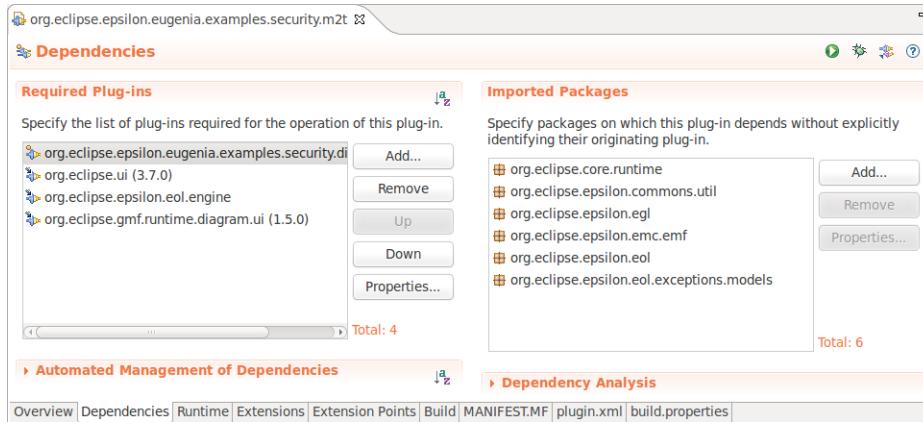


Figura 5.24.: *Plugin* y clase asociada

- Comprobar funcionalidad:
 - Una vez que se han realizado todos los pasos, ejecutamos Eclipse.
 - Miramos la ruta que aparece en *Location* al ejecutar Eclipse (*Run Configurations...*), ya que ahí es donde se encontrarán los proyectos que creemos y los ficheros en la segunda instancia de Eclipse.
 - Cuando nos encontramos con el editor gráfico, creamos un patrón de eventos complejo gráficamente utilizando la paleta del editor.
 - Pulsamos *botón derecho / Generate EPL code*.
- Código EPL generado:
 - El código EPL se habrá generado en el directorio local del proyecto en el que hemos creado el patrón en un fichero denominado *Code.epl* (ver ruta de *Location* y miramos en la carpeta del proyecto).
 - En nuestro caso en el campo *Location* aparece: \${workspace_loc}../runtime-New_configuration y el nombre del proyecto creado en la segunda instancia de Eclipse varios pasos atrás fue SecurityEPLdraw.

- Así que si tenemos nuestro *workspace* del Eclipse principal en */home/uca/workspace*, nuestro fichero *.epl* se encontrará en */home/uca/runtime-New_configuration/SecurityEPLdraw*.

5.5.4. Personalización del editor

La personalización realizada en el editor se ha llevado a cabo mediante las anotaciones GMF permitidas en Emfatic. Además, como las anotaciones son limitadas y no nos permite realizar una personalización completa de nuestro editor, se ha utilizado un *script* en EOL (denominado *ECore2GMF.eol*) que se divide en dos partes: una para la personalización de la paleta gráfica y otra para la personalización del resto del editor.

Paleta gráfica

El código del *script* en EOL que personaliza la paleta gráfica puede verse en Listado B.80.

La paleta gráfica se muestra en la Figura 5.25.

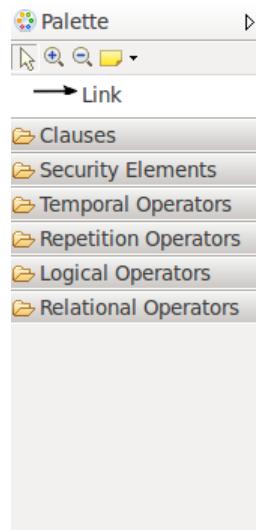


Figura 5.25.: Visión general de la paleta

- **Clauses:** Constituido por Pattern, Select e InsertInto tal como podemos ver en la Figura 5.26.

5. Desarrollo del proyecto

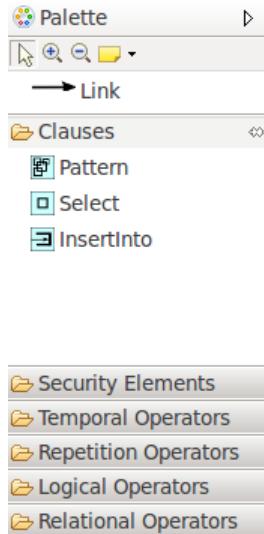


Figura 5.26.: Clauses en la paleta

- Security Events: Contiene Security Event, Value, y los atributos Port, IP, MAC y Timestamp (véase Figura 5.27).

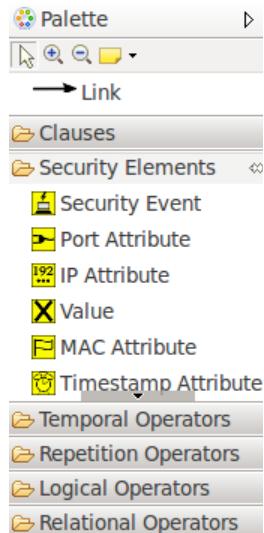


Figura 5.27.: Security Events en la paleta

- Temporal Operators: Se trata de Followed By, Timer Within, Timer Within Max, While, Timer Interval y Timer At (véase Figura 5.28).

5.5. Implementación

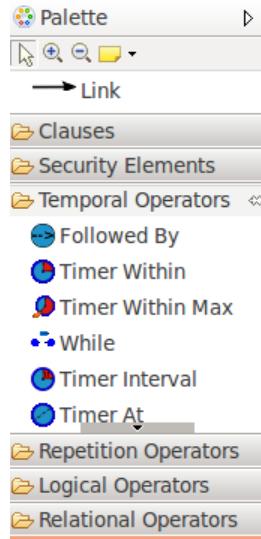


Figura 5.28.: Temporal Operators en la paleta

- Repetition Operators: Contiene Until, Every, Every Distinct y Range (véase Figura 5.29).

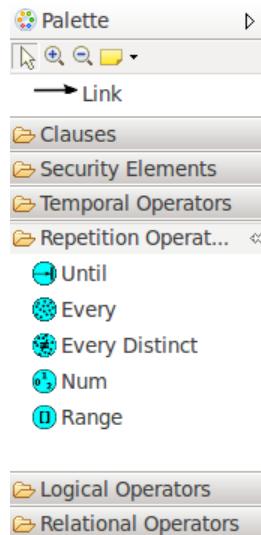


Figura 5.29.: Repetition Operators en la paleta

- Logical Operators: Tiene los elementos And, Or y Not (véase Figura 5.30).

5. Desarrollo del proyecto

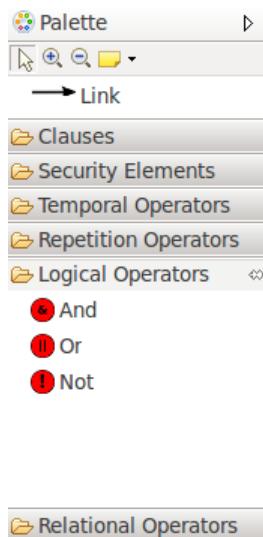


Figura 5.30.: Logical Operators en la paleta

- Relational Operators: Está formado por Equal, Not Equal, Less Than, Less Equal, Greater Than y Greater Equal Than (véase Figura 5.31).

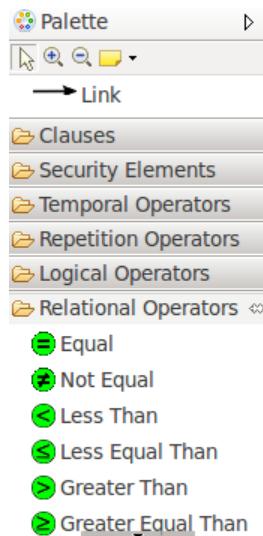


Figura 5.31.: Relational Operators en la paleta

Entorno principal del editor

El código del *script* en EOL que personaliza el entorno principal del editor puede verse en Listado B.81.

5.5. Implementación

El resultado de los elementos del entorno principal del editor gracias a las anotaciones GMF en Emfatic y del *script* en EOL, son los siguientes:

- Clauses. Figura 5.32.

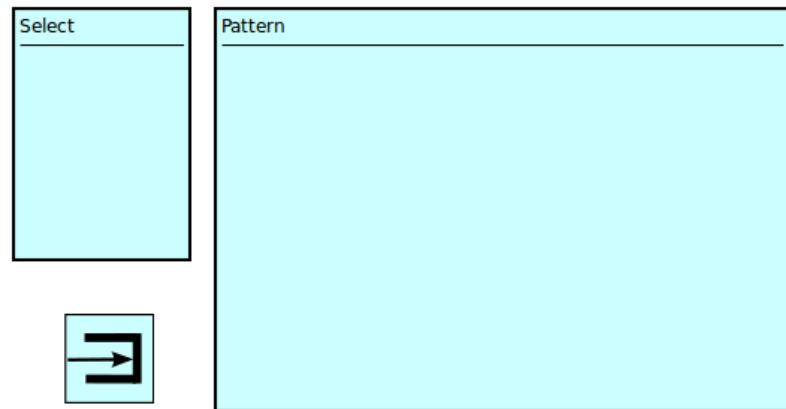


Figura 5.32.: Clauses en el entorno principal

- SecurityEvents. Figura 5.33.

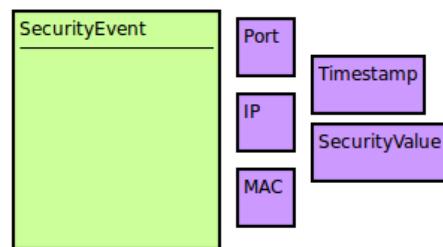


Figura 5.33.: Security Events en el entorno principal

- Temporal Operators. Figura 5.34.

5. Desarrollo del proyecto

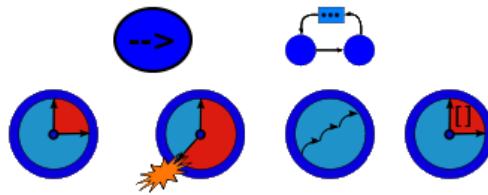


Figura 5.34.: Temporal Operators en el entorno principal

- Repetition Operators. Figura 5.35.

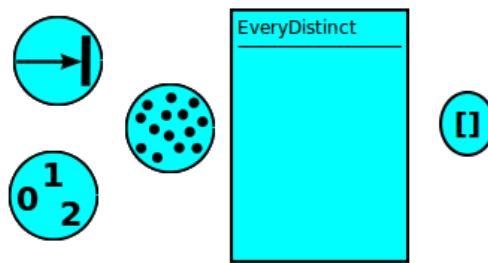


Figura 5.35.: Repetition Operators en el entorno principal

- Logical Operators. Figura 5.36.

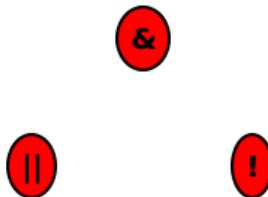


Figura 5.36.: Logical Operators en el entorno principal

- Relational Operators. Figura 5.37.

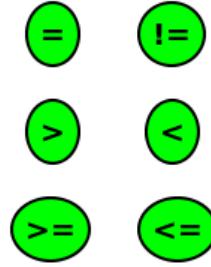


Figura 5.37.: Relational Operators en el entorno principal

5.6. Pruebas

Para la realización de las pruebas en esta aplicación, se ha utilizado EUnit (ya comentado anteriormente).

Se han realizado pruebas unitarias de caja negra, es decir, ejecutamos los casos de prueba y comparamos los resultados de las pruebas con las salidas esperadas.

En nuestro caso, hemos realizado las pruebas sobre las transformaciones generadas por la herramienta SecurityEPLdraw. Los modelos utilizados para las pruebas han sido definidos en HUTN [24].

En el Listado 5.1 podemos ver un ejemplo de cómo se define el patrón 5.2, a partir de nuestro metamodelo, en HUTN.

Listado 5.1: Ejemplo de modelo definido en HUTN

```

@Spec {
    Metamodel "security" {
        nsUri: "security"
    }
}

Model {
    Pattern{
        name: "PatternTimerInterval1"
        elements: SecurityEvent "SB"{
            name: "B"
            Alias: "b"
            order: "2"
        },
        Every "Every"{},
        TimerInterval "TI"{
            seconds: "1"
        },
    }
}

```

5. Desarrollo del proyecto

```
        FollowedBy "FB";
    }
    InsertInto{
        newStreamName: "TimerInterval1"
    }
    Select{
    }
    Link{
        source: FollowedBy "FB"
        target: Every "Every"
    }
    Link{
        source: FollowedBy "FB"
        target: SecurityEvent "SB"
    }
    Link{
        source: Every "Every"
        target: TimerInterval "TI"
    }
}
```

Listado 5.2: Patrón en EPL

```
String PatternTimerInterval1 = "@Name('PatternTimerInterval1')"
insert into TimerInterval1
select *
from pattern [ every timer:interval(1 seconds)
               -> b = B
]
```

Por tanto, el proceso de la ejecución de las pruebas en EUnit es el siguiente:

- Definimos nuestro modelo (caso de prueba) en notación HUTN.
- Indicamos a EUnit cuál es nuestro fichero EGL para realizar la generación de código EPL sobre ese modelo.
- Al generar el código EPL, se comprueba que la salida es la misma que la esperada, si no fuera así fallaría la prueba.

Para la herramienta desarrollada, se ha utilizado un gran conjunto de pruebas que cubre parte de los modelos que pueden ser generados a partir de nuestro metamodelo.

6. Conclusiones y trabajo futuro

En este capítulo trataremos las diferentes conclusiones extraídas de este PFC y el trabajo futuro.

6.1. Conclusiones

6.1.1. Valoración general

La herramienta SecurityEPLdraw desarrollada en este PFC ha sido llevada a cabo dentro del Grupo de Investigación “Grupo UCASE de Ingeniería del Software” (TIC-025) dentro de las áreas de Desarrollo del Software Dirigido por Modelos y Arquitecturas Dirigidas por Eventos.

En este PFC se ha desarrollado un editor gráfico (SecurityEPLdraw) en el que se puede diseñar patrones de eventos complejos para, posteriormente, generar código EPL basado en el modelo creado. Esta herramienta es un DSL orientada al campo de la seguridad informática. Se han utilizado técnicas de Desarrollo del Software Dirigido por Modelos para desarrollar la aplicación.

Se ha diseñado un metamodelo con la capacidad de representación para todos los modelos posibles que definen los patrones de eventos complejos sobre seguridad informática. Este metamodelo se ha implementado mediante el lenguaje diseñado para representar metamodelos Ecore textualmente, Emfatic.

Se ha conseguido la personalización completa del editor, utilizando todo el potencial que nos proporciona GMF pero utilizando EuGENia con la ayuda de un *script* en EOL desarrollado específicamente para esta aplicación.

Los modelos diseñados se validan para garantizar que los patrones diseñados son correctos mediante el lenguaje de validación EVL.

A partir de los modelos creados en el editor y validados, se puede transformar a código EPL en el propio editor gráfico, ya que se ha desarrollado un *plugin* que se integra con el editor. Esta transformación se consigue gracias al lenguaje EGL, lenguaje de la familia Epsilon que se encarga de transformaciones M2T.

Mediante un conjunto de pruebas unitarias definido en EUnit, se han validado las transformaciones realizadas por el editor.

6. Conclusiones y trabajo futuro

6.1.2. Valoración personal

He adquirido conocimientos en áreas en las que el alumnado no ha tenido la oportunidad de conocer durante la carrera (Ingeniería Técnica en Informática de Sistemas). Durante la realización del presente PFC he adquirido conocimientos sobre:

- Desarrollo del Software Dirigido por Modelos.
- Herramientas del entorno Eclipse que proveen todas las características para desarrollar este tipo de software.
- Procesamiento de eventos complejos.
- Elaboración de la memoria de un proyecto de mayor envergadura a los desarrollados durante la carrera.
- Trabajo en equipo con la participación en un grupo de investigación y la colaboración recíproca de cada uno de los componentes que forman parte de éste.
- Búsqueda bibliográfica avanzada sobre temas específicos para obtener documentación con mayor rigor técnico. Al tratarse de un proyecto de investigación, dicha búsqueda ha sido de gran dificultad al no encontrarse extensa documentación que albergara los temas tratados.

6.2. Trabajo futuro

La herramienta desarrollada podría mejorarse ampliando las posibilidades de creación de modelos sobre el lenguaje EPL.

El requisito fundamental para conseguir esta ampliación es mediante la modificación del metamodelo diseñado para este PFC, ya que con un buen diseño del metamodelo podremos continuar con el resto de fases que conforman el desarrollo del software para este tipo de herramienta.

Con el metamodelo actual se pueden diseñar la mayoría de los patrones más utilizados, pero para profundizar aún más en el diseño de los patrones de manera gráfica, habría que modificar el metamodelo actual y las fases posteriores hasta la generación de código.

Además, otra posible vía para ampliar este trabajo podría ser el desarrollo de herramientas de las mismas características aplicadas a otras áreas específicas, como la medicina o la banca. En conclusión, la herramienta desarrollada puede servir como base para lo expuesto anteriormente gracias a toda la documentación generada en este PFC.

7. Agradecimientos

- A Inmaculada Medina Bulo, por permitirme colaborar en el Grupo de Investigación UCASE.
- A Juan Boubeta Puig por darme la oportunidad de poder trabajar en este proyecto y por su gran dedicación, conocimiento y apoyo aportados durante la realización de éste.
- Al resto de miembros del Grupo de Investigación UCASE, por sus consejos para la elaboración de este proyecto.
- A mi familia, por el apoyo incondicional que me presta siempre y, en especial, el apoyo y comprensión recibidos durante la carrera.
- A María, porque sin su apoyo se hubieran hecho más duros los años de carrera y la realización de este proyecto.
- A mis compañeros de carrera, los cuales han hecho más fácil el día a día durante estos años.

7. Agradecimientos

8. Manual de instalación

En este capítulo se describirán los pasos necesarios para la instalación de la herramienta.

8.1. Instalación de Eclipse

La versión de Eclipse necesaria para la instalación de la herramienta se encuentra en el siguiente enlace [6] (esta versión es para la instalación de Eclipse en *Linux 64 bit*). Para ver el resto de versiones para otros sistemas operativos visitar el siguiente enlace [11]. Se encuentran concretamente en el apartado *Step 1:Download Eclipse*.

8.2. Instalación del resto de software

Una vez que tenemos esta versión de Eclipse, tenemos que seguir los siguientes pasos:

- Instalamos GMF Tooling a través del menú *Help->Install Modeling Components* dentro del entorno eclipse.
- Instalamos Emfatic a través de la url de actualización [10]. Para instalar algún componente a través de la url de actualización tenemos que hacerlo a través del menú *Help-> Install New Software... -> Add...* y en el apartado *Name* ponemos el nombre que creamos conveniente y en el apartado *Location* ponemos la url. Luego a través de la ventana *Install*, seleccionamos la url que hemos introducido, pulsamos *Next >* y seguimos las instrucciones de instalación.
- Instalamos Epsilon a través de la url de actualización [15].
- Instalamos Epsilon Interim a través de la url de actualización [13].

8.3. Importación de código

El código necesario para el funcionamiento de nuestra herramienta se puede importar de dos formas distintas:

8. Manual de instalación

- Instalación de Subclipse e importación de código mediante el repositorio: tenemos que seguir los pasos descritos en el Anexo A.
- Importación de código mediante una ubicación local: descomprimimos el fichero .zip que contiene los proyectos necesarios para la ejecución de la herramienta en un directorio. Nos dirigimos al Package Explorer de nuestro entorno Eclipse. Seguimos los siguientes pasos: *Botón derecho -> Import... -> Existing Projects into Workspace -> Browse... en Select root directory -> Elegimos nuestro directorio que contiene los proyectos con el código -> Next > y seguimos las instrucciones por defecto.*

9. Manual de usuario

En este capítulo se muestra un manual de usuario sobre cómo puede utilizarse la herramienta SecurityEPLdraw.

9.1. Ejecución del editor

Para ejecutar el editor, previamente tenemos que tener instalado todo el software necesario y el código de la aplicación (seguir los pasos descritos en 8).

Una vez que tenemos todo el software necesario para ejecutar nuestra aplicación, seguimos el siguiente menú *Run -> Run Configurations...* y en el menú que aparece en la izquierda seleccionar *Eclipse Application -> New_configuration* (véase Figura 9.1).

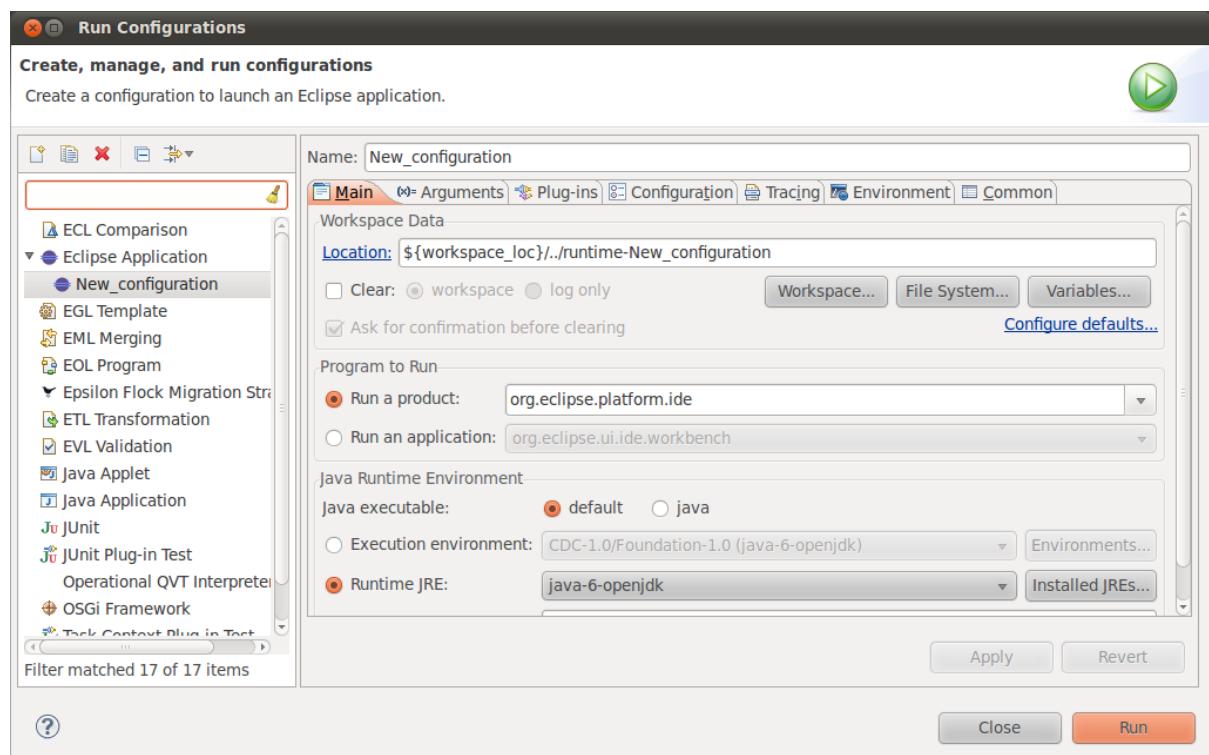


Figura 9.1.: Entorno de ejecución

9. Manual de usuario

En la pestaña *Arguments* cambiar los *VM Arguments* con: *-XX:PermSize=64M -XX:MaxPermSize=128M -Xms512M -Xmx1024M* tal como se indica en [21].

Ahora que tenemos la nueva instancia de Eclipse seguimos los siguientes pasos:

- Creamos un proyecto: *File -> New -> Project... -> General -> Project*
- Indicamos un nombre al nuevo proyecto, por ejemplo *SecurityEPLdraw*.

A continuación debemos de crear nuestra instancia del editor:

- *Botón derecho sobre el proyecto creado -> New -> Other... -> Security Diagram*
- Indicamos un nombre con extensión *.security_diagram*, indicando *SecurityEPLdraw* como *parent folder*.
- En el caso de dejarlo por defecto, se crean dos ficheros:
 - *default.security*
 - *default.security_diagram*
- Abrimos el archivo *default.security_diagram*
- El entorno del editor con el que nos encontramos lo podemos ver en la Figura 9.2.

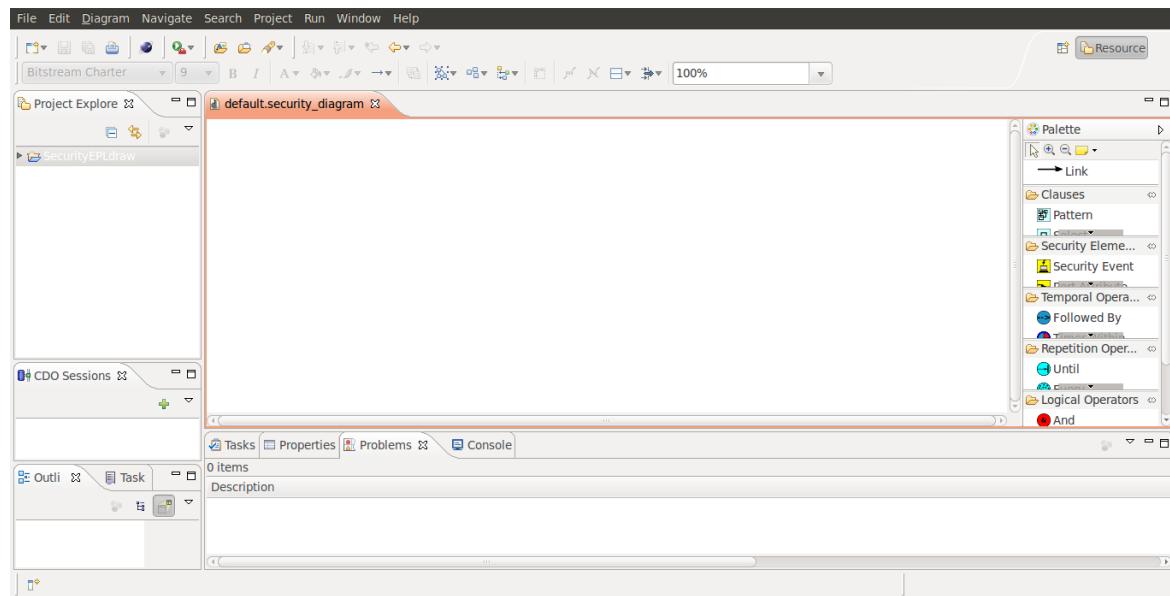


Figura 9.2.: Entorno vacío

9.2. Diseño de patrones de eventos complejos

9.2.1. Primeros elementos

Cuando estamos frente al editor, lo primero que tenemos que hacer es arrastrar los elementos Select, InsertInto (opcional) y Pattern al centro del editor. Debemos darle un valor al nombre de Pattern y si tenemos InsertInto, hay que darle un valor a newStreamName. Lo podemos ver en la Figura 9.3.

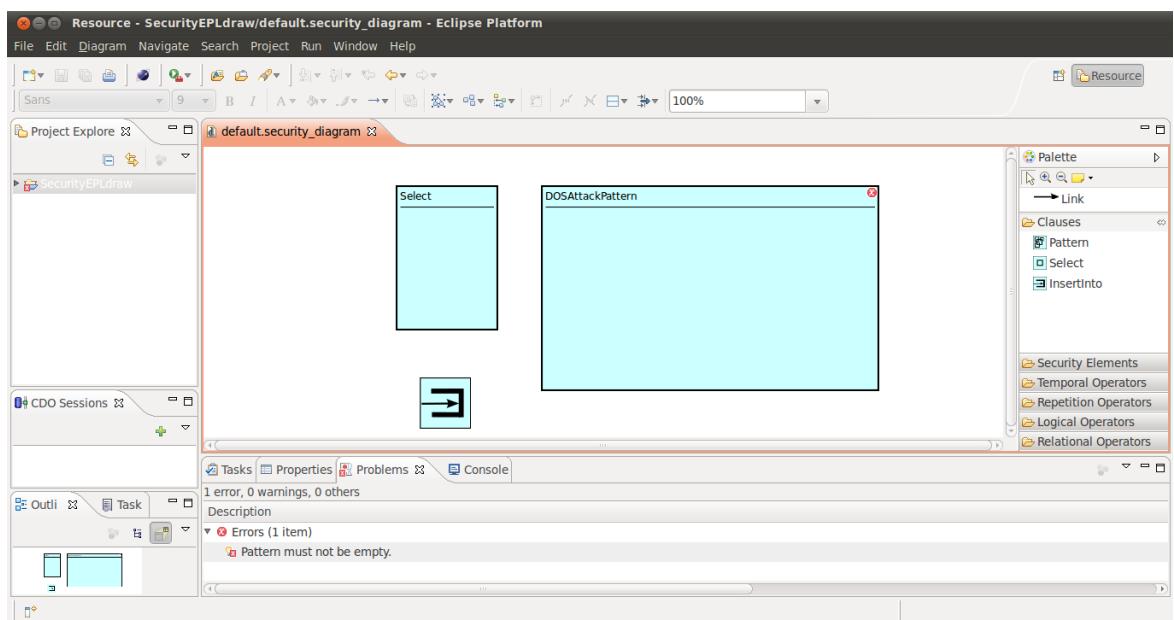


Figura 9.3.: Primeros Elementos

Como se puede observar en la Figura 9.3, nos sale un error debido a que Pattern no puede estar vacío (el editor valida si es correcto o no cada vez que guardamos el documento).

9.2.2. Ejemplos de patrones diseñados

A continuación se muestran algunos ejemplos de patrones de eventos complejos que podemos construir (ejemplos extraídos de [38]) y el código generado por la herramienta (en la siguiente sección se explica cómo generar el código del patrón diseñado):

- Ataque DOS: Figura 9.4 y código generado en el Listado 9.1.

9. Manual de usuario

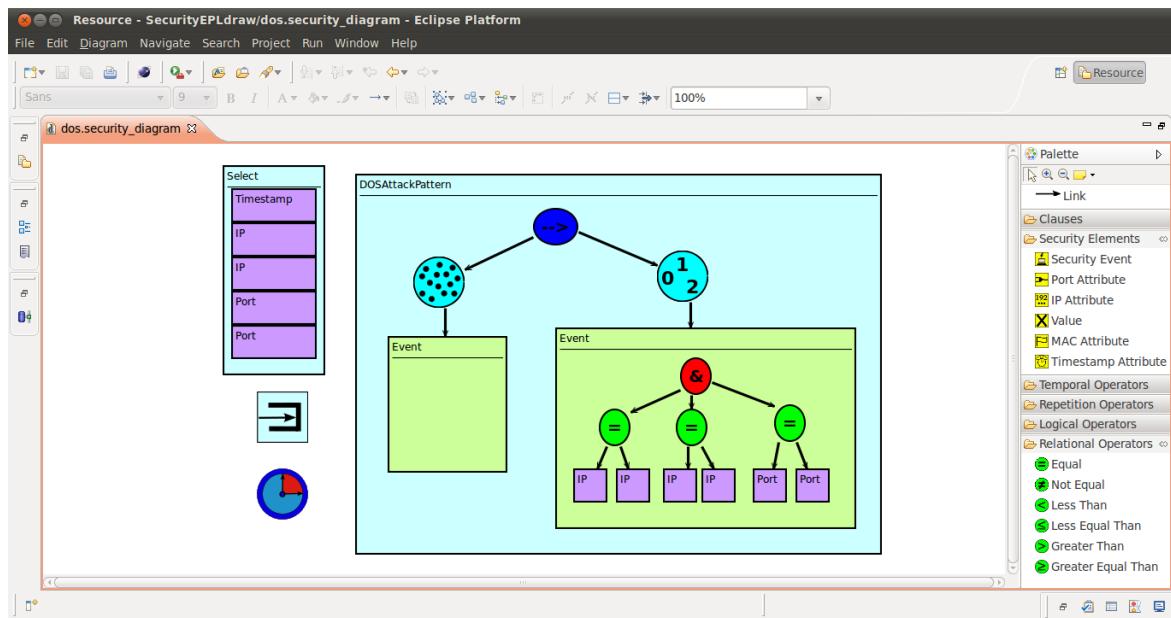


Figura 9.4.: Ataque DOS

Listado 9.1: DOS en EPL

```
String DOSAttackPattern = "@Name('DOSAttackPattern')  
insert into DOSAttack  
select  
    SuspectedDOSAttack.Timestamp,  
    SuspectedDOSAttack.IPSource as ipS,  
    SuspectedDOSAttack.IPTarget as ipT,  
    SuspectedDOSAttack.PortSource as ps,  
    SuspectedDOSAttack.PortTarget as pt  
from pattern [ every SuspectedDOSAttack = Event  
                -> [9] Event(IPSource = SuspectedDOSAttack.IPSource and  
                           IPTarget = SuspectedDOSAttack.IPTarget and  
                           PortTarget = SuspectedDOSAttack.PortTarget)  
where timer:within(1 minutes)  
] ";
```

- Ataque DDOS: Figura 9.5 y código generado en el Listado 9.2.

9.2. Diseño de patrones de eventos complejos

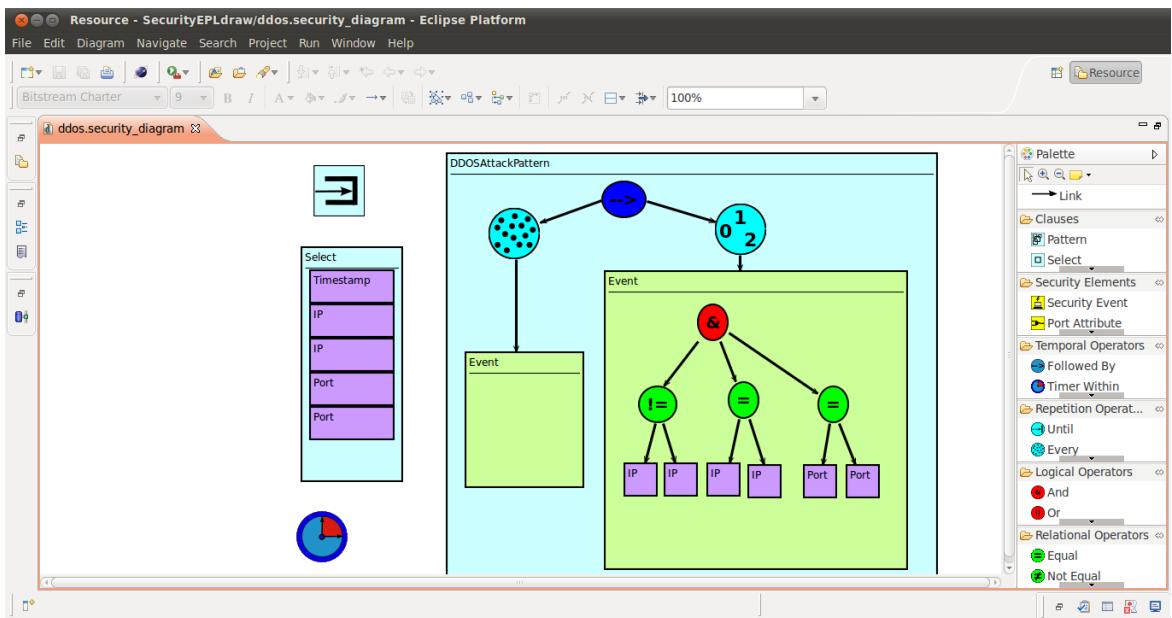


Figura 9.5.: Ataque DDOS

Listado 9.2: DDOS en EPL

```

String DDOSAttackPattern = "@Name( 'DDOSAttackPattern' )
insert into DDOSAttack
select
    SuspectedDDOSAttack.Timestamp,
    SuspectedDDOSAttack.IPSource as ipS,
    SuspectedDDOSAttack.IPSource as ipT,
    SuspectedDDOSAttack.PortSource as pS,
    SuspectedDDOSAttack.PortTarget as pT
from pattern [ every SuspectedDDOSAttack = Event
    -> [9] Event(IPSource != SuspectedDDOSAttack.IPSource
        and
        IPTarget = SuspectedDDOSAttack.IPTarget and
        PortTarget = SuspectedDDOSAttack.PortTarget)
where timer:within(1 minutes)
] ";

```

- Ataque Smurf: Figura 9.6 y código generado en el Listado 9.3.

9. Manual de usuario

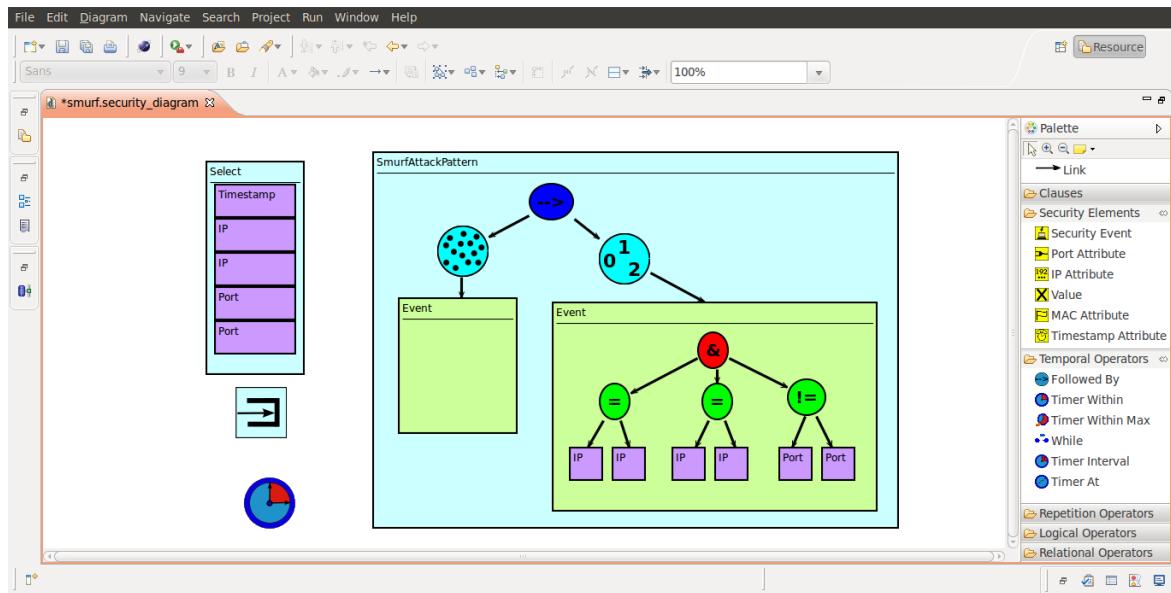


Figura 9.6.: Ataque Smurf

Listado 9.3: Smurf en EPL

```
String SmurfAttackPattern = "@Name( 'SmurfAttackPattern ' )
insert into SmurfAttack
select
    SuspectedSmurfAttack .Timestamp ,
    SuspectedSmurfAttack .IPSource as ipS ,
    SuspectedSmurfAttack .IPTarget as ipT ,
    SuspectedSmurfAttack .PortSource as pS ,
    SuspectedSmurfAttack .PortTarget as pT
from pattern [ every SuspectedSmurfAttack = Event
                -> [2] Event(IPSource = SuspectedSmurfAttack .IPTarget
                           and
                           IPTarget = SuspectedSmurfAttack .IPTarget and
                           PortSource != SuspectedSmurfAttack .PortTarget)
where timer:within(1 minutes)
] ";
```

- Ataque Land: Figura 9.7 y código generado en el Listado 9.4.

9.2. Diseño de patrones de eventos complejos

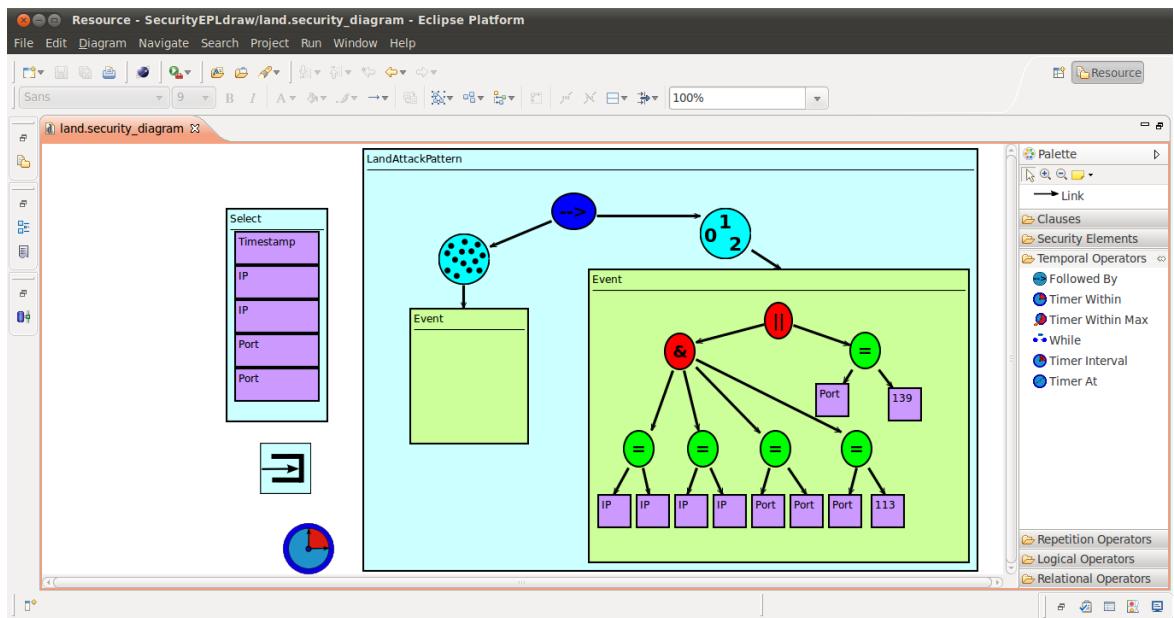


Figura 9.7.: Ataque Land

Listado 9.4: Land en EPL

```

String LandAttackPattern = "@Name('LandAttackPattern')"
insert into LandAttack
select
    SuspectedLandAttack.Timestamp,
    SuspectedLandAttack.IPSource as ipS,
    SuspectedLandAttack.IPTarget as ipT,
    SuspectedLandAttack.PortTarget as pS,
    SuspectedLandAttack.PortTarget as pT
from pattern [ every SuspectedLandAttack = Event
    -> [1] Event(IPSource = SuspectedLandAttack.IPTarget and
        IPTarget = SuspectedLandAttack.IPTarget and
        PortSource = SuspectedLandAttack.PortTarget and
        PortTarget = 113 or
        PortTarget = 139)
where timer:within(1 minutes)
];
  
```

- Ataque Supernuke: Figura 9.8 y código generado en el Listado 9.5.

9. Manual de usuario

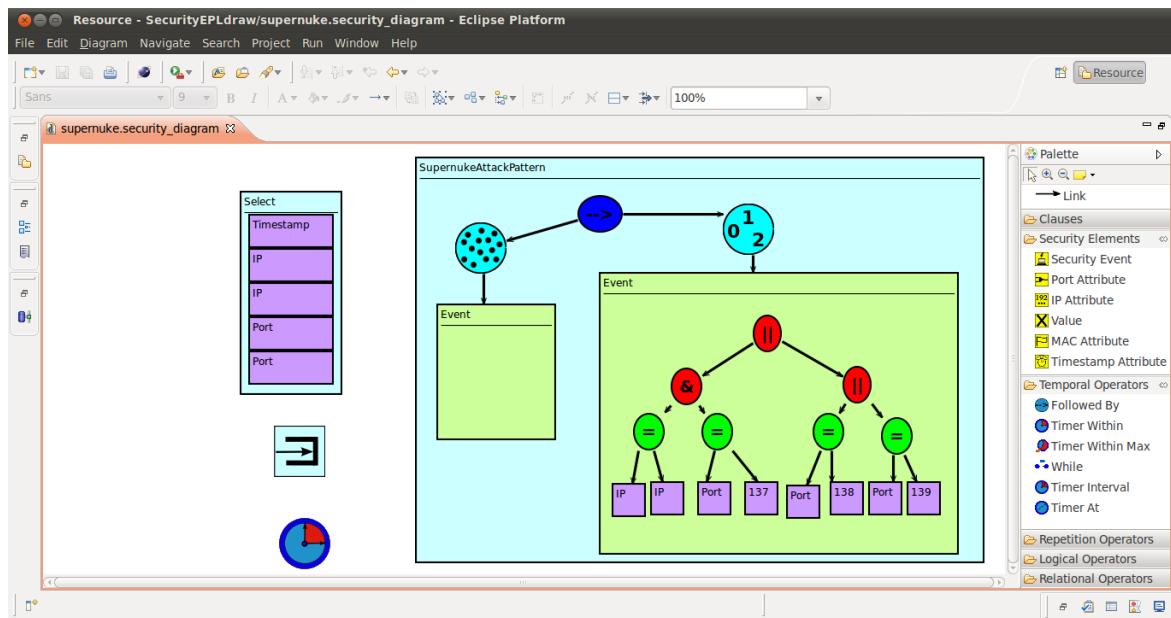


Figura 9.8.: Ataque Supernuke

Listado 9.5: Supernuke en EPL

```
String SupernukeAttackPattern = "@Name( 'SupernukeAttackPattern' )  
insert into SupernukeAttack  
select  
SuspectedSupernukeAttack.Timestamp,  
SuspectedSupernukeAttack.IPSource as ipS,  
SuspectedSupernukeAttack.IPTarget as ipT,  
SuspectedSupernukeAttack.PortSource as pS,  
SuspectedSupernukeAttack.PortTarget as pT  
from pattern [ every SuspectedSupernukeAttack = Event  
-> [9] Event(IPTarget = SuspectedSupernukeAttack.  
IPTarget and  
PortTarget = 137 or  
PortTarget = 138 or  
PortTarget = 139)  
where timer:within(1 minutes)  
] ";
```

- Ataque Escaneo de puertos TCP: Figura 9.9 y código generado en el Listado 9.6.

9.2. Diseño de patrones de eventos complejos

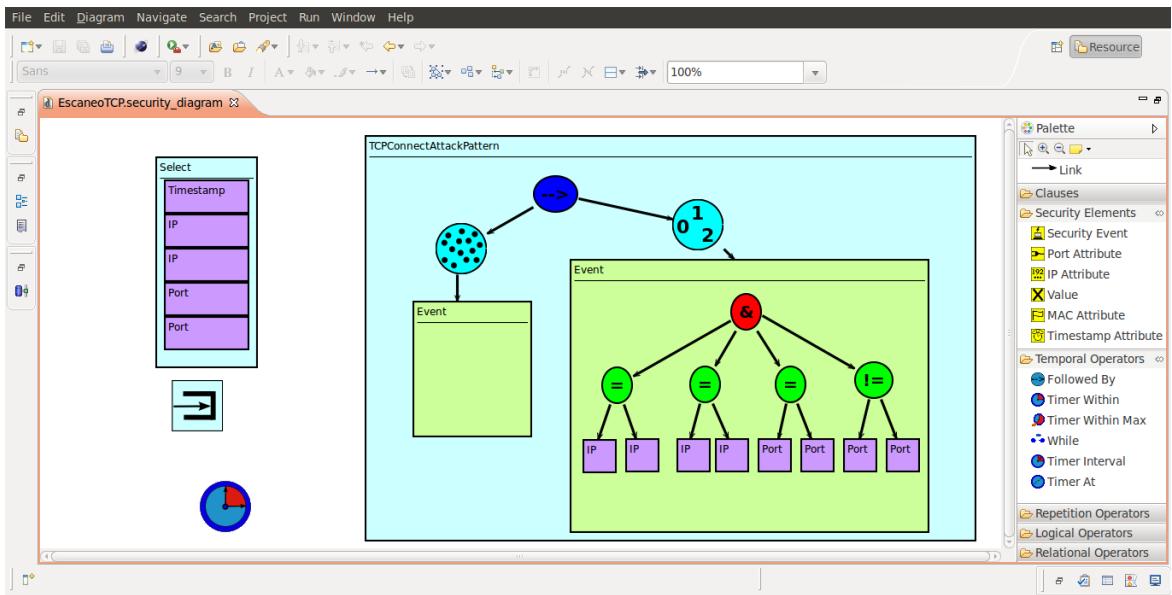


Figura 9.9.: Ataque escaneo de puertos TCP

Listado 9.6: Escaneo de puertos TCP en EPL

```
String TCPConnectAttackPattern = "@Name('TCPConnectAttackPattern')  
insert into TCPConnectAttack  
select  
SuspectedTCPConnectAttack.Timestamp,  
SuspectedTCPConnectAttack.IPSource as ipS,  
SuspectedTCPConnectAttack.IPTarget as ipT,  
SuspectedTCPConnectAttack.PortSource as pS,  
SuspectedTCPConnectAttack.PortTarget as pT  
from pattern [ every SuspectedTCPConnectAttack = Event  
-> [9] Event(IPSource = SuspectedTCPConnectAttack.  
IPSource and  
IPTarget = SuspectedTCPConnectAttack.IPTarget and  
PortSource = SuspectedTCPConnectAttack.PortSource and  
PortTarget != SuspectedTCPConnectAttack.PortTarget)  
where timer:within(1 minutes)  
] ";
```

- Ataque Flood al email: Figura 9.10 y código generado en el Listado 9.7.

9. Manual de usuario

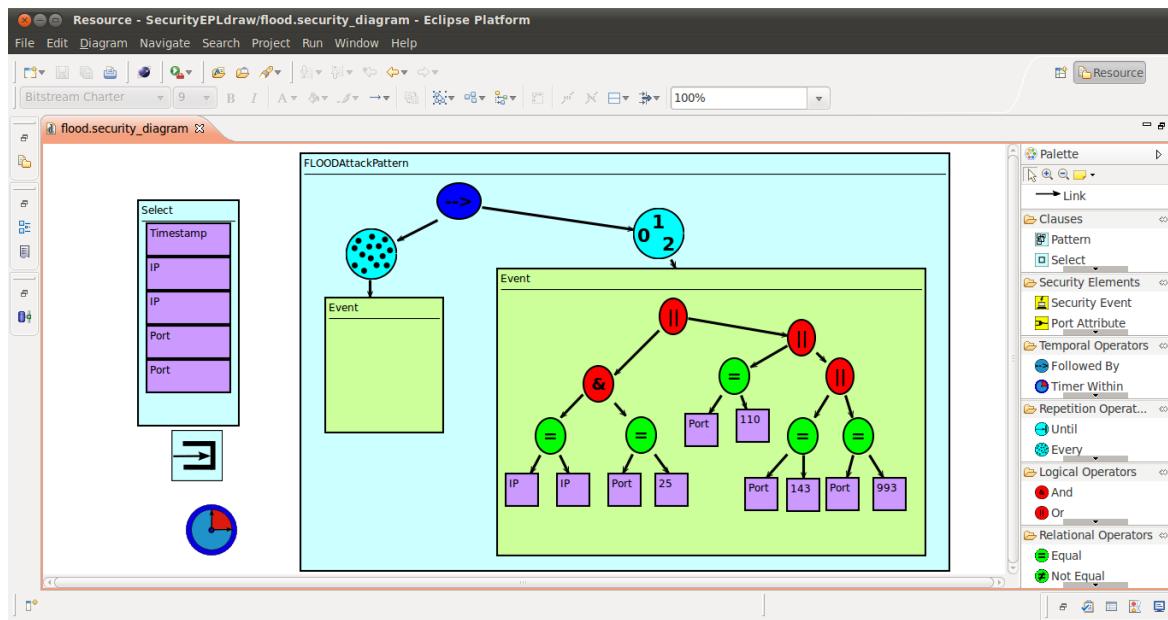


Figura 9.10.: Ataque Flood al email

Listado 9.7: Flood al email en EPL

```
String FLOODAttackPattern = "@Name( 'FLOODAttackPattern ' )
insert into FLOODAttack
select
SuspectedFLOODAttack . Timestamp ,
SuspectedFLOODAttack . IPSource as ipS ,
SuspectedFLOODAttack . IPTarget as ipT ,
SuspectedFLOODAttack . PortSource as pS ,
SuspectedFLOODAttack . PortTarget as pT
from pattern [ every SuspectedFLOODAttack = Event
-> [9] Event(IPTarget = SuspectedFLOODAttack . IPTarget
and
PortTarget = 25 or
PortTarget = 110 or
PortTarget = 143 or
PortTarget = 993)
where timer :within(1 minutes)
] ";
```

- Ataque al FTP: Figura 9.11 y código generado en el Listado 9.8.

9.2. Diseño de patrones de eventos complejos

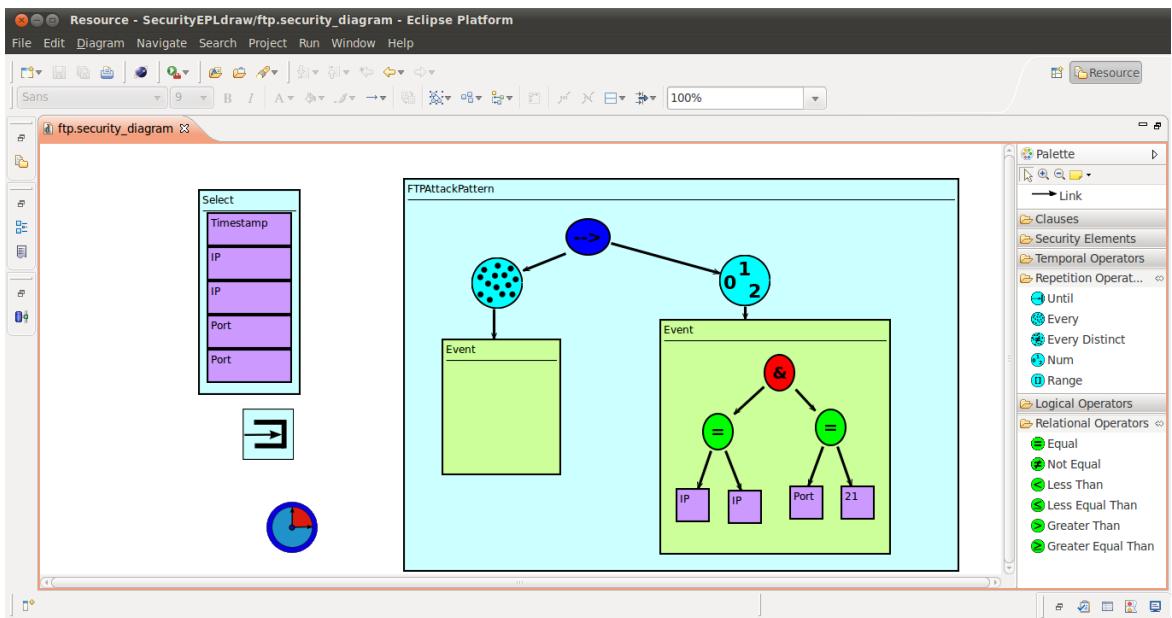


Figura 9.11.: Ataque al FTP

Listado 9.8: FTP en EPL

```
String FTPAttackPattern = "@Name( 'FTPAttackPattern' )  
insert into FTPAttack  
select  
SuspectedFTPAttack.Timestamp,  
SuspectedFTPAttack.IPSource as ipS,  
SuspectedFTPAttack.IPTarget as ipT,  
SuspectedFTPAttack.PortSource as pS,  
SuspectedFTPAttack.PortTarget as pT  
from pattern [ every SuspectedFTPAttack = Event  
-> [9] Event(IPTarget = SuspectedFTPAttack.IPTarget and  
PortTarget = 21)  
where timer:within(1 minutes)  
] ";
```

9.2.3. Generación de código EPL

Para generar el código en EPL del patrón construido hay que: *pulsar Botón derecho sobre el fondo del editor -> Generate EPL code*.

El código EPL se habrá generado en el directorio local del proyecto en el que hemos creado el patrón en un fichero denominado *Code.epl* (ver ruta de *Location* al ejecutar la instancia de Eclipse y miramos en la carpeta del proyecto).

9. Manual de usuario

Por ejemplo si en el campo *Location* aparece: *\$workspace_loc/..../runtime-New_-configuration*, el nombre del proyecto creado en la segunda instancia de eclipse varios pasos atrás fue *SecurityEPLdraw* y nuestro *workspace* de nuestro Eclipse principal lo tenemos en la ruta */home/uca/workspace*, nuestro fichero *Code.epl* se encontrará en */home/uca/runtime-New_configuration/SecurityEPLdraw*.

10. Manual del desarrollador

A continuación se presenta un manual para un desarrollador que desee colaborar en las modificaciones de la herramienta.

- Ante todo, el desarrollador debe de tener instalado todo el software necesario descrito en el capítulo 8.
- Una vez que el desarrollador tiene el software instalado y ha descargado todo el código, se debe centrar en los siguientes ficheros y rutas para hacer las modificaciones:
 - El metamodelo se encuentra en el fichero:
org.eclipse.epsilon.eugenia.examples.security/model/security.emf
 - El *script* encargado de la personalización al 100 % de la herramienta se encuentra en:
org.eclipse.epsilon.eugenia.examples.security/model/ECore2GMF.eol
 - La ubicación del fichero de las validaciones de los modelos es:
org.eclipse.epsilon.eugenia.examples.security.validation/validation/security.evl
 - El fichero de la transformación M2T se encuentra en:
org.eclipse.epsilon.eugenia.examples.security.m2t/src/eglExample.egl
 - El fichero con las pruebas en EUnit se encuentra en:
org.eclipse.epsilon.eugenia.examples.security.eunit/tests.eunit
 - Para incluir las imágenes de los nodos en el entorno principal, seguir este tutorial [20].
 - Las imágenes pequeñas (en la paleta, en las propiedades, etc...) se encuentran en:
org.eclipse.epsilon.eugenia.examples.security.edit/icons/full/obj16

10. Manual del desarrollador

A. Subclipse

En este Anexo, mostraremos cómo instalar y configurar la herramienta Subclipse, necesaria si queremos importar el código desde el repositorio en el que se aloja, o si queremos seguir desarrollando la aplicación.

A.1. Instalación

En el siguiente enlace [2] podremos descargar la versión utilizada en este PFC. La versión utilizada ha sido *1.8.x Release* y además hay unas instrucciones que hay que seguir para instalar y abrir una perspectiva en Eclipse.

A.2. Configuración

Una vez que tenemos que abrimos la perspectiva SVN, seguimos los siguientes pasos: *botón derecho -> New -> Repository Location.. -> insertamos la url del repositorio del editor*, que en nuestro caso es
<https://neptuno.uca.es/svn/pfc-imorales/> *SecurityEPLdraw-code*.

Ahora *seleccionamos todos los proyectos -> pulsamos botón derecho -> checkout...* (este paso es la importación del código del repositorio).

A. Subclipse

B. Código Fuente

B.1. Metamodelo

A continuación se muestra cada una de las partes del metamodelo con su código asociado.

- **Root.** Figura B.1 y código en el Listado B.1.

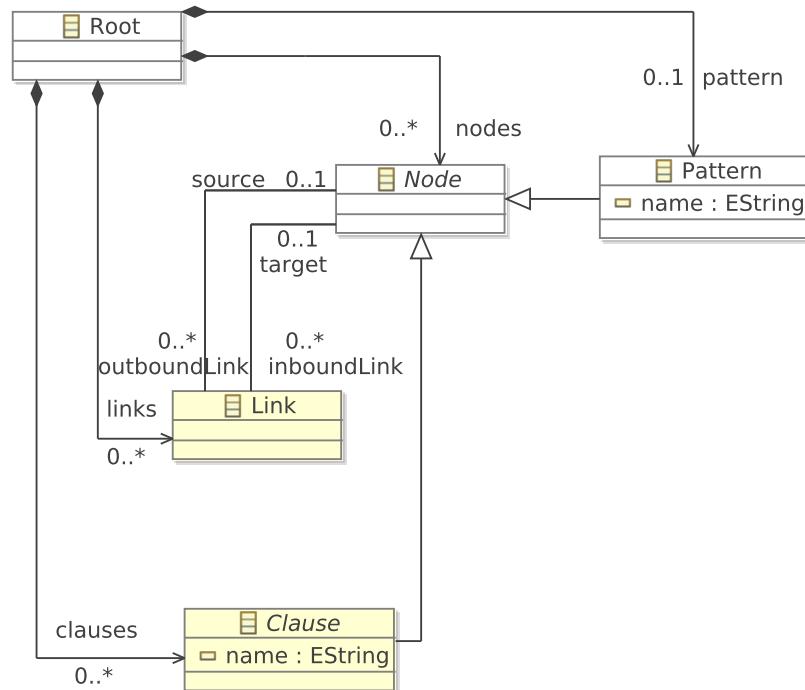


Figura B.1.: Root

Listado B.1: Root en Emfatic

```
@gmf.diagram(foo="bar")
class Root {
```

B. Código Fuente

```

val Clause [*] clauses ;
val Pattern [0..1] pattern ;
val Link [*] links ;
val Node [*] nodes ;
}

```

- **Node.** Figura B.2 y su código en el Listado B.2.

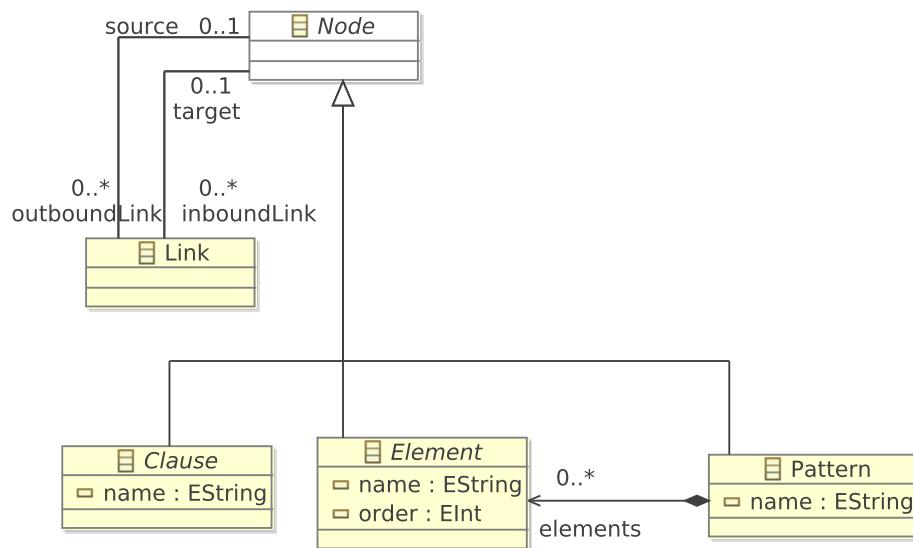


Figura B.2.: Node

Listado B.2: Node en Emfatic

```

abstract class Node{
    ref Link [*]# source outboundLink ;
    ref Link [*]# target inboundLink ;
}

```

- **Link.** Figura B.3 y su código en el Listado B.3 .

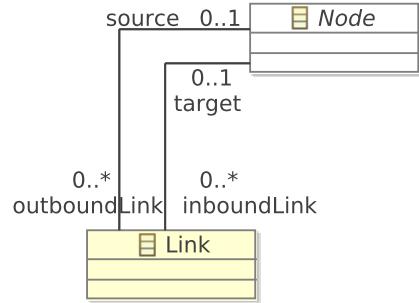


Figura B.3.: Link

Listado B.3: Link en Emfatic

```

@gmf.link(source="source", source.decoration="none", target="target",
target.decoration="arrow", style="solid", width="3", color="0,0,0",
tool.name="Link", tool.description="Add link",
tool.small.path="FlowIcon.png", tool.small.bundle="org.eclipse.epsilon.eugenia.examples.security.figures/images")
class Link{
    ref Node#outboundLink source;
    ref Node#inboundLink target;
}
  
```

- **Clause.** Figura B.4 y su código en el Listado B.4.

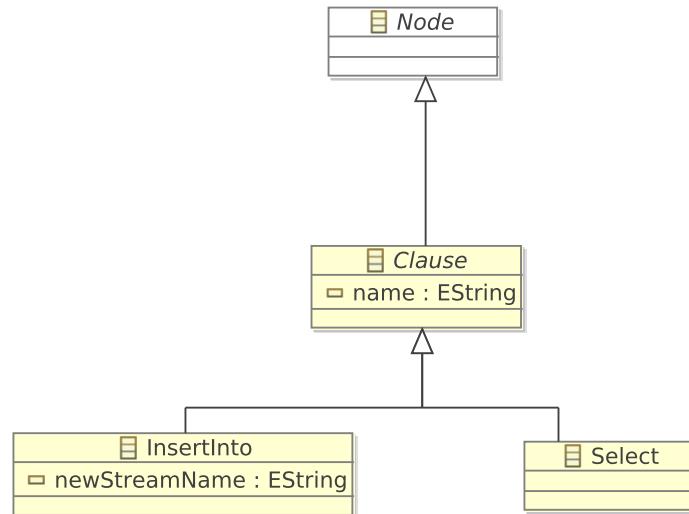


Figura B.4.: Clause

B. Código Fuente

Listado B.4: Clause en Emfatic

```
abstract class Clause extends Node{
    attr String name;
}
```

- **Select.** Figura B.5 y su código en el Listado B.5.

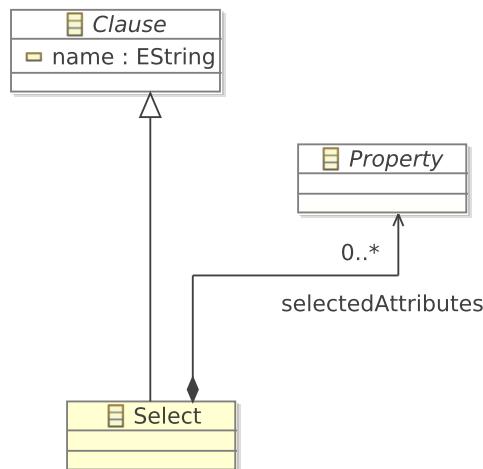


Figura B.5.: Select

Listado B.5: Select en Emfatic

```
@gmf.node(figure = "rectangle", label = "name", label.icon = "false",
label.readOnly="true",
color = "204,255,255", tool.name="Select", tool.description="Add Select
Clause",
border.color = "0,0,0", border.width="2", size="120,170")
class Select extends Clause {

    @gmf.compartment(foo="bar", layout = "list")
    val Property [*] selectedAttributes;
}
```

- **InsertInto.** Figura B.6 y su código en el Listado B.6.

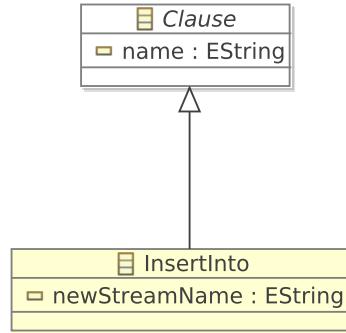


Figura B.6.: InsertInto

Listado B.6: InsertInto en Emfatic

```

@gmf.node(figure="figures.InsertIntoFigure", label.placement="none",
          tool.name="InsertInto", tool.description="Add Insert Into Operator",
          tool.small.path="InsertInto.gif", tool.small.bundle="/org.eclipse.
          epsilon.eugenia.examples.security.edit/icons/full/obj16/")
class InsertInto extends Clause {
    attr String newStreamName;
}
  
```

- **Pattern.** Figura B.7 y su código en el Listado B.7.

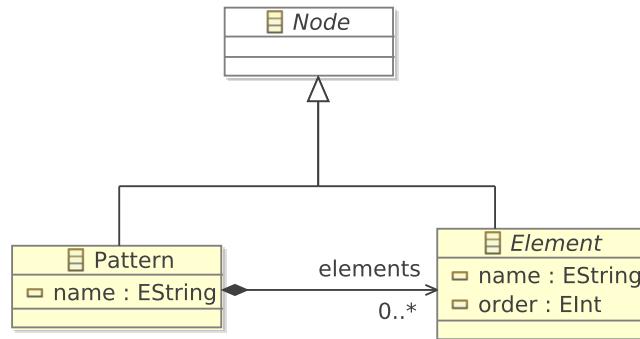


Figura B.7.: Pattern

Listado B.7: Pattern en Emfatic

```

@gmf.node(figure = "rectangle", label = "name", label.icon = "false",
          label.readOnly="true",
  
```

B. Código Fuente

```
color = "204,255,255", tool.name="Pattern", tool.description="Add
    Pattern Clause",
border.color = "0,0,0", border.width="2", size = "400,400")
class Pattern extends Node{
    attr String name;
    @gmf.compartment(foo="bar")
    val Element [*] elements;
}
```

- **Element.** Figura B.8 y su código en el Listado B.8.

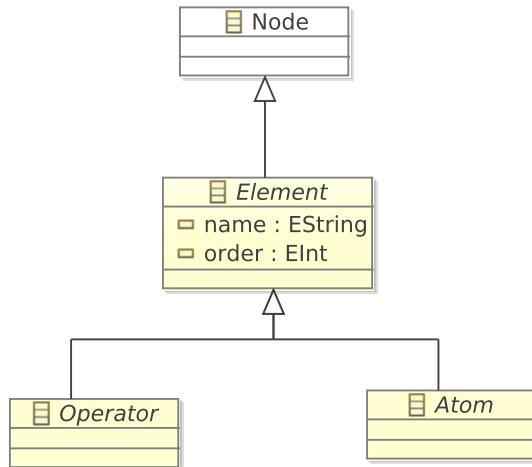


Figura B.8.: Element

Listado B.8: Element en Emfatic

```
abstract class Element extends Node{
    attr String name;
    attr int order = 1;
}
```

- **Atom - Operator - Observer.** Figura B.9 y su código en el Listado B.9.

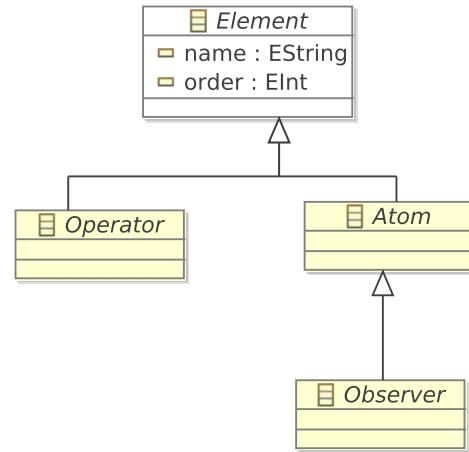


Figura B.9.: Atom - Operator - Observer

Listado B.9: Atom - Operator - Observer en Emfatic

```
abstract class Atom extends Element {
}

abstract class Operator extends Element {
}

abstract class Observer extends Atom {
```

- **Event - SecurityEvent.** Figura B.10 y su código asociado en el Listado B.10.

B. Código Fuente

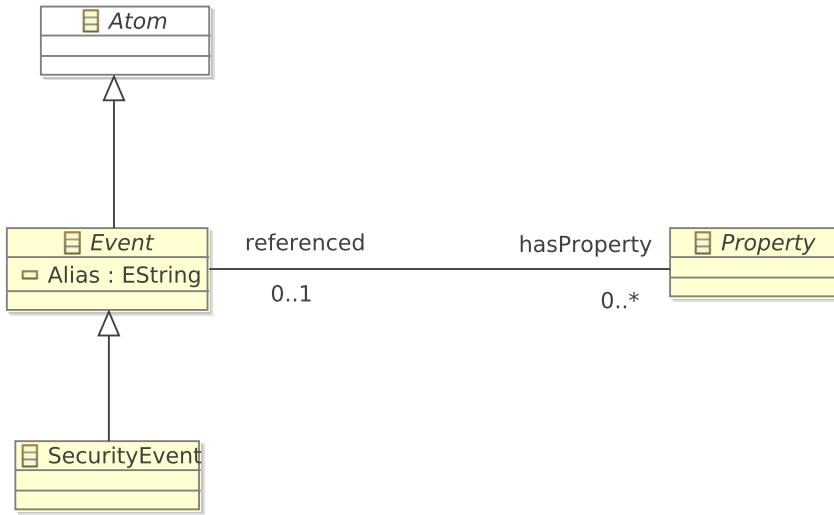


Figura B.10.: Event - SecurityEvent

Listado B.10: Event - SecurityEvent en Emfatic

```

abstract class Event extends Atom {
    attr String Alias;

    ref Property [*]# referenced hasProperty;

    @gmf.compartment( foo="bar ")
    val Element [*] conditions;
}

@gmf.node( figure = "rectangle", label = "name", label.icon = "false",
    label.readOnly="false",
    color = "204,255,153", tool.name="SecurityEvent", tool.description="Add
    Security Event",
    border.color = "0,0,0", border.width="2", size="140,160")
class SecurityEvent extends Event {

}
  
```

- **Attribute - Property - Value - SecurityValue.** CAMBIAR.

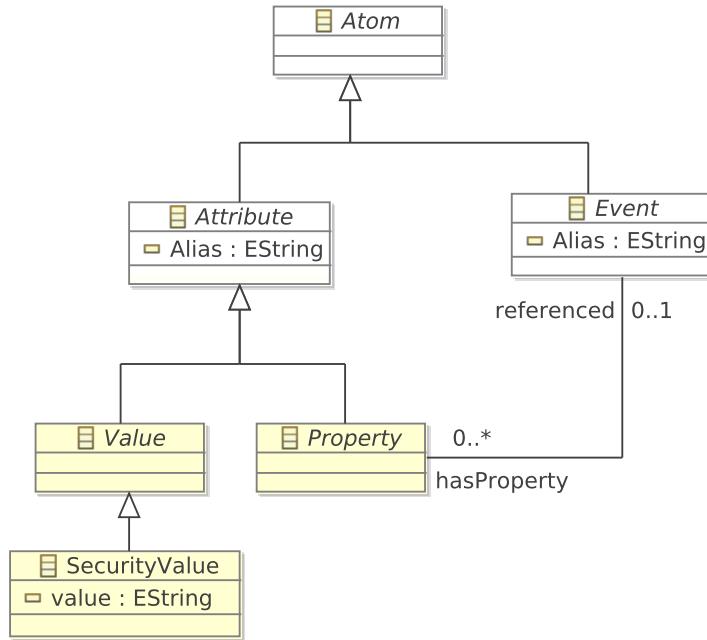


Figura B.11.: Event - SecurityEvent

Listado B.11: Attribute - Property - Value - SecurityValue en Emfatic

```

abstract class Attribute extends Atom {
    attr String Alias;
}

abstract class Property extends Attribute {
    ref Event[0..1]#hasProperty referenced;
}

abstract class Event extends Atom {
    attr String Alias;

    ref Property[*]#referenced hasProperty;

    @gmf.compartment(foo="bar")
    val Element[*] conditions;
}

@gmf.node(figure = "rectangle", label = "name", label.icon = "false",
    label.readOnly="false",
    color = "204,255,153", tool.name="SecurityEvent", tool.description="Add
        Security Event",
    border.color = "0,0,0", border.width="2", size="140,160")
  
```

B. Código Fuente

```
class SecurityEvent extends Event {  
}
```

- **IP - Port - MAC -Timestamp.** Figura B.12 y su código en el Listado B.12.

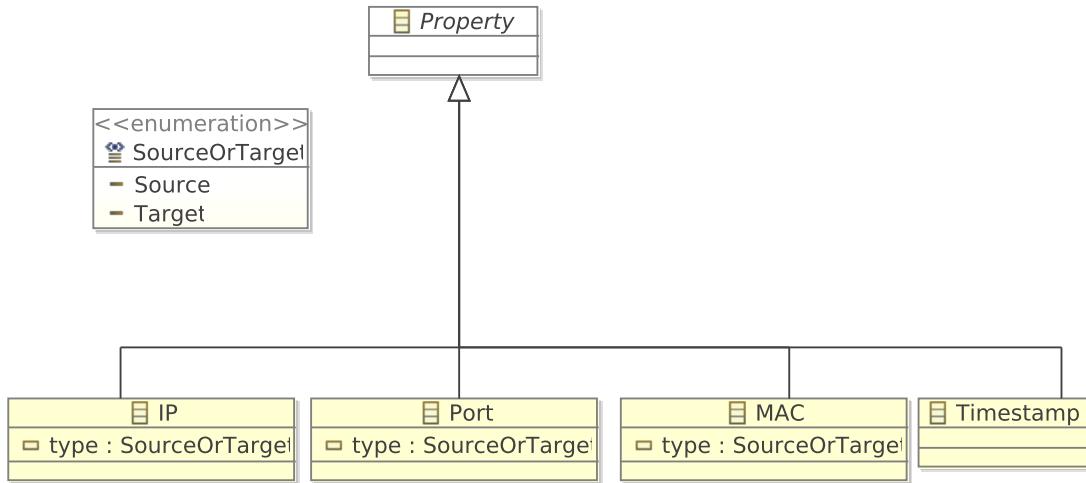


Figura B.12.: IP - Port - MAC - Timestamp

Listado B.12: IP - Port - MAC - Timestamp en Emfatic

```
enum SourceOrTarget {  
    Source;  
    Target;  
}  
  
@gmf.node(figure = "rectangle", label = "name", label.icon = "false",  
    label.readOnly="true",  
color = "204,153,255", tool.name="IP", tool.description="Add IP  
    Attribute",  
border.color = "0,0,0", border.width="2")  
class IP extends Property {  
    attr SourceOrTarget type;  
}  
  
@gmf.node(figure = "rectangle", label = "name", label.icon = "false",  
    label.readOnly="true",  
color = "204,153,255", tool.name="Port", tool.description="Add Port  
    Attribute",  
border.color = "0,0,0", border.width="2")  
class Port extends Property {  
    attr SourceOrTarget type;  
}  
  
@gmf.node(figure = "rectangle", label = "name", label.icon = "false",  
    label.readOnly="true",  
color = "204,153,255", tool.name="MAC", tool.description="Add MAC  
    Attribute",  
border.color = "0,0,0", border.width="2")  
class MAC extends Property {  
    attr SourceOrTarget type;  
}  
  
@gmf.node(figure = "rectangle", label = "name", label.icon = "false",  
    label.readOnly="true",  
color = "204,153,255", tool.name="Timestamp", tool.description="Add  
    Timestamp Attribute",  
border.color = "0,0,0", border.width="2")  
class Timestamp extends Property {  
    attr SourceOrTarget type;  
}
```

```

border.color = "0,0,0", border.width="2")
class Port extends Property {
    attr SourceOrTarget type;
}

@gmf.node(figure = "rectangle", label = "name", label.icon = "false",
label.readOnly="true",
color = "204,153,255", tool.name="MAC", tool.description="Add MAC
Attribute",
border.color = "0,0,0", border.width="2")
class MAC extends Property {
    attr SourceOrTarget type;
}

@gmf.node(figure = "rectangle", label = "name", label.icon = "false",
label.readOnly="true",
color = "204,153,255", tool.name="Timestamp", tool.description="Add
Timestamp Attribute",
border.color = "0,0,0", border.width="2")
class Timestamp extends Property {
}

```

- **Unary - Binary - Nary - Repetition - Logical - Temporal - Guard - Relational.** Figura B.13 y su código en el Listado B.13.

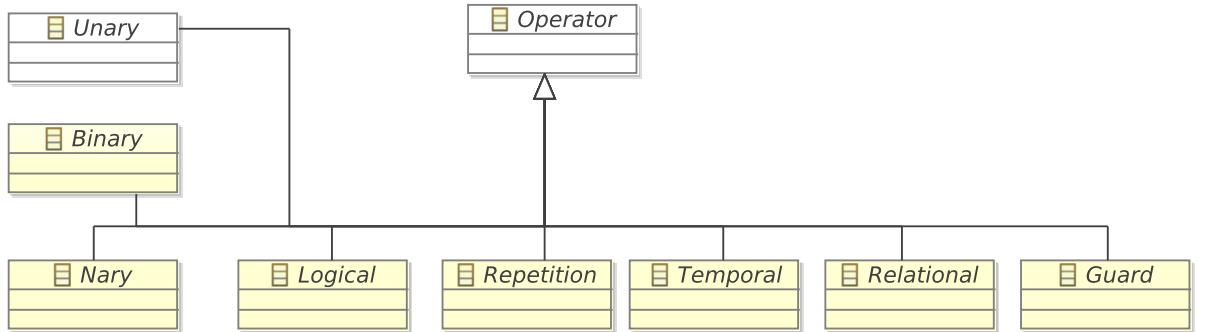


Figura B.13.: Unary-Binary-Nary-Repetition-Logical-Temporal-Guard-Relational

Listado B.13: Unary-Binary-Nary-Repetition-Logical-Temporal-Guard-Relational
en Emfatic

```

abstract class Unary extends Operator {

```

B. Código Fuente

```
}
```

```
abstract class Binary extends Operator {
```

```
}
```

```
abstract class Nary extends Operator {
```

```
}
```

```
abstract class Repetition extends Operator {
```

```
}
```

```
abstract class Logical extends Operator{
```

```
}
```

```
abstract class Temporal extends Operator {
```

```
}
```

```
abstract class Guard extends Operator {
```

```
}
```

```
abstract class Relational extends Operator {
```

```
}
```

- **Until - Every - EveryDistinct - Num - Range.** Figura B.13 y su código en el Listado B.14.

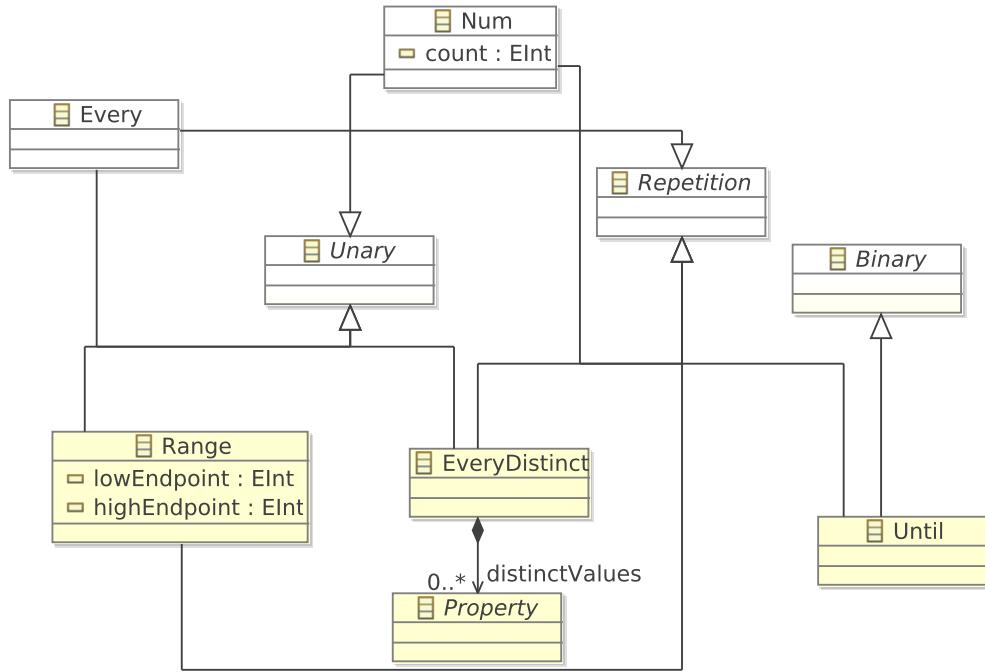


Figura B.14.: Until - Every - EveryDistinct - Num - Range

Listado B.14: Until-Every-EveryDistinct-Num-Range en Emfatic

```

@gmf.node(figure="figures.UntilFigure", label.placement = "none",
tool.name="Until", tool.description="Add Until Operator")
class Until extends Binary, Repetition {

}

@gmf.node(figure="figures.EveryFigure", label.placement = "none",
tool.name="Every", tool.description="Add Every Operator")
class Every extends Unary, Repetition {

}

@gmf.node(figure = "rectangle", label = "name", label.icon = "false",
label.readOnly="true",
color = "0,255,255", tool.name="EveryDistinct", tool.description="Add
Every Distinct Operator",
border.color = "0,0,0", border.width="2", size="120,170")
class EveryDistinct extends Unary, Repetition {
    @gmf.compartment(foo="bar", layout = "list")
    val Property [*] distinctValues;
}
  
```

B. Código Fuente

```

@gmf.node(figure="figures.NumFigure", label.placement = "none",
tool.name="Num", tool.description="Add Num Operator")
class Num extends Unary, Repetition {
    attr int count = 1;
}

@gmf.node(figure="ellipse", label="name", label.icon = "false", label.
    readOnly="true", color = "0,255,255",
tool.name="Range", tool.description="Add Range Operator", border.color
= "0,0,0", border.width="2")
class Range extends Unary, Repetition {
    attr int lowEndpoint;
    attr int highEndpoint;
}

```

- **And - Or - Not.** Figura B.15 y su código en el Listado B.15.

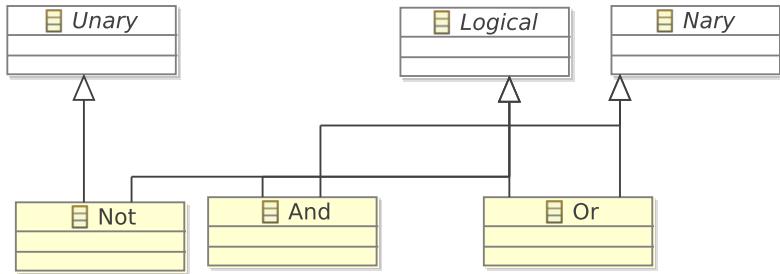


Figura B.15.: And - Or - Not

Listado B.15: And - Or - Not en Emfatic

```

@gmf.node(figure="ellipse", label="name", label.icon = "false", label.
    readOnly="true", color = "255,0,0",
tool.name="And", tool.description="Add And operator", border.color =
"0,0,0", border.width="2")
class And extends Nary, Logical {

}

@gmf.node(figure="ellipse", label="name", label.icon = "false", label.
    readOnly="true", color = "255,0,0",
tool.name="Or", tool.description="Add Or operator", border.color =
"0,0,0", border.width="2")
class Or extends Nary, Logical {
}

```

```

}

@gmf.node(figure="ellipse", label="name", label.icon = "false", label.
    readOnly="true", color = "255,0,0",
    tool.name="Not", tool.description="Add Not operator", border.color =
    "0,0,0", border.width="2")
class Not extends Unary, Logical {
}

```

- **FollowedBy.** Figura B.16 y su código en el Listado B.16.

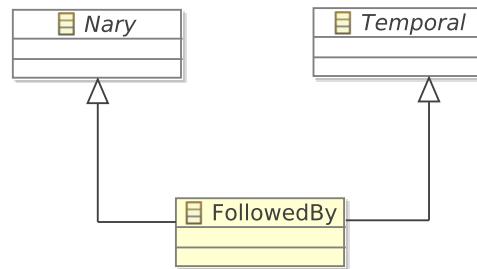


Figura B.16.: FollowedBy

Listado B.16: FollowedBy en Emfatic

```

@gmf.node(figure="ellipse", label="name", label.icon = "false", label.
    readOnly="true", color = "0,0,255",
    tool.name="FollowedBy", tool.description="Add Followed By Operator",
    border.color = "0,0,0", border.width="2")
class FollowedBy extends Nary, Temporal {
}

```

- **TimerWithin - TimerWithinMax - While.** Figura B.17 y su código en el Listado B.17.

B. Código Fuente

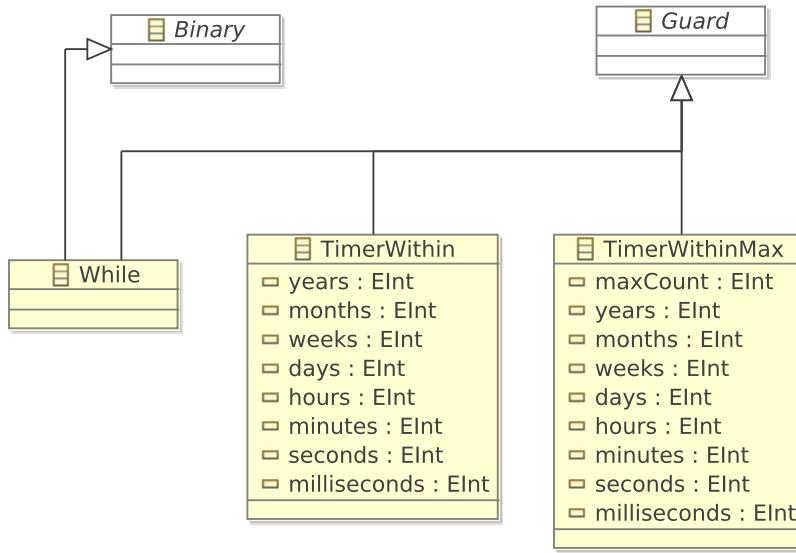


Figura B.17.: TimerWithin - TimerWithinMax - While

Listado B.17: TimerWithin - TimerWithinMax - While en Emfatic

```

@gmf.node( figure="figures.TimerWithinFigure", label.placement = "none",
tool.name="TimerWithin", tool.description="Add Timer Within Operator")
class TimerWithin extends Guard {
    attr int years;
    attr int months;
    attr int weeks;
    attr int days;
    attr int hours;
    attr int minutes;
    attr int seconds;
    attr int milliseconds;
}

@gmf.node( figure="figures.TimerWithinMaxFigure", label.placement = "
none",
tool.name="TimerWithinMax", tool.description="Add Timer Within Max
Operator")
class TimerWithinMax extends Guard {
    attr int maxCount;
    attr int years;
    attr int months;
    attr int weeks;
    attr int days;
    attr int hours;
    attr int minutes;
}
  
```

```

    attr int seconds;
    attr int milliseconds;
}

@gmf.node(figure="figures.WhileFigure", label="name",
tool.name="While", tool.description="Add While Operator")
class While extends Binary, Guard {

}

```

- **TimerInterval - TimerAt.** Figura B.18 y su código en el Listado B.18.

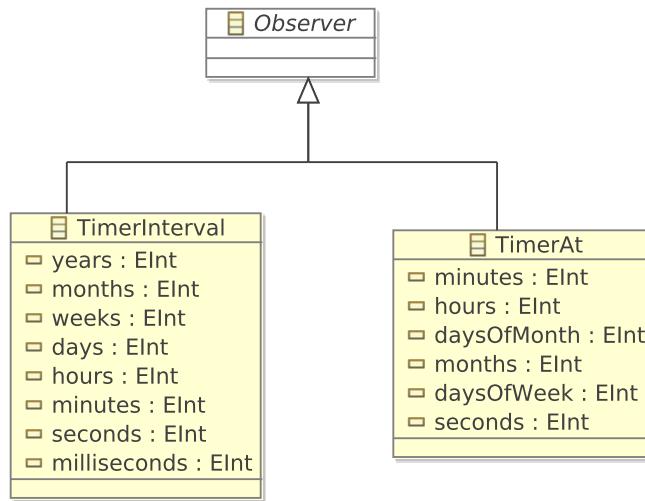


Figura B.18.: TimerInterval - TimerAt

Listado B.18: TimerInterval - TimerAt en Emfatic

```

@gmf.node(figure="figures.TimerIntervalFigure", label.placement = "none"
",
tool.name="TimerInterval", tool.description="Add Timer Interval
Operator")
class TimerInterval extends Observer{
    attr int years;
    attr int months;
    attr int weeks;
    attr int days;
    attr int hours;
    attr int minutes;
    attr int seconds;
}

```

B. Código Fuente

```

        attr int milliseconds;
    }

@gmf.node( figure="figures.TimerAtFigure", label.placement = "none",
tool.name="TimerAt", tool.description="Add Timer At Operator")
class TimerAt extends Observer {
    attr int minutes;
    attr int hours;
    attr int daysOfMonth;
    attr int months;
    attr int daysOfWeek;
    attr int seconds;
}

```

- **Equal - NotEqual - LessThan - GreaterThan - LessEqual - GreaterEqual.** Figura B.19 y su código en el Listado B.19.

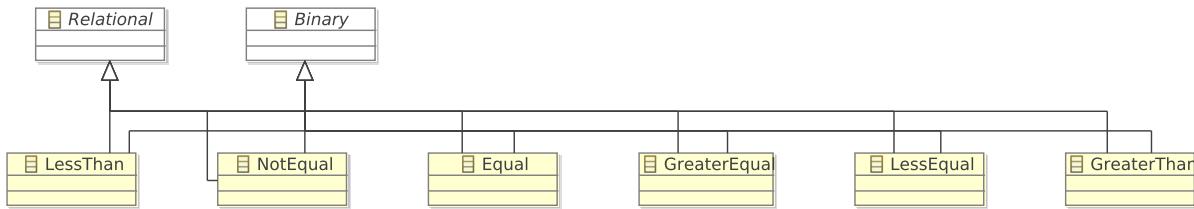


Figura B.19.: Operadores Relacionales

Listado B.19: Relational en Emfatic

```

@gmf.node( figure="ellipse", label="name", label.icon = "false", label.
    readOnly="true", color = "0,255,0",
tool.name="Equal", tool.description="Add Equal Operator", border.color
    = "0,0,0", border.width="2")
class Equal extends Binary, Relational {

}

@gmf.node( figure="ellipse", label="name", label.icon = "false", label.
    readOnly="true", color = "0,255,0",
tool.name="NotEqual", tool.description="Add Not Equal Operator", border
    .color = "0,0,0", border.width="2")
class NotEqual extends Binary, Relational {
}

```

B.2. Validación de modelos

```
@gmf.node(figure="ellipse", label="name", label.icon = "false", label.readOnly="true", color = "0,255,0", tool.name="LessThan", tool.description="Add Less Than Operator", border.color = "0,0,0", border.width="2")
class LessThan extends Binary, Relational {

}

@gmf.node(figure="ellipse", label="name", label.icon = "false", label.readOnly="true", color = "0,255,0", tool.name="GreaterThan", tool.description="Add Greater Than Operator", border.color = "0,0,0", border.width="2")
class GreaterThan extends Binary, Relational {

}

@gmf.node(figure="ellipse", label="name", label.icon = "false", label.readOnly="true", color = "0,255,0", tool.name="LessEqual", tool.description="Add Less Equal Operator", border.color = "0,0,0", border.width="2")
class LessEqual extends Binary, Relational {

}

@gmf.node(figure="ellipse", label="name", label.icon = "false", label.readOnly="true", color = "0,255,0", tool.name="GreaterEqual", tool.description="Add Greater Equal Operator", border.color = "0,0,0", border.width="2")
class GreaterEqual extends Binary, Relational {

}
```

B.2. Validación de modelos

A continuación se muestra el código fuente de la validación de los modelos, clasificado por tipo de restricción.

- **Restricciones generales.**

Listado B.20: Origen y destinos de Link diferentes

```
//Source and target must be different
context Link {

    constraint differentLink {
        check : self.source <> self.target
    }
}
```

B. Código Fuente

```

        message : 'Source and target must be different'
    }
}
}
```

Listado B.21: Como máximo un nodo puede ser apuntado por un Link

```

//1 inboundLink max.
context Node {

    constraint max1InboundLink {

        check : self.inboundLink.size() < 2

        message : self.eClass().name + ' must be 0 or 1 inboundLink
        .

    }
}
```

Listado B.22: Node permitidos fuera de Pattern

```

//Nodes outside the Pattern: Select , InsertInto , TimerWithin ,
TimerWithinMax.
context Root {

    constraint nodesOutPattern {

        check {
            var Nodes : Sequence = Node.allInstances();
            var sizeNodes : Integer = Nodes.size();
            var countNodes : Integer = 0;
            //If there is the Pattern
            if(Pattern.allInstances().size() > 0){
                //In every Pattern
                for(p : Pattern in Pattern.allInstances()){
                    var elements : OrderedSet = p.elements;
                    //In every Element of Pattern
                    for (e : Element in elements){
                        for(n : Node in Nodes){
                            //If it is inside of the Pattern
                            if(e.id == n.id){
                                countNodes = countNodes + 1;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

B.2. Validación de modelos

```

        }
    }
    for(n : Node in Nodes){
        if(n.isTypeOf(Pattern) or n.isTypeOf(Select) or
           n.isTypeOf(InsertInto) or n.isTypeOf(TimerWithin)
           or
           n.isTypeOf(TimerWithinMax)){
            countNodes = countNodes + 1;
        }
        if(n.isTypeOf(Select)){
            countNodes = countNodes + n.selectedAttributes.size
                ();
        }
        if(n.isTypeOf(EveryDistinct)){
            countNodes = countNodes + n.distinctValues.size();
        }
        if(n.isTypeOf(SecurityEvent)){
            countNodes = countNodes + n.conditions.size();
        }
    }
    if(countNodes <> sizeNodes){
        return false;
    } else{
        return true;
    }
}

message : 'Nodes outside the Pattern: Select , InsertInto ,
TimerWithin , TimerWithinMax. '
}
}

```

Listado B.23: Timestamp sólo dentro de Select

```

//Timestamp only inside Select
context Root {

    constraint timestampOnlySelect {
        check {
            var sizeTimestamp = Timestamp.allInstances().size();
            var count : Integer = 0;
            if(Select.allInstances().size() > 0){
                for(a : Attribute in Select.allInstances.at(0).
                    selectedAttributes){
                    if(a.isTypeOf(Timestamp)){
                        count = count + 1;
                    }
                }
            }
        }
    }
}

```

B. Código Fuente

```
        }
    }
    if(sizeTimestamp <> count){
        return false;
    }else{
        return true;
    }
}
message : 'Timestamp Attribute only must be inside Select.'
}
}
```

Listado B.24: Al menos un Pattern en el editor

```
//We can not work with an empty editor
context Root {

    constraint anElement {

        check : Pattern.allInstances().size() > 0

        message : 'A Pattern must be defined.'

    }
}
```

■ Restricciones que afectan a Pattern.

Listado B.25: Cada Pattern debe de tener un nombre

```
//Pattern has a name
context Pattern {

    constraint namePattern {

        check : self.name.isDefined()

        message : 'The name attribute of the pattern must have a
                  value. Unnamed '
                  + self.eClass().name + ' not allowed.'

        fix{
            title : "Set the name of " + self.eClass().name
            do{
                var defName : String;
                defName = UserInput.prompt("New name for " + self.
                                          eClass().name);
            }
        }
    }
}
```

```

        self.name = defName;
    }
}

}

```

Listado B.26: Un Node raíz dentro de Pattern

```

//Only one Node without inboundLink is allowed inside the Pattern
Clause
context Pattern {

    constraint oneInPattern {

        check {
            //Number of Node without inboundLink
            var numNode : Integer = 0;
            for(class in self.elements){
                if(class.inboundLink.size() == 0){
                    numNode = numNode + 1;
                }
            }
            if(numNode > 1){
                return false;
            }else{
                return true;
            }
        }

        message : 'Only one Node without inboundLink is allowed
inside the Pattern Clause.'
    }
}

```

- **Restricciones que afectan a Select.**

Listado B.27: Un Select debe ser definido

```

//A select must be defined
context Root {

    constraint anElement {

        check : Select.allInstances().size() > 0

        message : 'A Select must be defined.'
    }
}

```

B. Código Fuente

```
}
```

Listado B.28: Attribute dentro de Select deben tener distintos Alias

```
//Attributes inside of Select must have distinct alias
//if there are Alias empties don't be repeated
context Select {

    constraint attributesAliasSelect {
        check {
            var aliasAttributes : Set;
            var empties : Integer;
            var sizeAttributes : Integer;
            empties = 0;
            for (s : Attribute in self.selectedAttributes){
                aliasAttributes.add(s.Alias);
                //Count empties Alias because Alias empties are
                //repeated
                if (s.Alias.isUndefined()){
                    empties = empties + 1;
                }
            }
            //If we have more than one Alias empty, in the set will
            //be only one
            if (empties > 1){
                sizeAttributes = aliasAttributes.size() + (empties -
                    1);
            }
            else{
                sizeAttributes = aliasAttributes.size();
            }
            //Check the size
            if (sizeAttributes <> self.selectedAttributes.size()){
                return false;
            }
            else{
                return true;
            }
        }
        message : 'There are Alias repeated in the Select Clause.'
    }
}
```

Listado B.29: Sólo un Select está permitido en el editor

```
//Only one Select Clause in the editor
context Root {
```

```

constraint onlyOneSelect {
    check : Select.allInstances().size() <= 1
    message : 'Only one Select Clause in the editor is allowed.'
}
}

```

- **Restricciones que afectan a Property.**

Listado B.30: Property debe de tener un Link que le apunte

```

//Check links of the Properties
context Property {

    constraint checkLinksProperty {

        check {
            if (self.inboundLink.size() <> 1 and EveryDistinct.
                allInstances().size() == 0 and Select.allInstances().
                size() > 0 and
                Select.allInstances().at(0).selectedAttributes.isEmpty()
                ())
            return false;
        }
        else{
            var selfId : String = self.id;
            if(Select.allInstances().size() > 0){
                var compartSelect : OrderedSet = Select.
                    allInstances().at(0).selectedAttributes;
                for (a : Attribute in compartSelect){
                    if(a.id == selfId){
                        return true;
                    }
                }
            }
            if(EveryDistinct.allInstances().size() > 0){
                var compartED = EveryDistinct.allInstances().at(0).
                    distinctvalues;
                for (p : Property in compartED){
                    if(p.id == selfId){
                        return true;
                    }
                }
            }
            if(self.inboundLink.isEmpty()){
                return false;
            }
        }
    }
}

```

B. Código Fuente

```

        } else{
            return true;
        }
    }
    message : self.eClass().name + ' must be an inboundLink from
    relational operator. '
}
}

```

Listado B.31: Sólo un Link que le apunte está permitido como máximo

```

//Only one inboundLink in Property
context Property {

    constraint checkOneInboundLinkProperty {

        guard : self.satisfies("checkLinksProperty")
        check {
            var selfId : String = self.id;
            if(Select.allInstances().size() > 0){
                var compartSelect : OrderedSet = Select.
                    allInstances().at(0).selectedAttributes;
                for (a : Attribute in compartSelect){
                    if(a.id == selfId){
                        return true;
                    }
                }
            }
            if(EveryDistinct.allInstances().size() > 0){
                var compartED = EveryDistinct.allInstances().at(0).
                    distinctvalues;
                for (p : Property in compartED){
                    if(p.id == selfId){
                        return true;
                    }
                }
            }
            if(self.inboundLink.size() <> 1){
                return false;
            } else{
                return true;
            }
        }
        message : self.eClass().name + ' must have only one
        inboundLink.'
    }
}

```

```

    }
}
```

Listado B.32: Comprobación de Node que apuntan a Property

```
//Check inboundLink in Properties
context Property {

    constraint checkInboundLinkProperty {

        guard : self.satisfies("checkLinksProperty") and self.
            satisfies("checkOneInboundLinkProperty")
        check {
            var selfId : String = self.id;
            if(Select.allInstances().size() > 0){
                var compartSelect : OrderedSet = Select.allInstances().
                    at(0).selectedAttributes;
                for (a : Attribute in compartSelect){
                    if(a.id == selfId){
                        return true;
                    }
                }
            }
            if(EveryDistinct.allInstances().size() > 0){
                var compartED = EveryDistinct.allInstances().at(0).
                    distinctvalues;
                for (p : Property in compartED){
                    if(p.id == selfId){
                        return true;
                    }
                }
            }
            if(self.inboundLink.at(0).source.isKindOf(Relational)){
                return true;
            }else{
                return false;
            }
        }
        message : self.eClass().name + ' must have a Relational
        inboundLink.'
    }
}
```

Listado B.33: Comprobación del Event referenciado por Property

```
//Check referenced Event in Properties
```

B. Código Fuente

```
context Property {

    constraint checkReferencedEvent {

        check {
            var selfId : String = self.id;
            if(Select.allInstances().size() > 0){
                var compartSelect : OrderedSet = Select.allInstances()
                    .at(0).selectedAttributes;
                for(selects in Select.allInstances){
                    for(attributes in selects.selectedAttributes){
                        if(attributes.id == selfId){
                            if(attributes.referenced.isDefined() == false){
                                return false;
                            }
                        }
                    }
                }
            }
            if(EveryDistinct.allInstances().size() > 0){
                var compartED = EveryDistinct.allInstances().at(0).
                    distinctvalues;
                for(p : Property in compartED){
                    if(p.id == selfId){
                        if(p.referenced.isDefined() == false){
                            return false;
                        }
                    }
                }
            }
            if(Pattern.allInstances().size() > 0){
                for(patterns in Pattern.allInstances()){
                    for(element in patterns.elements){
                        if(element.id == selfId){
                            if(element.referenced.isDefined() == false){
                                return false;
                            }
                        }
                    }
                }
            }
            return true;
        }
        message : self.eClass().name + ' must have a referenced
Event.';
    }
}
```

}

Listado B.34: Property con order

```
// If Property has the order Attribute = 2 need an Event in the
// referenced attribute inside of SecurityEvent and is not a Value
// Attribute
context Property {

    constraint checkReferencedEventInside {

        check {
            var selfId : String = self.id;
            // If it is inside of SecurityEvent
            if(SecurityEvent.allInstances().size() > 0){
                for(class in SecurityEvent.allInstances()){
                    for(conditions in class.conditions){
                        if(conditions.id == selfId){
                            if(conditions.referenced.isDefined() == false
                                and self.order == 2 and
                                self.isTypeOf(Value) == false){
                                return false;
                            }
                        }
                    }
                }
            }
            return true;
        }
        message : self.eClass().name + ' must have a referenced
        Event (Attribute with order = 2).'

    }
}
}
```

- **Restricciones que afectan a SecurityValue.**

Listado B.35: El atributo Value tiene que tener un valor

```
// Check value in SecurityValue
context SecurityValue {

    constraint checkValue {

        check : self.value.isDefined()
        message : self.eClass().name + ' must have a value.'
        fix{
            title : "Set the value of " + self.eClass().name
        }
    }
}
```

B. Código Fuente

```
        do{
            var defValue : String;
            defValue = UserInput.prompt("New value for " + self.
                eClass().name);
            self.value = defValue;
        }
    }
}
```

Listado B.36: El atributo order tiene que ser

```
//Check order in SecurityValue
context SecurityValue {

    constraint checkOrderValue {

        check : self.order <> 1
        message : self.eClass().name + ' must not has order = 1.'
        fix{
            title : "Set the value of " + self.eClass().name
            do{
                var defOrder : String;
                defValue = UserInput.prompt("New order for " + self.
                    eClass().name);
                self.order = defOrder;
            }
        }
    }
}
```

■ Restricciones que afectan a Binary.

Listado B.37: Los operadores Binary tienen que apuntar a 2 Node destino

```
//Check links in Binary Operators
context Binary {

    constraint checkLinksBinary {

        check {
            if (self.outboundLink.size() <> 2){
                return false;
            }
            return true;
        }
        message : self.eClass().name + ' operator must have 2
            outbound links .'
    }
}
```

```

    }
}
```

- Listado B.38: Los Node a los que apuntan los operadores Binary tienen que tener los atributos order

```
//Check orders in Binary Operators sources
context Binary {

    constraint checkOrdersBinary {

        guard : self.satisfies("checkLinksBinary") and
            self.outboundLink.at(0).satisfies("relationalLink") and
            self.outboundLink.at(1).satisfies("relationalLink")
        check {
            if((self.outboundLink.at(0).target.order == 1 and
                self.outboundLink.at(1).target.order == 2) or
                (self.outboundLink.at(0).target.order == 2 and
                self.outboundLink.at(1).target.order == 1)){
                return true;
            }else{
                return false;
            }
        }
        message : self.eClass().name + ' operator must have targets
            with order 1 and order 2.'
    }
}
```

▪ Restricciones que afectan a Nary.

- Listado B.39: Los Node a los que apuntan los operadores Nary tienen que tener los atributos order

```
//Check orders in Nary Operators sources
context Nary {

    constraint checkOrdersNary {

        guard : self.satisfies("checkLinksNary")
        check {
            var sizeNary : Integer = self.outboundLink.size();
            var nodesTarget : Sequence;
            for(1 : Link in self.outboundLink){
                nodesTarget.add(1.target.order);
            }
        }
    }
}
```

B. Código Fuente

```
        }
        for(i in Sequence{1..sizeNary}){
            if(nodesTarget.excludes(i)){
                return false;
            }
        }
        return true;
    }
    message : self.eClass().name + ' operator must have targets
    with the value in the order attribute 1, 2...N.'
}
}
```

Listado B.40: Los operadores Nary tienen que apuntar al menos 2 Node destino

```
//Check links in Nary Operators
context Nary {

    constraint checkLinksNary {

        check : self.outboundLink.size() >= 2
        message : self.eClass().name + ' operator must have at least
        2 outbound links.'

    }
}
```

- **Restricciones que afectan a los operadores Unary.**

Listado B.41: Los operadores Unary tienen que apuntar a 1 Node destino

```
//Check links in Unary Operators
context Unary {

    constraint checkLinksUnary {

        check : self.outboundLink.size() == 1
        message : self.eClass().name + ' operator must have 1
        outbound link.'

    }
}
```

- **Restricciones que afectan a los operadores Relational.**

B.2. Validación de modelos

Listado B.42: Los operadores Relational tienen que apuntar a Attribute no Element

```
//A relational operator only has Attribute Object, not Element
context Link {

    constraint relationalLink {

        check {
            if (self.source.isKindOf(Relational)){
                return (self.source.isKindOf(Relational) and self.
                    target.isKindOf(Attribute));
            }
            else{ return true; }
        }
        message : 'Relational operators targets must have Attribute
(IP, MAC, Port, Value), not Element'
    }
}
```

Listado B.43: Sólo se puede apuntar como máximo a un SecurityValue

```
//Check targets in relational operators must not has more
//than one SecurityValue, only one is allowed
context Relational{

    constraint relationalSecurityValues {

        guard : self.satisfies("checkLinksBinary")

        check {
            if(self.outboundLink.at(0).target.isTypeOf(SecurityValue)
                and
                self.outboundLink.at(1).target.isTypeOf(SecurityValue)){
                    return false;
                }else{
                    return true;
                }
        }
        message : 'Two Values in ' + self.eClass().name + ' is not
allowed.'

    }
}
```

Listado B.44: Los Node a los que apuntan tienen que ser del mismo tipo o con un
SecurityValue

B. Código Fuente

```
//Targets of Relational must be of the same type, or one type with
// a value.
context Relational{

    constraint relationalSecurityValues {

        guard : self.satisfies("checkLinksBinary") and self.satisfies
            ("relationalSecurityValues")

        check : ((self.outboundLink.at(0).target.isTypeOf(IP) and
            self.outboundLink.at(1).target.isTypeOf(Port)) == false)
            and
            ((self.outboundLink.at(1).target.isTypeOf(IP) and
            self.outboundLink.at(0).target.isTypeOf(Port)) == false)
            and
            ((self.outboundLink.at(0).target.isTypeOf(IP) and
            self.outboundLink.at(1).target.isTypeOf(MAC)) == false)
            and
            ((self.outboundLink.at(1).target.isTypeOf(IP) and
            self.outboundLink.at(0).target.isTypeOf(MAC)) == false)
            and
            ((self.outboundLink.at(0).target.isTypeOf(MAC) and
            self.outboundLink.at(1).target.isTypeOf(Port)) == false)
            and
            ((self.outboundLink.at(1).target.isTypeOf(MAC) and
            self.outboundLink.at(0).target.isTypeOf(Port)) == false)
        message : 'The targets of ' + self.eClass().name + ' must be
            of the same type, or one type with a value.'

    }
}
}
```

Listado B.45: Relational dentro de SecurityEvent si no precede un While

```
// Relational Operators must be inside SecurityEvents (if do not
// have
//an inboundLink While)
context Relational {

    constraint relationalInSecurityEvent {

        check {
            //If is inside of a SecurityEvent, we do not evaluate the
            constraint
            var selfId : String = self.id;
            var flag : Integer = 0;
            //If there are SecurityEvents
            if(SecurityEvent.allInstances().size() > 0){
```

```

//In every SecurityEvent
for(s : SecurityEvent in SecurityEvent.allInstances())
{
    var conditions : OrderedSet = s.conditions;
    //In every Element of SecurityEvent
    for (class in conditions){
        //If it is inside of the SecurityEvent , we do
        //not evaluate
        if(class.id == selfId){
            flag = 1;
        }
    }
}
//Inside of SecurityEvent
if(flag == 1){
    if(self.inboundLink.size == 0){
        return true;
    } else{
        if(self.inboundLink.at(0).source.isKindOf(Logical)
            == false){
            return false;
        } else{
            return true;
        }
    }
}
//Outside of SecurityEvent
} else{
    //If there is an inboundLink and it is While , return
    //true
    if(self.inboundLink.size() > 0){
        if(self.inboundLink.at(0).source.isTypeOf(While)){
            return true;
        } else{
            return false;
        }
    } else{
        return false;
    }
}
}

message : 'Relational Operators must be inside SecurityEvent
or outside if there is a While operator in the
inboundLink .'

}
}

```

B. Código Fuente

■ Restricciones que afectan a InsertInto.

Listado B.46: El atributo NewStreamName tiene que tener un valor

```
//InsertInto has a NewStreamName
context InsertInto {

    constraint nameInsertInto {

        check : self.newStreamName.isDefined()

        message : 'The NewStreamName attribute of the InsertInto must
                   have a value.'

        fix{
            title : "Set the name of " + self.eClass().name
            do{
                var defName : String;
                defName = UserInput.prompt("New NewStreamName for " +
                                         self.eClass().name);
                self.newStreamName = defName;
            }
        }
    }
}
```

Listado B.47: Sólo un InsertInto está permitido en el editor

```
//Only one InsertInto Clause in the editor
context Root {

    constraint onlyOneInsertInto {
        check : InsertInto.allInstances().size() <= 1

        message : 'Only one InsertInto Clause in the editor is
                   allowed.'

    }
}
```

■ Restricciones que afectan a operadores Logical.

Listado B.48: And u Or fuera de SecurityEvent apuntan a 2 Node

B.2. Validación de modelos

```
//And and Or are Binary operators outside of SecurityEvent
context Logical {

    constraint logicalBinary {

        check {
            //If is the not operator, we do not evaluate the
            constraint
            if(self.isTypeOf(Not)){
                return true;
            }
            //If is inside of a SecurityEvent, we do not evaluate the
            constraint
            var selfId : String = self.id;
            //If there are SecurityEvents
            if(SecurityEvent.allInstances().size() > 0){
                //In every SecurityEvent
                for(s : SecurityEvent in SecurityEvent.allInstances()){
                    var conditions : OrderedSet = s.conditions;
                    //In every Element of SecurityEvent
                    for (class in conditions){
                        //If is inside of the SecurityEvent, we do not
                        evaluate
                        if(class.id == selfId){
                            return true;
                        }
                    }
                }
                if(self.outboundLink.size() <> 2){
                    return false;
                }
            }
            return true;
        }

        message : self.eClass().name + ' operator outside of
        SecurityEvent, must have two targets.'
    }
}
}
```

Listado B.49: Node a los que apuntan And y Or fuera de SecurityEvent

```
//Targets in And, Or must be: followedBy, SecurityEvent, Logical
Operators
//or timer:interval (outside SecurityEvent)
context Logical {
```

B. Código Fuente

```
constraint targetsAndOrOutEvent {
    guard {
        //If is not, we do not evaluate the constraint
        if( self.isTypeOf(Not)){
            return true;
        } else{
            if( self.satisfies("logicalBinary") and self.
                satisfies("checkOrdersNary")){
                return true;
            }
            else{
                return false;
            }
        }
    }
    check {
        //If is not, we do not evaluate the constraint
        if( self.isTypeOf(Not)){
            return true;
        }
        //If is inside of a SecurityEvent , we do not evaluate the
        constraint
        var selfId : String = self.id;
        //If there are SecurityEvents
        if(SecurityEvent.allInstances().size() > 0){
            //In every SecurityEvent
            for(s : SecurityEvent in SecurityEvent.allInstances()){
                var conditions : OrderedSet = s.conditions;
                //In every Element of SecurityEvent
                for (class in conditions){
                    //If is inside of the SecurityEvent , we do not
                    evaluate
                    if(class.id == selfId){
                        return true;
                    }
                }
            }
        }
        //The first target
        if(( self.outboundLink.at(0).target.isTypeOf(FollowedBy)
            or
            self.outboundLink.at(0).target.isTypeOf(SecurityEvent) or
            self.outboundLink.at(0).target.isKindOf(Logical) or
            self.outboundLink.at(0).target.isTypeOf(TimerInterval))
            == false){
            return false;
        }
        //The second target
        if(( self.outboundLink.at(1).target.isTypeOf(FollowedBy)
```

B.2. Validación de modelos

```

        or
self.outboundLink.at(1).target.isTypeOf(SecurityEvent) or
self.outboundLink.at(1).target.isKindOf(Logical) or
self.outboundLink.at(1).target.isTypeOf(TimerInterval))
    == false){
    return false;
}
return true;
}

message : 'Targets of ' + self.eClass().name + ' operator (
    outside of SecurityEvent), must be Followed By,
    SecurityEvent, Logical Operators or Timer Interval.'

}
}

```

Listado B.50: Node a los que apuntan And y Or dentro de SecurityEvent

```

//Targets of And, Or operators must be: Relational operators or
Logical Operators
//(inside SecurityEvent)
context Logical {

constraint targetsLogicalInEvent {
    guard {
        //If is not, we do not evaluate the constraint
        if(self.isTypeOf(Not)){
            return true;
        }else{
            if(self.satisfies("checkLinksNary") and self.
                satisfies("checkOrdersNary")){
                return true;
            }
            else{
                return false;
            }
        }
    }
    check {
        //If is not, we do not evaluate the constraint
        if(self.isTypeOf(Not)){
            return true;
        }
        //If is outside of a SecurityEvent, we do not evaluate
        //the constraint
        var selfId : String = self.id;
        var flag : Integer = 0;
    }
}

```

B. Código Fuente

```

//If there are SecurityEvents
if(SecurityEvent.allInstances().size() > 0){
    //In every SecurityEvent
    for(s : SecurityEvent in SecurityEvent.allInstances()){
        var conditions : OrderedSet = s.conditions;
        //In every Element of SecurityEvent
        for (class in conditions){
            //If is inside of the SecurityEvent , flag = 1
            if(class.id == selfId){
                flag = 1;
            }
        }
    }
    //If is outside the SecurityEvents , we do not evaluate
    if(flag == 0){
        return true;
    }
    //Check every target
    for(l : Link in self.outboundLink){
        if((l.target.isKindOf(Relational) or l.target.isKindOf
            (Logical)) == false){
            return false;
        }
    }
    return true;
}

message : 'Targets of ' + self.eClass().name + ' operator (
    inside of SecurityEvent) , must be Relational Operators
    or Logical Operators.'

}
}

```

Listado B.51: Node a los que apunta Not fuera de SecurityEvent

```

//Targets in Not must be: followedBy , SecurityEvent , Logical
    Operators or
//timer:interval (outside SecurityEvent)
context Not {

    constraint targetNotOutEvent {
        guard : self.satisfies("checkLinksUnary")
        check {
            //If is inside of a SecurityEvent , we do not evaluate the
            constraint
            var selfId : String = self.id;
        }
    }
}

```

B.2. Validación de modelos

```

//If there are SecurityEvents
if(SecurityEvent.allInstances().size() > 0){
    //In every SecurityEvent
    for(s : SecurityEvent in SecurityEvent.allInstances()){
        var conditions : OrderedSet = s.conditions;
        //In every Element of SecurityEvent
        for (e : Element in conditions){
            //If is inside of the SecurityEvent, we do not
            //evaluate
            if(class.id == selfId){
                return true;
            }
        }
    }
    //The target
    if((self.outboundLink.at(0).target.isTypeOf(FollowedBy)
        or
        self.outboundLink.at(0).target.isTypeOf(SecurityEvent) or
        self.outboundLink.at(0).target.isKindOf(Logical) or
        self.outboundLink.at(0).target.isTypeOf(TimerInterval))
        == false){
        return false;
    }
    return true;
}

message : 'Target of ' + self.eClass().name + ' operator (
    outside of SecurityEvent), must be: Followed By,
    SecurityEvent, Logical Operators, Timer Interval (
    outside SecurityEvent).'

}
}

```

Listado B.52: Node a los que apunta Not dentro de SecurityEvent

```

//Target of Not operators must be: Relational operators or Logical
//Operators
//(inside SecurityEvent)
context Not {

    constraint targetsLogicalInEvent {
        guard : self.satisfies("checkLinksUnary")

        check {
            //If is outside of a SecurityEvent, we do not evaluate
            //the constraint

```

B. Código Fuente

```
var selfId : String = self.id;
var flag : Integer = 0;
//If there are SecurityEvents
if(SecurityEvent.allInstances().size() > 0){
    //In every SecurityEvent
    for(s : SecurityEvent in SecurityEvent.allInstances()){
        var conditions : OrderedSet = s.conditions;
        //In every Element of SecurityEvent
        for (class in conditions){
            //If is inside of the SecurityEvent , flag = 1
            if(class.id == selfId){
                flag = 1;
            }
        }
    }
    //If is outside the SecurityEvents , we do not evaluate
    if(flag == 0){
        return true;
    }
    //Check target
    if((self.outboundLink.at(0).target.isKindOf(Relational)
        or
        self.outboundLink.at(0).target.isKindOf(Logical)) ==
        false){
        return false;
    }
    return true;
}

message : 'Target of ' + self.eClass().name + ' operator (
    inside of SecurityEvent) , must be Relational Operators
    or Logical Operators.'

}
}
```

■ Restricciones que afectan a SecurityEvent.

Listado B.53: Los elementos permitidos dentro de SecurityEvent son operadores Relational - operadores Logical y Attribute

```
//Inside Event only must be Relational Operators , Logical
    Operators and Attributes
context SecurityEvent {

    constraint insideSecurityEvent {
```

B.2. Validación de modelos

```
check {
    var conditions : OrderedSet = self.conditions;
    for(e : Element in conditions){
        if((e.isKindOf(Relational) or
            e.isKindOf(Logical) or
            e.isKindOf(Attribute)) == false){
            return false;
        }
    }
    return true;
}

message : 'One of the elements inside of the SecurityEvent
must be outside. Inside only is allowed Relational Op.,
Logical Op. and Attributes.'

}
```

Listado B.54: El atributo name tiene que tener un valor

```
//SecurityEvent must have a name
context SecurityEvent {

    constraint nameSecurityEvent {

        check : self.name.isDefined()

        message : 'The name attribute of the SecurityEvent must have
                   a value. Unnamed '
                   + self.eClass().name + ' not allowed.'

        fix{
            title : "Set the name of " + self.eClass().name
            do{
                var defName : String;
                defName = UserInput.prompt("New name for " + self.
                                           eClass().name);
                self.name = defName;
            }
        }
    }
}
```

Listado B.55: Sólo un Node sin que le apunte ningún nodo está permitido dentro de SecurityEvent

B. Código Fuente

```
//Only one Node without inboundLink is allowed inside
//SecurityEvent
context SecurityEvent {

    constraint oneInSecurityEvent {

        check {
            //Number of Node without inboundLink
            var numNode : Integer = 0;
            for(class in self.conditions){
                if(class.inboundLink.size() == 0){
                    numNode = numNode + 1;
                }
            }
            if(numNode > 1){
                return false;
            } else{
                return true;
            }
        }

        message : 'Only one Node without inboundLink is allowed
                   inside SecurityEvent (only a Logical Operator or
                   Relational Operator).'

    }
}
```

▪ Restricciones que afectan a TimerInterval.

Listado B.56: Los Node que le apuntan tienen que ser o Every u operadores Logical

```
//Timer Interval must have inboundLink from Every , or Logical
// Operators
context TimerInterval {

    constraint timerIntervalInbound {

        check {
            //If we have an inboundLink
            if(self.inboundLink.size() == 1){
                // If inboundLink is Every or Logical
                if(self.inboundLink.at(0).source.isTypeOf(Every) or
                   self.inboundLink.at(0).source.isKindOf(Logical)){
                    return true;
                } else{
                    return false;
                }
            } else{

```

```

        return false;
    }
}

message : 'Timer Interval must have 1 inboundLink from Every
or Logical operators.'

}
}

```

Listado B.57: Sólo un atributo tiene que ser mayor que 0

```

//Timer Interval must have only 1 Attribute > 0
context TimerInterval {

constraint timerIntervalGreater0 {

check {
    var count : Integer = 0;
    if( self.years > 0){
        count = count + 1;
    }
    if(self.months > 0){
        count = count + 1;
    }
    if(self.weeks > 0){
        count = count + 1;
    }
    if(self.days > 0){
        count = count + 1;
    }
    if(self.hours > 0){
        count = count + 1;
    }
    if(self.minutes > 0){
        count = count + 1;
    }
    if(self.seconds > 0){
        count = count + 1;
    }
    if(self.milliseconds > 0){
        count = count + 1;
    }
    if(count == 1){
        return true;
    } else{
        return false;
    }
}
}

```

B. Código Fuente

```
        }

        message : 'Timer Interval must have 1 Attribute > 0.'

    }

}
```

■ Restricciones que afectan a TimerAt.

Listado B.58: Los Node que le apuntan tienen que ser Every

```
//Timer At must have inboundLink from Every
context TimerAt {

    constraint timerAtInbound {

        check {
            //If we have an inboundLink
            if( self.inboundLink.size() == 1){
                //If inboundLink is Every
                if( self.inboundLink.at(0).source.isTypeOf(Every)){
                    return true;
                } else{
                    return false;
                }
            } else{
                return false;
            }
        }

        message : 'Timer At must have 1 inboundLink from Every
operator.'

    }
}
```

■ Restricciones que afectan a TimerWithin.

Listado B.59: Debe de estar fuera de Pattern

```
//Timer Within must be outside Pattern
context TimerWithin {

    constraint timerWithinOutPattern {

        check {
            var selfId : String = self.id;
```

B.2. Validación de modelos

```
//If there is the Pattern
if(Pattern.allInstances().size() > 0){
    //In every Pattern
    for(p : Pattern in Pattern.allInstances()){
        var elements : OrderedSet = p.elements;
        //In every Element of Pattern
        for (class in elements){
            //If it is inside of the Pattern, return false
            if(class.id == selfId){
                return false;
            }
        }
    }
    return true;
}

message : 'Timer Within must be outside Pattern.'

}
}
```

Listado B.60: Sólo un atributo tiene que ser mayor que 0

```
//Timer Within must have only 1 Attribute > 0
context TimerWithin {

    constraint timerWithinGreater0 {

        check {
            var count : Integer = 0;
            if(self.years > 0){
                count = count + 1;
            }
            if(self.months > 0){
                count = count + 1;
            }
            if(self.weeks > 0){
                count = count + 1;
            }
            if(self.days > 0){
                count = count + 1;
            }
            if(self.hours > 0){
                count = count + 1;
            }
            if(self.minutes > 0){
                count = count + 1;
            }
        }
    }
}
```

B. Código Fuente

```
        }
        if( self.seconds > 0){
            count = count + 1;
        }
        if( self.milliseconds > 0){
            count = count + 1;
        }
        if(count == 1){
            return true;
        } else{
            return false;
        }
    }

    message : 'Timer Within must have 1 Attribute > 0.'
}

}
```

■ Restricciones que afectan a TimerWithinMax.

Listado B.61: Debe de estar fuera de Pattern

```
//Timer Within Max must be outside Pattern
context TimerWithinMax {

    constraint timerWithinMaxOutPattern {

        check {
            var selfId : String = self.id;
            //If there is the Pattern
            if(Pattern.allInstances().size() > 0){
                //In every Pattern
                for(p : Pattern in Pattern.allInstances()){
                    var elements : OrderedSet = p.elements;
                    //In every Element of Pattern
                    for (class in elements){
                        //If it is inside of the Pattern , return false
                        if(class.id == selfId){
                            return false;
                        }
                    }
                }
            }
            return true;
        }

        message : 'Timer Within Max must be outside Pattern.'
    }
}
```

```

    }
}
```

Listado B.62: Sólo un atributo tiene que ser mayor que 0

```
//Timer Within Max must have only 1 Attribute > 0 (aside from
maxCount).
context TimerWithinMax {
    constraint timerWithinMaxGreater0 {
        check {
            var count : Integer = 0;
            if( self.years > 0){
                count = count + 1;
            }
            if( self.months > 0){
                count = count + 1;
            }
            if( self.weeks > 0){
                count = count + 1;
            }
            if( self.days > 0){
                count = count + 1;
            }
            if( self.hours > 0){
                count = count + 1;
            }
            if( self.minutes > 0){
                count = count + 1;
            }
            if( self.seconds > 0){
                count = count + 1;
            }
            if( self.milliseconds > 0){
                count = count + 1;
            }
            if( count == 1){
                return true;
            } else{
                return false;
            }
        }
        message : 'Timer Within Max must have 1 Attribute > 0 (aside
from maxCount).'
    }
}
```

B. Código Fuente

```
}
```

Listado B.63: El atributo maxCount tiene que ser mayor que 0

```
//Attribute maxCount > 0
context TimerWithinMax {

    constraint timerWithinMaxCount {

        check : self.maxCount > 0

        message : 'Timer Within Max: Attribute maxCount must be >
                   0.'

    }
}
```

Listado B.64: Sólo un TimerWithin o un TimerWithinMax está permitido en el editor

```
//Only one Timer Within or Timer Within Max is allowed
context Root {

    constraint onlyOneTimer {

        check {
            var count : Integer = 0;
            count = count + TimerWithin.allInstances().size();
            count = count + TimerWithinMax.allInstances().size();
            if(count > 1){
                return false;
            } else{
                return true;
            }
        }

        message : 'Only one Timer Within or Timer Within Max is
                   allowed.'

    }
}
```

- **Restricciones que afectan a FollowedBy.**

B.2. Validación de modelos

Listado B.65: Los Node a los que apunta FollowedBy son SecurityEvent - Every - EveryDistinct - Num o operadores Logical

```
//Followed By must have outboundLink to SecurityEvent , Every ,
Every-distinct ,
//Num or Logical Operators
context FollowedBy {

    constraint followedByOutbound {
        guard : self . satisfies ("checkOrdersNary ")
        check {
            //Check every target
            for(l : Link in self.outboundLink){
                if((l.target.isTypeOf(SecurityEvent) or l.target .
                    isKindOf(Logical) or
                    l.target.isTypeOf(Every) or l.target.isTypeOf(
                        EveryDistinct) or
                    l.target.isTypeOf(Num)) == false ){
                    return false ;
                }
            }
            return true ;
        }

        message : 'Followed By must have outboundLink to
                    SecurityEvent , Every , Every-distinct , Num or Logical
                    Operators ,
    }

}
```

- **Restricciones que afectan a Every.**

Listado B.66: Los Node a los que apunta Every tienen que ser SecurityEvent - FollowedBy - TimerInterval - TimerAt o Num

```
//Every operator must have outboundLink to SecurityEvent , Followed
By, Timer Interval ,
//Timer At or Num
context Every {

    constraint everyOutbound {
        guard : self . satisfies ("checkLinksUnary ")
        check {
            //Check every target
            for(l : Link in self.outboundLink){
                if((l.target.isTypeOf(SecurityEvent) or l.target .
                    isTypeOf(FollowedBy) or
```

B. Código Fuente

```

        l.target.isTypeOf(TimerInterval) or l.target.
            isTypeOf(TimerAt) or
            l.target.isTypeOf(Num)) == false){
                return false;
            }
        }
        return true;
    }

    message : 'Every operator must have outboundLink to
        SecurityEvent , Followed By, Timer Interval , Timer At or
        Num'

}
}

```

- Restricciones que afectan a EveryDistinct.

Listado B.67: Los Node a los que apunta EveryDistinct tienen que ser SecurityEvent
- FollowedBy - TimerInterval - TimerAt o Num

```

//Every Distinct operator must have outboundLink to SecurityEvent ,
//Followed By, Timer Interval ,
//Timer At or Num
context EveryDistinct {

    constraint everyDistinctOutbound {
        guard : self.satisfies("checkLinksUnary")
        check {
            //Check every target
            for(l : Link in self.outboundLink){
                if((l.target.isTypeOf(SecurityEvent) or l.target.
                    isTypeOf(FollowedBy) or
                    l.target.isTypeOf(TimerInterval) or l.target.
                    isTypeOf(TimerAt) or
                    l.target.isTypeOf(Num)) == false){
                        return false;
                    }
                }
            }
            return true;
        }

        message : 'Every Distinct operator must have outboundLink to
            SecurityEvent , Followed By, Timer Interval , Timer At or
            Num'

    }
}

```

| }

Listado B.68: Al menos debe contener a un Attribute

```
//Every Distinct has inside at least 1 Attribute
context EveryDistinct {

    constraint everyDistinctInside {

        check : self.distinctValues.size() > 0

        message : 'Every Distinct has inside at least 1 Attribute.'

    }

}
```

■ Restricciones que afectan a Num.

Listado B.69: Los Node a los que apunta Num tienen que ser SecurityEvent - FollowedBy - And u Or

```
//Num operator must have outboundLink to SecurityEvent , Followed
    By, And,
//or Or
context Num {

    constraint NumOutbound {
        guard : self.satisfies("checkLinksUnary")
        check {
            //Check every target
            for(l : Link in self.outboundLink){
                if((l.target.isTypeOf(SecurityEvent) or l.target.
                    isTypeOf(FollowedBy) or
                    l.target.isTypeOf(And) or l.target.isTypeOf(Or)) ==
                    false){
                    return false;
                }
            }
            return true;
        }

        message : 'Num operator must have outboundLink to
            SecurityEvent , Followed By, And, or Or'
    }
}
```

B. Código Fuente

▪ Restricciones que afectan a Range.

Listado B.70: Los Node a los que apunta Num tienen que ser SecurityEvent - FollowedBy - And u Or

```
//Range operator must have outboundLink to SecurityEvent , Followed  
By, And,  
//or Or  
context Range {  
  
    constraint RangeOutbound {  
        guard : self.satisfies("checkLinksUnary")  
        check {  
            //Check every target  
            for(l : Link in self.outboundLink){  
                if((l.target.isTypeOf(SecurityEvent) or l.target.  
                    isTypeOf(FollowedBy) or  
                    l.target.isTypeOf(And) or l.target.isTypeOf(Or)) ==  
                        false){  
                    return false;  
                }  
            }  
            return true;  
        }  
    }  
  
    message : 'Range operator must have outboundLink to  
        SecurityEvent , Followed By, And, or Or'  
}  
}
```

Listado B.71: El Node que apunta a Range tiene que ser Until

```
//Range operator must have inboundLink from Until  
context Range {  
  
    constraint RangeInbound {  
  
        check {  
            //Check every target  
            if( self.inboundLink.size() == 1){  
                if( self.inboundLink.at(0).source.isTypeOf(Until) ==  
                    false){  
                    return false;  
                } else{  
                    return true;  
                }  
            } else{  
                return false;  
            }  
        }  
    }  
}
```

```

        }
    }

    message : 'Range operator must have inboundLink from Until'
}

}

```

Listado B.72: Los atributos lowEndPoint y/o highEndPoint tienen que ser mayores que 0

```

//lowEndpoint or/and highEndpoint > 0
context Range {

    constraint RangePoint {

        check : self.lowEndpoint > 0 or self.highEndpoint > 0

        message : 'lowEndpoint or/and highEndpoint > 0'

    }
}

```

Listado B.73: El atributo highEndpoint tiene que ser mayor o igual que lowEndpoint

```

//highEndpoint must be >= lowEndpoint
context Range {

    constraint RangeLowHigh {

        check : self.highEndpoint >= self.lowEndpoint

        message : 'highEndpoint must be >= lowEndpoint'

    }
}

```

- **Restricciones que afectan a While.**

Listado B.74: Destinos While expresión izquierda

```

//While operator must have outboundLink (order 1) to SecurityEvent
, Every, Every Distinct
context While {
    guard : self.satisfies("checkOrdersBinary")
    constraint whileOut1 {

```

B. Código Fuente

```
    check {
        var whileOut = self.outboundLink;
        for(l : Link in whileOut){
            if(l.target.order = 1){
                if((l.target.isTypeOf(SecurityEvent) or l.target.
                    isTypeOf(Every) or
                    l.target.isTypeOf(EveryDistinct)) == false){
                    return false;
                }
            }
        }
        return true;
    }

    message : 'While operator must have outboundLink (order 1)
        to SecurityEvent , Every , Every Distinct .'

}
}
```

Listado B.75: Destinos While expresión derecha

```
//While operator must have outboundLink (order 2) to Relational
Operators
context While {
    guard : self.satisfies("checkOrdersBinary")
    constraint whileOut2 {

        check {
            var whileOut = self.outboundLink;
            for(l : Link in whileOut){
                if(l.target.order = 2){
                    if(l.target.isKindOf(Relational)== false){
                        return false;
                    }
                }
            }
        }
        return true;
    }

    message : 'While operator must have outboundLink (order 2)
        to Relational Operator .'

}
}
```

- **Restricciones que afectan a Until.**

Listado B.76: Destinos Until expresión izquierda

```
//Until operator must have outboundLink (order 1) to Range
Operator.
context Until {
    guard : self.satisfies("checkOrdersBinary")
    constraint UntilOut1 {

        check {
            var whileOut = self.outboundLink;
            for(l : Link in whileOut){
                if(l.target.order == 1){
                    if(l.target.isTypeOf(Range) == false){
                        return false;
                    }
                }
            }
            return true;
        }

        message : 'Until operator must have outboundLink (order 1)
        to Range Operator.'
    }
}
```

Listado B.77: Destinos Until expresión derecha

```
//Until operator must have outboundLink (order 2) to Event, And,
Or or Timer Interval.
context Until {
    guard : self.satisfies("checkOrdersBinary")
    constraint UntilOut2 {

        check {
            var whileOut = self.outboundLink;
            for(l : Link in whileOut){
                if(l.target.order == 2){
                    if((l.target.isTypeOf(SecurityEvent) or l.target.
                        isKindOf(And) or
                        l.target.isKindOf(Or) or l.target.isKindOf(
                        TimerInterval)) == false){
                        return false;
                    }
                }
            }
            return true;
        }

    }
}
```

B. Código Fuente

```
    }

    message : 'Until operator must have outboundLink (order 2)
              to Event, And, Or or Timer Interval.'

}
```

B.3. Transformación Model-to-Text

B.3.1. Fichero EGL

El fichero completo se puede visualizar en el Listado B.78.

Listado B.78: Código M2T

```
1 String [%]
2   var p : Pattern = getPattern();
3   %][%==p.name%] = "@Name('[%==p.name%]')"
4 [% if (InsertInto.allInstances().size() > 0) { %
5 insert into [%]
6   var i : InsertInto = getInsertInto();
7   %][%==i.newStreamName%]
8 [% } %
9 select[%
10  var all : String;
11  var Properties : OrderedSet = getSelectAttributes();
12  var sizeAttributes : Integer = Properties.size();
13  var count : Integer = 1;
14  if (sizeAttributes == 0){
15    all = "*";%] [%=all%]
16  [%}else{
17    var t : Integer;
18    for(a in Properties){
19      t = typeSelectAttributes(a);
20      //Not the last
21      if (count <> sizeAttributes){%
22        [% if (t == 1) {
23          if(a.referenced.Alias == null){%] [%=a.eClass().
24              name%][%==a.type%],
25          [% }else{%
26            [%=a.referenced.Alias%].[%=a.eClass().name
27              %][%==a.type%],[%}
28        }
29        if (t == 2) {
30          if(a.referenced.Alias == null){%] [%=a.eClass().
31              name%][%==a.type%] as [%=a.Alias%],
```

B.3. Transformación Model-to-Text

```

28      [% } else{[%]    [%=a . referenced . Alias %].[%=a . eClass () . name
29          %][%a . type %] as [%=a . Alias %],[%]
30  }
31      if (t == 3) {
32          if(a . referenced . Alias == null){[%]    [%=a . eClass () .
33              name %],
34          [% } else{[%]    [%=a . referenced . Alias %].[%=a . eClass () . name
35              %], [%}
36      }
37      count = count + 1;
38  } else{[%]
39      [% if (t == 1) {
40          if(a . referenced . Alias == null){[%]    [%=a . eClass () .
41              name %][%a . type %]
42          [% } else{[%]    [%=a . referenced . Alias %].[%=a . eClass () . name
43              %][%a . type %]
44          [%}
45      }
46      if (t == 2) {
47          if(a . referenced . Alias == null){[%]    [%=a . eClass () .
48              name %][%a . type %] as [%=a . Alias %]
49          [% } else{[%]    [%=a . referenced . Alias %].[%=a . eClass () . name
50              %][%a . type %] as [%=a . Alias %]
51          [%}
52      }
53      count = count + 1;
54  }
55  }
56 from pattern [ [% var p = getRoot();%][%=traversal(p)%]
57 [% if (TimerWithin . allInstances () . size () > 0){%]
58 where timer : within([%= getAttributeTimerGuard(TimerWithin . allInstances
59     () . at(0)) %]) [% } %]
60 [% if (TimerWithinMax . allInstances () . size () > 0){%]
61 where timer : withinmax([%= getAttributeTimerGuard(TimerWithinMax .
62     allInstances () . at(0)) %], [%=TimerWithinMax . allInstances () . at(0) .
63     maxCount %]) [% } %]];
64 [% operation getSelectAttributes() : OrderedSet {
65     var s : new Sequence;
66     for (class in Select . allInstances ()) {
67         s.add(class);

```

B. Código Fuente

```
65      }
66      var Properties = s.at(0).selectedAttributes;
67      return Properties;
68  }
69  operation typeSelectAttributes(a){
70      if (a.type().name <> 'Timestamp' and a.Alias.isUndefined()) {
71          return 1;
72      }
73      if (a.type().name <> 'Timestamp' and a.Alias.isDefined()) {
74          return 2;
75      }
76      if (a.type().name == 'Timestamp') {
77          return 3;
78      }
79  }
80  operation getPattern() : Pattern{
81      var patterns : new Sequence;
82      for (class in Pattern.allInstances()){
83          patterns.add(class);
84      }
85      return patterns.at(0);
86  }
87  operation getInsertInto() : InsertInto{
88      var i : new Sequence;
89      for (class in InsertInto.allInstances()){
90          i.add(class);
91      }
92      return i.at(0);
93  }
94  operation getPatternElements(p : Pattern) : Sequence{
95      var i : new Sequence;
96      for (class in getPattern().elements){
97          i.add(class);
98      }
99      return i;
100 }
101 operation getRoot(){
102     for (class in getPatternElements(getPattern())){
103         if (class.inboundLink.isEmpty()){
104             return class;
105         }
106     }
107 }
108 operation getEventElements(s : SecurityEvent) : Sequence{
109     var i : new Sequence;
110     for (class in s.conditions){
111         i.add(class);
112     }
113     return i;
```

B.3. Transformación Model-to-Text

```

114 }
115 operation getEventRoot(s : SecurityEvent){
116     for (class in getEventElements(s)){
117         if (class.inboundLink.isEmpty()){
118             return class;
119         }
120     }
121 }
122 operation traversal(r) : String{
123     //If it is a leaf
124     if (numChild(r) == 0){
125         //SecurityEvent
126         if (r.isTypeOf(SecurityEvent)){
127             if (r.conditions.isEmpty()){
128                 if (r.Alias == null){
129                     return r.name;
130                 } else{
131                     return r.Alias + ' = ' + r.name;
132                 }
133             } else{
134                 return r.name + '(' + traversal(getEventRoot(r)) + ')';
135                 ;
136             }
137         } else if (r.isTypeOf(IP) or r.type().name == 'Port' or r.
138             isTypeOf(MAC)){
139             if (r.referenced.isDefined()){
140                 if (r.referenced.Alias.isDefined()){
141                     return r.referenced.Alias + '.' + r.type().name +
142                         r.type;
143                 } else{
144                     return r.type().name + r.type;
145                 }
146             } else{
147                 return r.type().name + r.type;
148             }
149         } else if (r.isTypeOf(SecurityValue)){
150             return r.value;
151         //Timer Interval
152         } else if (r.isTypeOf(TimerInterval)){
153             return 'timer:interval(' + getAttributeTimerGuard(r) +
154                 ')';
155         //Timer At
156         } else if (r.isTypeOf(TimerAt)){
157             return 'timer:at(' + getAttributesTimerAt(r) + ')';
158         } else{
159             return r.type().name;
160         }
161     }
162 }
163 
```

B. Código Fuente

```

159         }
160     } else{
161         //One child
162         if (numChild(r) == 1){
163             //Every
164             if (r.isTypeOf(Every)){
165                 return 'every ' + traversal(child(r,0));
166             }
167             //EveryDistinct
168             if (r.isTypeOf(EveryDistinct)){
169                 return 'every-distinct (' + getEveryDistinctProperty(
170                     r) + ') ' + traversal(child(r,0));
171             }
172             //Not
173             if (r.isTypeOf(Not)){
174                 return 'not ' + traversal(child(r,0));
175             }
176             //Num
177             if (r.isTypeOf(Num)){
178                 return '[' + r.count + '] ' + traversal(child(r,0));
179             }
180             //Range
181             if (r.isTypeOf(Range)){
182                 return getValuesRange(r) + traversal(child(r,0));
183             }
184             //Two children
185             if (numChild(r) == 2){
186                 //And
187                 if (r.isTypeOf(And)){
188                     if (r.inboundLink.isEmpty() == false){
189                         if (r.inboundLink.at(0).source.isTypeOf(Num) or
190                             r.inboundLink.at(0).source.isTypeOf(Range)){
191                             return '(' + traversal(child(r,0)) + ' and
192                             ' + traversal(child(r,1)) + ')';
193                         } else{
194                             return traversal(child(r,0)) + ' and
195                             ' + traversal(child(r,1));
196                         }
197                     } else{
198                         return traversal(child(r,0)) + ' and
199                         ' + traversal(child(r,1));
200                     }
201                 }
202                 //Equal
203                 if (r.isTypeOf(Equal)){
204                     return traversal(child(r,0)) + ' = ' + traversal(
205                         child(r,1));
206                 }

```

```

206 //FollowedBy
207 if(r .isTypeOf(FollowedBy)){
208     return traversal(child(r ,0)) + ,
209 -> ' + traversal(child(r ,1));
210 }
211 //GreaterEqual
212 if(r .isTypeOf(GreaterEqual)){
213     return traversal(child(r ,0)) + ' >= ' + traversal(
214         child(r ,1));
215 }
216 //GreaterThan
217 if(r .isTypeOf(GreaterThan)){
218     return traversal(child(r ,0)) + ' > ' + traversal(
219         child(r ,1));
220 }
221 //LessEqual
222 if(r .isTypeOf(LessEqual)){
223     return traversal(child(r ,0)) + ' <= ' + traversal(
224         child(r ,1));
225 }
226 //LessThan
227 if(r .isTypeOf(LessThan)){
228     return traversal(child(r ,0)) + ' < ' + traversal(
229         child(r ,1));
230 }
231 //NotEqual
232 if(r .isTypeOf(NotEqual)){
233     return traversal(child(r ,0)) + ' != ' + traversal(
234         child(r ,1));
235 }
236 //Or
237 if(r .isTypeOf(Or)){
238     if(r .inboundLink.isEmpty() == false){
239         if(r .inboundLink.at(0).source.isTypeOf(Num) or
240             r .inboundLink.at(0).source.isTypeOf(Range)){
241             return '(' + traversal(child(r ,0)) + ' or
242                 ' + traversal(child(r ,1)) + ')';
243         } else{
244             return traversal(child(r ,0)) + ' or
245                 ' + traversal(child(r ,1));
246         }
247     } else{
248         return traversal(child(r ,0)) + ' until ' + traversal(
249             child(r ,1));
250     }
251 }
252 //Until
253 if(r .isTypeOf(Until)){
254     return traversal(child(r ,0)) + ' until ' + traversal(
255         child(r ,1));
256 }
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
779

```

B. Código Fuente

```

    child(r,1));
250
251 }
252 //While
253 if(r.isTypeOf(While)){
254     return '(' + traversal(child(r,0)) + ') ' + ' while ' +
255         + '(' + traversal(child(r,1)) + ')';
256 } else{
257     //Three children
258     if(numChild(r) == 3){
259         //And
260         if(r.isTypeOf(And)){
261             return traversal(child(r,0)) + ' and '
262                 + traversal(child(r,1)) + ' and '
263                 + traversal(child(r,2));
264         }
265         //FollowedBy
266         if(r.isTypeOf(FollowedBy)){
267             return traversal(child(r,0)) + ' --> '
268                 + traversal(child(r,1)) + ' --> '
269                 + traversal(child(r,2));
270         }
271         //Or
272         if(r.isTypeOf(Or)){
273             return traversal(child(r,0)) + ' or '
274                 + traversal(child(r,1)) + ' or '
275                 + traversal(child(r,2));
276         }
277         //Four Children
278         if(numChild(r) == 4){
279             //And
280             if(r.isTypeOf(And)){
281                 return traversal(child(r,0)) + ' and '
282                     + traversal(child(r,1)) + ' and '
283                     + traversal(child(r,2)) + ' and '
284                     + traversal(child(r,3));
285             }
286             //FollowedBy
287             if(r.isTypeOf(FollowedBy)){
288                 return traversal(child(r,0)) + ' --> '
289                     + traversal(child(r,1)) + ' --> '
290                     + traversal(child(r,2)) + ' --> '
291                     + traversal(child(r,3));
292             }
293             //Or
294             if(r.isTypeOf(Or)){
295                 return traversal(child(r,0)) + ' or '
296                     + traversal(child(r,1)) + ' or '

```

B.3. Transformación Model-to-Text

```

297     ' + traversal(child(r,2)) + ' or
298     ' + traversal(child(r,3));
299         }
300     }
301   }
302   }
303 }
304 }
305 //If n = 0, child with order 1, n = m, order m - 1
306 operation child(p,n : Integer){
307   if(p.outboundLink.isEmpty()){
308     return 0;
309   }else{
310     for(class in p.outboundLink){
311       if(class.target.order == n + 1){
312         return class.target;
313       }
314     }
315     return p.outboundLink.at(n).target;
316   }
317 }
318 operation numChild(p) : Integer{
319   if(p.outboundLink.isEmpty()){
320     return 0;
321   }else{
322     return p.outboundLink.size();
323   }
324 }
325 operation getEveryDistinctProperty(r){
326   var properties : String;
327   var numDistinct : Integer = r.distinctValues.size();
328   var count : Integer = 0;
329   for(class in r.distinctValues){
330     count = count + 1;
331     if(count > numDistinct){
332       properties = properties + class.referenced.Alias + '.' +
333                     class.type().name + class.type + ',';
334     }else{
335       properties = properties + class.referenced.Alias + '.' +
336                     class.type().name + class.type;
337     }
338   }
339   return properties;
340 }
341 operation getAttributesTimerAt(r) : String{
342   var attributes : String;
343   //minutes
344   if(r.minutes > 0){
345     attributes = attributes + r.minutes + ',';

```

B. Código Fuente

```
344     } else{
345         attributes = attributes + '*', ',';
346     }
347     //hours
348     if(r.hours > 0){
349         attributes = attributes + r.hours + ', ,';
350     } else{
351         attributes = attributes + '*', ',';
352     }
353     //daysOfMonth
354     if(r.daysOfMonth > 0){
355         attributes = attributes + r.daysOfMonth + ', ,';
356     } else{
357         attributes = attributes + '*', ',';
358     }
359     //months
360     if(r.months > 0){
361         attributes = attributes + r.months + ', ,';
362     } else{
363         attributes = attributes + '*', ',';
364     }
365     //daysOfWeek
366     if(r.daysOfWeek > 0){
367         attributes = attributes + r.daysOfWeek + ', ,';
368     } else{
369         attributes = attributes + '*', ',';
370     }
371     //seconds
372     if(r.seconds > 0){
373         attributes = attributes + r.seconds;
374     } else{
375         attributes = attributes + '*';
376     }
377     return attributes;
378 }
379 operation getAttributeTimerGuard(t) : String{
380     if(t.years > 0){
381         return t.years + ' years';
382     }
383     if(t.months > 0){
384         return t.months + ' months';
385     }
386     if(t.weeks > 0){
387         return t.weeks + ' weeks';
388     }
389     if(t.days > 0){
390         return t.days + ' days';
391     }
392     if(t.hours > 0){
```

B.3. Transformación Model-to-Text

```
393         return t.hours + ' hours';
394     }
395     if(t.minutes > 0){
396         return t.minutes + ' minutes';
397     }
398     if(t.seconds > 0){
399         return t.seconds + ' seconds';
400     }
401     if(t.milliseconds > 0){
402         return t.milliseconds + ' milliseconds';
403     }
404 }
405 operation getValuesRange(r){
406     var range : String;
407     if(r.lowEndpoint > 0){
408         range = range + '[' + r.lowEndpoint + ':';
409     } else{
410         range = range + '[:';
411     }
412     if(r.highEndpoint > 0){
413         range = range + r.highEndpoint + '] ';
414     } else{
415         range = range + '] ';
416     }
417     return range;
418 }
419 %]
```

B.3.2. Plug-in de integración de la transformación M2T

El fichero encargado de añadir funcionalidad al botón del editor de Eclipse se encuentra en el Listado B.79

Listado B.79: Archivo Java para añadir funcionalidad al botón

```
1 package org.eclipse.epsilon.eugenia.examples.security.m2t;
2
3 import java.io.File;
4 import java.io.FileWriter;
5 import java.io.IOException;
6 import java.net.MalformedURLException;
7 import java.net.URI;
8 import java.net.URISyntaxException;
9 import java.net.URL;
10 import java.util.Iterator;
11
12 import org.eclipse.core.resources.IFile;
13 import org.eclipse.core.resources.ResourcesPlugin;
```

B. Código Fuente

```
14 import org.eclipse.core.runtime.FileLocator;
15 import org.eclipse.core.runtime.Path;
16 import org.eclipse.emf.ecore.EObject;
17 import org.eclipse.emf.ecore.resource.Resource;
18 import org.eclipse.epsilon.commons.util.StringProperties;
19 import org.eclipse.epsilon.egl.EglTemplateFactory;
20 import org.eclipse.epsilon.egl.EglTemplateFactoryModuleAdapter;
21 import org.eclipse.epsilon.emc.emf.EmfModel;
22 import org.eclipse.epsilon.emc.emf.InMemoryEmfModel;
23 import org.eclipse.epsilon.eol.IEolExecutableModule;
24 import org.eclipse.epsilon.eol.exceptions.models.
25     EolModelLoadingException;
26 import org.eclipse.jface.action.IAction;
27 import org.eclipse.jface.viewers.ISelection;
28 import org.eclipse.jface.viewers.IStructuredSelection;
29 import org.eclipse.ui.IObjectActionDelegate;
30 import org.eclipse.ui.IWorkbenchPart;
31 import org.eclipse.epsilon.eugenia.examples.security.*;
32 import security.diagram.edit.parts.RootEditPart;
33
34 public class EglTransformModelToEPL implements IObjectActionDelegate{
35
36     protected IEolExecutableModule module;
37
38     private Resource resource;
39
40     protected Object result;
41
42     public void execute() throws Exception {
43
44         module = createModule();
45         module.parse(getFile());
46         EmfModel Modelo = createEmfModel("security_egl", "security", true,
47             , true);
48
49         module.getContext().getModelRepository().addModel(Modelo);
50
51         result = module.execute();
52         postProcess();
53
54         module.getContext().getModelRepository().dispose();
55     }
56
57     protected EmfModel createEmfModel(String name, String metamodel,
58         boolean readOnLoad, boolean storeOnDisposal) throws
59         EolModelLoadingException, URISyntaxException, IOException {
60         InMemoryEmfModel emfModel = new InMemoryEmfModel(resource);
61         StringProperties properties = new StringProperties();
```

B.3. Transformación Model-to-Text

```
59     properties .put(EmfModel.PROPERTY_NAME, name);
60     properties .put(EmfModel.PROPERTY_METAMODEL_URI, metamodel);
61     properties .put(EmfModel.PROPERTY_IS_METAMODEL_FILE_BASED, "false"
62                     );
62     properties .put(EmfModel.PROPERTY_READONLOAD, readOnLoad + " ");
63     properties .put(EmfModel.PROPERTY_STOREONDISPOSAL, storeOnDisposal
64                     +
65                     );
64     emfModel.load(properties , null);
65     return emfModel;
66 }
67
68
69     protected File getFile() throws URISyntaxException ,
70     MalformedURLException , IOException {
71
72         String fileName = "/eglExample.egl";
73         URI uri = EglTransformModelToEPL.class.getResource(fileName) .
74             toURI();
73         URL resolve = FileLocator.resolve(uri.toURL());
74         uri = resolve.toURI();
75         return new File(uri);
76     }
77
78
79     public IEolExecutableModule createModule() {
80         return new EglTemplateFactoryModuleAdapter(new EglTemplateFactory
81             ());
82     }
83
84
84     public void postProcess() throws IOException {
85         String resultado = new String(result.toString());
86         final Path pathToModel = new Path(resource.getURI() .
87             toPlatformString(true));
87         final IFile ifModel = ResourcesPlugin.getWorkspace().getRoot() .
88             getFile(pathToModel);
88         final File fModel = ifModel.getLocation().toFile() .
89             getCanonicalFile();
89         final File fTarget = new File(fModel.getParentFile(), "Code.epl")
90             ;
90
91         final FileWriter fw = new FileWriter(fTarget);
92         try {
93             fw.write(resultado);
94         } finally {
95             fw.close();
96         }
97     }
98     public String result(){
```

B. Código Fuente

```
99     return result.toString();
100 }
101
102 @Override
103 public void run(IAction action) {
104     try {
105         this.execute();
106     } catch (Exception e) {
107         // TODO Auto-generated catch block
108         e.printStackTrace();
109     }
110 }
111
112 @SuppressWarnings("rawtypes")
113 @Override
114 public void selectionChanged(IAction action, ISelection selection) {
115     if (selection instanceof IStructuredSelection) {
116         final IStructuredSelection sel = (IStructuredSelection)
117             selection;
118
119         for (Iterator it = sel.iterator(); it.hasNext(); ) {
120             final Object e = it.next();
121             if (e instanceof RootEditPart) {
122                 final RootEditPart aed = (RootEditPart)e;
123                 final EObject model = aed.resolveSemanticElement();
124                 this.resource = model.eResource();
125             }
126         }
127     }
128 }
129
130 @Override
131 public void setActivePart(IAction action, IWorkbenchPart targetPart)
132 {
133     // TODO Auto-generated method stub
134 }
135
136 }
```

B.4. Personalización del editor

El *script* en EOL (denominado *ECore2GMF.eol*) encargado de la personalización del editor se divide en dos partes: una para la personalización de la paleta gráfica y otra para la personalización del resto del editor.

B.4.1. Paleta gráfica

El código del *script* en EOL que personaliza la paleta gráfica puede verse en Listado B.80.

Listado B.80: Personalización de la paleta

```

1  —— Palette Personalization
2
3  var palette = GmfTool!Palette.all.first();
4
5  var ObjectGroup = palette.tools.selectOne(o|o.title='Objects');
6  var ConnectionsGroup = palette.tools.selectOne(o|o.title='Connections')
7    ;
8
9  var AndTool = ObjectGroup.tools.selectOne(o|o.title='And');
10 var EqualTool = ObjectGroup.tools.selectOne(o|o.title='Equal');
11 var EveryTool = ObjectGroup.tools.selectOne(o|o.title='Every');
12 var EveryDistinctTool = ObjectGroup.tools.selectOne(o|o.title='
13   EveryDistinct');
14 EveryDistinctTool.title = 'Every\u2014Distinct';
15 var FollowedByTool = ObjectGroup.tools.selectOne(o|o.title='FollowedBy'
16   );
17 FollowedByTool.title = 'Followed\u2014By';
18 var GreaterThanTool = ObjectGroup.tools.selectOne(o|o.title='
19   GreaterThan');
20 GreaterThanTool.title = 'Greater\u2014Than';
21 var GreaterEqualTool = ObjectGroup.tools.selectOne(o|o.title='
22   GreaterEqual');
23 GreaterEqualTool.title = 'Greater\u2014Equal\u2014Than';
24 var InsertIntoTool = ObjectGroup.tools.selectOne(o|o.title='InsertInto'
25   );
26 var IPTool = ObjectGroup.tools.selectOne(o|o.title='IP');
27 IPTool.title = 'IP\u2014Attribute';
28 var LessEqualTool = ObjectGroup.tools.selectOne(o|o.title='LessEqual');
29 LessEqualTool.title = 'Less\u2014Equal\u2014Than';
30 var LessThanTool = ObjectGroup.tools.selectOne(o|o.title='LessThan');
31 LessThanTool.title = 'Less\u2014Than';
32 var LinkTool = ConnectionsGroup.tools.selectOne(o|o.title='Link');
33 var MACTool = ObjectGroup.tools.selectOne(o|o.title='MAC');
34 MACTool.title = 'MAC\u2014Attribute';
35 var NotTool = ObjectGroup.tools.selectOne(o|o.title='Not');
36 var NotEqualTool = ObjectGroup.tools.selectOne(o|o.title='NotEqual');
37 NotEqualTool.title = 'Not\u2014Equal';
38 var NumTool = ObjectGroup.tools.selectOne(o|o.title='Num');
39 var OrTool = ObjectGroup.tools.selectOne(o|o.title='Or');
40 var SecurityEventTool = ObjectGroup.tools.selectOne(o|o.title='
41   SecurityEvent');
42 SecurityEventTool.title = 'Security\u2014Event';
43 var PatternTool = ObjectGroup.tools.selectOne(o|o.title='Pattern');
```

B. Código Fuente

```
37 var PortTool = ObjectGroup.tools.selectOne(o|o.title='Port');
38 PortTool.title = 'Port\u2014Attribute';
39 var RangeTool = ObjectGroup.tools.selectOne(o|o.title='Range');
40 var SelectTool = ObjectGroup.tools.selectOne(o|o.title='Select');
41 var SecurityValueTool = ObjectGroup.tools.selectOne(o|o.title='
    SecurityValue');
42 SecurityValueTool.title = 'Value';
43 var TimerAtTool = ObjectGroup.tools.selectOne(o|o.title='TimerAt');
44 TimerAtTool.title = 'Timer\u2014At';
45 var TimerIntervalTool = ObjectGroup.tools.selectOne(o|o.title='
    TimerInterval');
46 TimerIntervalTool.title = 'Timer\u2014Interval';
47 var TimerWithinTool = ObjectGroup.tools.selectOne(o|o.title='
    TimerWithin');
48 TimerWithinTool.title = 'Timer\u2014Within';
49 var TimerWithinMaxTool = ObjectGroup.tools.selectOne(o|o.title='
    TimerWithinMax');
50 TimerWithinMaxTool.title = 'Timer\u2014Within\u2014Max';
51 var TimestampTool = ObjectGroup.tools.selectOne(o|o.title='Timestamp');
52 TimestampTool.title = 'Timestamp\u2014Attribute';
53 var UntilTool = ObjectGroup.tools.selectOne(o|o.title='Until');
54 var WhileTool = ObjectGroup.tools.selectOne(o|o.title='While');
55
56 ---Create ClauseGroup
57 var ClauseGroup = new GmfTool!ToolGroup;
58 ClauseGroup.title = 'Clauses';
59 ClauseGroup.collapsible = true;
60 palette.tools.add(ClauseGroup);
61 ClauseGroup.tools.add(PatternTool);
62 ClauseGroup.tools.add(SelectTool);
63 ClauseGroup.tools.add(InsertIntoTool);
64
65 ---Create EventGroup
66 var EventGroup = new GmfTool!ToolGroup;
67 EventGroup.title = 'Security\u2014Elements';
68 EventGroup.collapsible = true;
69 palette.tools.add(EventGroup);
70 EventGroup.tools.add(SecurityEventTool);
71 EventGroup.tools.add(PortTool);
72 EventGroup.tools.add(IPTool);
73 EventGroup.tools.add(SecurityValueTool);
74 EventGroup.tools.add(MACTool);
75 EventGroup.tools.add(TimestampTool);
76
77 ---Create TemporalOpGroup
78 var TemporalOpGroup = new GmfTool!ToolGroup;
79 TemporalOpGroup.title = 'Temporal\u2014Operators';
80 TemporalOpGroup.collapsible = true;
81 palette.tools.add(TemporalOpGroup);
```

```

82 | TemporalOpGroup.tools.add(FollowedByTool);
83 | TemporalOpGroup.tools.add(TimerWithinTool);
84 | TemporalOpGroup.tools.add(TimerWithinMaxTool);
85 | TemporalOpGroup.tools.add(WhileTool);
86 | TemporalOpGroup.tools.add(TimerIntervalTool);
87 | TemporalOpGroup.tools.add(TimerAtTool);
88 |
89 |
90 | —Create RepetitionOpGroup
91 | var RepetitionOpGroup = new GmfTool!ToolGroup;
92 | RepetitionOpGroup.title = 'RepetitionOperators';
93 | RepetitionOpGroup.collapsible = true;
94 | palette.tools.add(RepetitionOpGroup);
95 | RepetitionOpGroup.tools.add(UntilTool);
96 | RepetitionOpGroup.tools.add(EveryTool);
97 | RepetitionOpGroup.tools.add(EveryDistinctTool);
98 | RepetitionOpGroup.tools.add(NumTool);
99 | RepetitionOpGroup.tools.add(RangeTool);
100 |
101 | —Create Link
102 | palette.tools.add(LinkTool);
103 |
104 | —Create LogicalOpGroup
105 | var LogicalOpGroup = new GmfTool!ToolGroup;
106 | LogicalOpGroup.title = 'LogicalOperators';
107 | LogicalOpGroup.collapsible = true;
108 | palette.tools.add(LogicalOpGroup);
109 | LogicalOpGroup.tools.add(AndTool);
110 | LogicalOpGroup.tools.add(OrTool);
111 | LogicalOpGroup.tools.add(NotTool);
112 |
113 | —Create RelationalOpGroup
114 | var RelationalOpGroup = new GmfTool!ToolGroup;
115 | RelationalOpGroup.title = 'RelationalOperators';
116 | RelationalOpGroup.collapsible = true;
117 | palette.tools.add(RelationalOpGroup);
118 | RelationalOpGroup.tools.add(EqualTool);
119 | RelationalOpGroup.tools.add(NotEqualTool);
120 | RelationalOpGroup.tools.add(LessThanTool);
121 | RelationalOpGroup.tools.add(LessEqualTool);
122 | RelationalOpGroup.tools.add(GreaterThanTool);
123 | RelationalOpGroup.tools.add(GreaterEqualTool);
124 |
125 | —Delete ObjectGroup
126 | palette.tools.remove(ObjectGroup);
127 | —Delete LinksGroup
128 | palette.tools.remove(ConnectionsGroup);

```

B. Código Fuente

B.4.2. Entorno principal del editor

El código del *script* en EOL que personaliza el entorno principal del editor puede verse en Listado B.81.

Listado B.81: Personalización del entorno

```
1 -- In the emf file if I try label = "none" for the While class , I have
2   an error
3 -- this fix the error
4 basicstylevar labelWhile = GmfMap!FeatureLabelMapping.all.selectOne(r|r.
5   DiagramLabel.name = 'WhileLabel');
6 delete labelWhile;
7
8 basicstylevar labelSelectName = GmfMap!FeatureLabelMapping.all.selectOne(
9   r|r.DiagramLabel.name = 'SelectLabel');
10 labelSelectName.editMethod = basicstylenull;
11
12 -- Figures Personalization
13
14 -- Logical Operators
15 personalization("And", "Ellipse", "&", 15, 20);
16 personalization("Or", "Ellipse", "||", 15, 20);
17 personalization("Not", "Ellipse", "!", 15, 20);
18
19 -- Relational Operators
20 personalization("Equal", "Ellipse", "=", 15, 20);
21 personalization("NotEqual", "Ellipse", "!=" ,15,20);
22 personalization("LessThan", "Ellipse", "<", 15, 20);
23 personalization("GreaterThan", "Ellipse", ">", 15, 20);
24 personalization("LessEqual", "Ellipse", "<=", 15, 20);
25 personalization("GreaterEqual", "Ellipse", ">=", 15, 20);
26
27 -- Temporal Operators
28 personalization("FollowedBy", "Ellipse", "-->", 15, 20);
29 personalization("Range", "Ellipse", "[ ]", 15, 20);
30
31 -- Operation personalization
32 operation personalization(type : String, figure : String, value :
33   String, tamFont : Integer,
34   tamPref : Integer){
35   basicstylevar strFigure = basicstylenull;
36   basicstyleif (figure == "Ellipse"){
37     basicstylevar nodePreferredSize = basicstylenew GmfGraph!Dimension;
38     nodePreferredSize.dx = tamPref;
39     nodePreferredSize.dy = tamPref;
40     strFigure = GmfGraph!Ellipse.all.selectOne(r|r.name = type +
41       Figure');
42     strFigure.preferredSize = nodePreferredSize;
43     strFigure.layout = basicstylenew GmfGraph!GridLayout;
44     basicstylevar strLabel = GmfGraph!Label.all.selectOne(l|l.name =
45       type + 'LabelFigure');
```

B.4. Personalización del editor

```
38     strLabel.text = value;
39     strLabel.font = basicstylenew GmfGraph! BasicFont;
40     strLabel.font.style = GmfGraph! FontStyle#BOLD;
41     strLabel.font.height = tamFont;
42     strLabel.layoutData = basicstylenew GmfGraph! GridLayoutData;
43     strLabel.layoutData.horizontalAlignment = GmfGraph! Alignment#
44         CENTER;
45     strLabel.layoutData.verticalAlignment = GmfGraph! Alignment#CENTER
46     ;
47 }
48 basicstyleif (figure == "Rectangle"){
49     strFigure = GmfGraph! Rectangle.all.selectOne(r|r.name = type +
50         'Figure');
51     basicstylevar strLabel = GmfGraph! Label.all.selectOne(l|l.name=
52         type + 'LabelFigure');
53     strLabel.text = value;
54     strLabel.font = basicstylenew GmfGraph! BasicFont;
55     strLabel.font.style = GmfGraph! FontStyle#BOLD;
56     strLabel.font.height = tamFont;
57     strLabel.layoutData = basicstylenew GmfGraph! BorderLayoutData;
58 }
59 — Operation delWrongActions
60 operation delWrongActions(name : String, title : String){
61     basicstylevar del = GmfMap! ChildReference.all.select(d|d.
62         ContainmentFeature.name = name and d.Child.Tool.title = title);
63     delete del;
64 }
```

B. Código Fuente

Bibliografía

- [1] GPL v2 (Jun 1991), <http://www.gnu.org/licenses/gpl-2.0.html>
- [2] Subclipse Download (Jul 1999), <http://subclipse.tigris.org/servlets/ProjectProcess;jsessionid=C809587C868FEF117F6F619E1A67393F?pageID=p4wYuA>
- [3] Software Engineering (7th Edition). Addison Wesley, 7 edn. (May 2004), <http://www.worldcat.org/isbn/0321210263>
- [4] Object Constraint Language, Version 2.0 (May 2006), <http://www.omg.org/spec/OCL/2.0/>
- [5] Eclipse (Jul 2012), <http://www.eclipse.org/>
- [6] Eclipse Modeling Indigo. Linux x86_64. (Jul 2012), http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/indigo/R/eclipse-modeling-indigo-linux-gtk-x86_64.tar.gz
- [7] Ecore Tools (Jul 2012), <http://www.eclipse.org/modeling/emft/?project=ecoretools>
- [8] EMF (Jul 2012), <http://www.eclipse.org/modeling/emf/>
- [9] Emfatic (Jul 2012), <http://www.eclipse.org/epsilon/doc/articles/emfatic/>
- [10] Emfatic Update Site (Jul 2012), <http://download.eclipse.org/emfatic/update>
- [11] Epsilon Download (Jul 2012), <http://www.eclipse.org/epsilon/download/>
- [12] Epsilon Generation Language (Jul 2012), <http://www.eclipse.org/epsilon/doc/egl/>
- [13] Epsilon Interim Update Site (Jul 2012), <http://download.eclipse.org/epsilon/interim/>

Bibliografía

- [14] Epsilon Object Language (Jul 2012), <http://www.eclipse.org/epsilon/doc/eol/>
- [15] Epsilon Update Site (Jul 2012), <http://download.eclipse.org/epsilon/updates/>
- [16] Epsilon Validation Language (Jul 2012), <http://www.eclipse.org/epsilon/doc/evl/>
- [17] Esper 4.6.0 - Esper reference (Jul 2012), <http://esper.codehaus.org/esper-4.6.0/doc/reference/en-US/html/index.html>
- [18] Esper Product (Jul 2012), <http://www.espertech.com/products/esper.php>
- [19] EuGENia (Jul 2012), <http://www.eclipse.org/epsilon/doc/eugenia/>
- [20] EuGENia Imágenes (Jul 2012), <http://www.eclipse.org/epsilon/doc/articles/eugenia-nodes-with-images/>
- [21] EuGENia Tutorial (Jul 2012), <http://www.eclipse.org/epsilon/doc/articles/eugenia-gmf-tutorial/>
- [22] EUnit (Jul 2012), <http://www.eclipse.org/epsilon/doc/eunit/>
- [23] GMF Tooling (Jul 2012), <http://eclipse.org/gmf-tooling/>
- [24] HUTN Epsilon (Jul 2012), <http://www.omg.org/spec/HUTN/>
- [25] Inkscape (Jul 2012), <http://www.inkscape.org/?lang=es/>
- [26] Modelo Cascada (Jul 2012), http://commons.wikimedia.org/wiki/File:Modelo_Cascada_Secuencial.jpg?uselang=es
- [27] MOF 2 XMI Mapping. OMG (Jul 2012), <http://www.omg.org/spec/XMI/>
- [28] Subclipse (Jul 2012), <http://subclipse.tigris.org/>
- [29] Ayllon, V., Reina, J.M.: CEP/ESP: procesamiento y correlacion de gran cantidad de eventos en arquitecturas SOA. In: IV Jornadas Cientifico-Tecnicas en Servicios Web y SOA. pp. 97–110. Sevilla (Oct 2008)
- [30] Boubeta Puig, J., Ortiz, G., Medina Bulo, I.: Procesamiento de Eventos Completos en Entornos SOA: Caso de Estudio para la Detección Temprana de Epidemias. In: Actas de las VII Jornadas de Ciencia e Ingeniería de Servicios. pp. 63–76. Servizo de publicacions da Universidade da Coruña, A Coruña, Spain (Sep 2011)

- [31] Bruce, D.: What makes a good domain-specific language? APOSTLE, and its approach to parallel discrete event simulation (1997)
- [32] Christopher, D.L., Ramming, J.C.: Two Application Languages in Software Production. In: In USENIX Symposium on Very High Level Languages (1994)
- [33] Collins-Sussman, B., Fitzpatrick, B.W., Pilato, C.M.: Version Control with Subversion (Jun 2004), <http://svnbook.red-bean.com/nightly/en/index.html>
- [34] D., B.: Functional Requirements and Use Cases
- [35] Deursen, A.V., Klint, P.: Little languages: little maintenance? (1998)
- [36] Deursen, A., Klint, P., Visser, J.: Domain-specific languages: an annotated bibliography (2000), <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=doi=10.1.1.33.8207>
- [37] Djuric, D., Gasevic, D., Devedzic, V.: The Tao of Modeling Spaces (2006), http://www.jot.fm/issues/issue_2006_11/article4/
- [38] Dorado-Cerón, J.A., Boubeta-Puig, J.: Detección de Ataques de Seguridad mediante Procesamiento de Eventos Complejos (Mar 2012), <http://hdl.handle.net/10498/14675>
- [39] Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications (Aug 2010), <http://www.manning.com/etzion/>
- [40] Frankel, D.: The Eclipse Modeling Framework and MDA (2004), http://www.eclipsecon.org/2004/EclipseCon_2004_TechnicalTrackPresentations/40_Frankel.pdf
- [41] Greenfield, J., Short, K.: Software factories: assembling applications with patterns, models, frameworks and tools. In: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. pp. 16–27. OOPSLA ’03, ACM, New York, NY, USA (2003), <http://doi.acm.org/10.1145/949344.949348>
- [42] Groth, R.: Is the software industry’s productivity declining? Software, IEEE 21(6), 92 – 94 (nov-dec 2004)
- [43] Kieburtz, R.B., McKinney, L., Bell, J.M., Hook, J., Kotov, A., Lewis, J., Oliva, D.P., Sheard, T., Smith, I., Walton, L.: A software engineering experiment in software component generation. In: Proceedings of the 18th international conference on Software engineering. pp. 542–552. ICSE ’96, IEEE Computer Society, Washington, DC, USA (1996), <http://dl.acm.org/citation.cfm?id=227726.227842>

Bibliografía

- [44] Kleppe, A.G., Warmer, J., Bast, W.: MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
- [45] Kolovos, D., Rose, L., Paige, R., García, A.: The Epsilon Book (2012), <http://www.eclipse.org/epsilon/doc/book/>
- [46] Krueger, C.W.: Software reuse. ACM Comput. Surv. 24(2), 131–183 (Jun 1992), <http://doi.acm.org/10.1145/130844.130856>
- [47] Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2001)
- [48] Make, M.: Pragmatic Version Control : Using Subversion. Raleigh, EUA : Pragmatic Programmers (2006), <http://148.201.96.14/dc/ver.aspx?ns=000190521>
- [49] Mernik, M., Heering, J., Sloane, A.M.: When and how to develop domain-specific languages. ACM Comput. Surv. 37(4), 316–344 (2005), <http://doi.acm.org/10.1145/1118890.1118892>
- [50] Miller, J., Mukerji, J.: MDA Guide Version 1.0.1. OMG (Jun 2003), <http://www.omg.org/cgi-bin/doc?omg/03-06-01>
- [51] OMG: Meta Object Facility (MOF) Core Specification v2 (2006), <http://www.omg.org/spec/MOF/2.0/>
- [52] OMG Architecture Board ORMSC: Model driven architecture (MDA). OMG document number ormsc/2001-07-01 (Jul 2001), <http://www.omg.org>
- [53] Paschke, A.: Design Patterns for Complex Event Processing (Jun 2008), <http://arxiv.org/abs/0806.1100>
- [54] Puttanna, P.: Is your company ready for MDA ? (2011), <http://smlk.es/model-driven-architecture-mda>
- [55] Seidewitz, E.: What models mean. Software, IEEE 20(5), 26 – 32 (sept-oct 2003)
- [56] Sirer, E.G., Bershad, B.N.: Using production grammars in software testing. SIGPLAN Not. 35(1), 1–13 (Dec 1999), <http://doi.acm.org/10.1145/331963.331965>

Bibliografía

- [57] Stahl, T., Voelter, M., Czarnecki, K.: Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons (2006)
- [58] Standish Group International: Chaos: A Recipe for Success (1999),
http://www4.informatik.tu-muenchen.de/lehre/vorlesungen/vse/WS2004/1999_Standish_Chaos.pdf
- [59] Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework, 2nd Edition. Pearson Education (2004)
- [60] Tony Clark, Andy Evans, P.S.J.W.: Applied Metamodelling A Foundation for Language Driven Development. Xactium (2004), [http://eprints.mdx.ac.uk/6060/1/Clark-Applied_Metamodelling_\(Second_Edition\)\[1\].pdf](http://eprints.mdx.ac.uk/6060/1/Clark-Applied_Metamodelling_(Second_Edition)[1].pdf)