



TESINA DE LICENCIATURA

Título: Análisis de herramientas de generación automática de código Android a partir de modelos

Autores: Gabriel Adrian Vidal

Director: Dra. Claudia Pons

Codirector: -

Asesor profesional: -

Carrera: Licenciatura en Sistemas

Resumen

Inicialmente solo se desarrollaban aplicaciones con herramientas y lenguajes de bajo nivel, destinadas a un tipo de arquitectura en particular, los cuales requerían de complicadas instrucciones para completar su objetivo. Hace ya varios años se implementaron los lenguajes de alto nivel, permitiéndonos algo más de abstracción en el desarrollo y la consecuente portabilidad de las aplicaciones a través de diferentes sistemas operativos, donde el compilador automáticamente se encargará de la traducción al lenguaje de bajo nivel.

En la actualidad, adoptando el concepto de desarrollo dirigido por modelos, se pretende subir la apuesta para ahora abstraernos también del lenguaje de programación, con lo cual se han realizado productos de software, donde a partir del diseño y modelado, se produce la generación automática del código de la aplicación.

La tesina se enfoca específicamente en la generación de código para ser ejecutado en el sistema operativo Android. El objetivo de nuestro trabajo consiste en la comparación de las herramientas disponibles, para lo cual se logró identificar las más relevantes (App Inventor, Rational Rhapsody y Acceleo), analizar sus características y presentar ejemplos concretos de aplicación para cada una.

Palabras Claves

Android, generación, código, modelo, herramienta, aplicación, MDD, Actris, App Inventor, Rational Rhapsody, Acceleo.

Trabajos Realizados

- *Investigación y definición del marco teórico que describe las características de las herramientas analizadas.*
- *Instalación y configuración de las herramientas para la posterior implementación de modelos, generación y ejecución del código Android, tanto en el simulador de la PC como en un dispositivo móvil.*
- *Comparación de características teóricas y prácticas de las herramientas.*

Conclusiones

- *App Inventor es fácil de usar, dirigido a aplicaciones simples que pueden ser compartidas en Google Play.*
- *Rhapsody es una herramienta completa y estable, mantiene sincronizado el modelo y el código, aunque el entorno gráfico no sea amigable para el usuario y no sea software libre. Se usa en sistemas embebidos de tiempo real e interfaces de usuario en PC.*
- *Acceleo se integra con Eclipse y el Framework EMF. Sus modelos de entrada son independientes de los lenguajes de salida. Es una herramienta completa basada en un estándar internacional de la OMG.*

Trabajos Futuros

- *Investigar el diseño y generación automática de código Android a través de modelos, para el IDE Android Studio.*
- *Producir la creación de nuevos bloques o investigar como lo hace Google/MIT con App Inventor.*
- *Para Rational Rhapsody, crear la trazabilidad faltante entre el modelo y las propiedades de generación que dan como resultado el código Android.*
- *Desarrollar y producir algún mecanismo de roundtripping en Acceleo.*

Análisis de herramientas de generación automática de código Android a partir de modelos

Tesina de Licenciatura

Alumnos:

Gabriel Adrian Vidal

Director:

Dra. Claudia Pons

Noviembre 2016



Facultad de Informática
Universidad Nacional de La Plata

Agradecimientos

A mis padres Marta e Ismael, por haberme dado la posibilidad de estudiar y apoyarme todo este tiempo. También a mi hermana Rocio con quien comparto la convivencia.

A mi abuelo y mi tía que siempre estuvieron presentes.

A mis amigos de Las Flores y aquellos que conocí en mi transcurso por La Plata.

A la profesora Claudia Pons por su ayuda constante e inmediata.

Índice

1	INTRODUCCIÓN	7
1.1	OBJETIVOS	7
1.2	MOTIVACIÓN	7
1.3	HERRAMIENTAS ANALIZADAS	8
2	CONCEPTOS BÁSICOS	9
2.1	MDD (MODEL DRIVEN DEVELOPMENT)	9
2.2	MDA (MODEL DRIVEN ARCHITECTURE)	9
2.3	MODELOS	10
2.4	METAMODELOS	11
2.5	TRANSFORMACIÓN	11
2.5.1	OMG (Object Management Group)	12
2.5.2	MOF y la arquitectura de cuatro capas	12
2.5.3	MOFM2T	14
2.6	SISTEMA OPERATIVO ANDROID	14
2.6.1	Características	14
2.6.2	Arquitectura	15
2.6.3	SDK de Android	16
2.6.4	Android Development Tools (ADT)	16
2.6.5	Android API Level	16
3	ARCTIS	18
3.1	BLOQUES DE CONSTRUCCIÓN REACTIVA	18
3.2	MÁQUINA DE ESTADO EXTERNO (ESM)	20
3.3	TIPOS DE BLOQUES	20
3.4	BENEFICIOS	21
3.5	INSTALANDO ARCTIS EN ECLIPSE	22
3.6	CONSTRUYENDO UNA SIMPLE APLICACIÓN	23
3.7	TESIS: "DESARROLLO DE APLICACIONES ANDROID CON ARCTIS"	24
3.7.1	Descripción abstracta	25
3.7.2	Contribución	25
3.7.3	Conclusión	26
4	APP INVENTOR	27
4.1	HISTORIA	27
4.2	ESTRUCTURA	27
4.3	MÁS SOBRE MIT APP INVENTOR	28
4.4	REQUISITOS DEL SISTEMA PARA APP INVENTOR	28
4.5	CONFIGURACIÓN DE APP INVENTOR 1	29
4.6	DESVENTAJAS	32
4.6.1	Desventajas solucionadas en App Inventor 2	32
4.7	COMPONENTES	32
4.8	SELECCIONAR LOS COMPONENTES PARA DISEÑAR LA APLICACIÓN	32
4.9	BLOQUES	33
4.10	COMPRESIÓN DE BLOQUES Y CÓMO FUNCIONAN	33
4.10.1	Bloques controladores de eventos	33
4.10.2	Comandos y expresiones	34

4.11	MANIPULAR EL ESTADO DE LOS COMPONENTES.....	35
4.12	NOVEDADES DE APP INVENTOR 2.....	37
4.12.1	<i>Dropdowns (menú desplegable)</i>	37
4.12.2	<i>Mutadores</i>	38
4.12.3	<i>Variables globales y locales</i>	38
4.12.4	<i>Al2 colores</i>	38
4.12.5	<i>Otras características modificadas</i>	39
4.13	CONFIGURACIÓN DE APP INVENTOR 2.....	39
4.14	DISEÑADOR Y EDITOR DE BLOQUES.....	41
4.14.1	<i>Diseñador de App Inventor 2</i>	41
4.14.2	<i>Editor de bloques de App Inventor 2</i>	41
4.15	DESARROLLO INCREMENTAL CON EL EDITOR DE BLOQUES.....	42
4.15.1	<i>Do it</i>	42
4.15.2	<i>Observación de las variables</i>	42
4.16	ADVERTENCIAS Y ERRORES.....	42
4.17	COMPARTIR Y EMPAQUETAR APLICACIONES.....	44
4.17.1	<i>Compartir su aplicación para que otros puedan editarlo (.aia)</i>	44
4.17.2	<i>Empaquetar su aplicación para su uso por otros (.apk)</i>	44
4.18	CONSEJOS Y TRUCOS DE APP INVENTOR 2.....	45
5	IBM RATIONAL RHAPSODY.....	46
5.1	HISTORIA.....	46
5.2	CARACTERÍSTICAS.....	46
5.3	INSTALAR EL SDK DE ANDROID Y VINCULARLO CON ECLIPSE.....	48
5.3.1	<i>Instalando el JDK</i>	48
5.3.2	<i>Instalando Eclipse</i>	51
5.3.3	<i>Instalar el SDK de Android</i>	52
5.3.4	<i>Instalar el plugin ADT</i>	55
5.4	DESCARGA E INSTALACIÓN DE IBM RATIONAL RHAPSODY DEVELOPER PARA JAVA... 58	58
5.4.1	<i>Preparación de la instalación de Rational Rhapsody Developer para Java</i> ..	58
5.4.2	<i>Instalar Rational Rhapsody para Windows</i>	59
5.4.3	<i>Instalación de la clave de licencia (Original)</i>	59
5.4.4	<i>Instalación de la clave de licencia (evaluación por 30 días)</i>	60
5.5	CONFIGURACIÓN DE ECLIPSE Y DE RATIONAL RHAPSODY.....	61
5.5.1	<i>Requisitos de integración de la plataforma</i>	61
5.5.2	<i>Vinculación de Rational Rhapsody para Eclipse</i>	62
5.5.3	<i>Instalando los plugins para integración con Eclipse</i>	62
5.5.4	<i>Comprobación de la configuración de Eclipse</i>	63
5.6	UTILIZANDO LA PLATAFORMA DE INTEGRACIÓN CON ECLIPSE.....	63
5.6.1	<i>Perspectivas de Rational Rhapsody en Eclipse</i>	63
5.7	CONFIGURACIÓN DE RATIONAL RHAPSODY CON PROPIEDADES.....	64
5.8	PROYECTO.....	64
5.8.1	<i>Perfiles del proyecto</i>	64
5.8.2	<i>Perfil para aplicaciones Android</i>	65
5.8.3	<i>El uso de unidades de proyecto</i>	65
5.9	DISEÑO DE PROYECTOS.....	65
5.9.1	<i>Agregando elementos al modelo Rational Rhapsody</i>	66
5.9.2	<i>Agregar elementos a las aplicaciones Android</i>	66
5.9.3	<i>Creación de paquetes</i>	66
5.9.4	<i>Creación de componentes bajo un paquete</i>	66
5.9.5	<i>Crear diagramas</i>	67
5.9.6	<i>Definición de los atributos de una clase</i>	67
5.9.7	<i>Administración de operaciones y creación de constructores</i>	68
5.9.8	<i>Adición de enumeraciones de Java a un modelo</i>	68

5.9.9	<i>Bloques estáticos de clases en un modelo</i>	69
5.9.10	<i>Comentarios Javadoc</i>	69
5.9.11	<i>Definición de los estereotipos</i>	69
5.9.12	<i>El establecimiento de la herencia de estereotipos</i>	70
5.9.13	<i>La asociación de los estereotipos con un elemento</i>	70
5.9.14	<i>Definición de librerías de modelos</i>	70
5.9.15	<i>Rational Rhapsody API</i>	70
5.9.16	<i>Características de la API de Java</i>	71
5.10	PROYECTOS ANDROID Y RATIONAL RHAPSODY SOBRE ECLIPSE	71
5.10.1	<i>Crear proyecto Rhapsody y asociarlo a un nuevo proyecto Android</i>	71
5.10.2	<i>Crear proyecto Rhapsody y asociarlo a un proyecto Android importado</i>	71
5.10.3	<i>Importar un proyecto Rhapsody y asociarlo a un nuevo proyecto Android</i>	72
5.10.4	<i>Importar y asociar proyectos Rhapsody y Android</i>	72
5.10.5	<i>Ejemplo de Android Home Alarm</i>	72
5.11	GENERACIÓN DE CÓDIGO A PARTIR DE UN MODELO DE RATIONAL RHAPSODY	74
5.11.1	<i>Sincronización entre modelos y código</i>	74
5.11.2	<i>Código roundtripping</i>	75
5.11.3	<i>Generación de código de forma incremental</i>	75
5.11.4	<i>La generación de código en paralelo</i>	75
5.11.5	<i>Generación de código en proyectos multilingües</i>	76
5.12	GENERAR Y COMPILAR EL CÓDIGO DE UNA APLICACIÓN RATIONAL RHAPSODY INTEGRADA A ECLIPSE	76
5.12.1	<i>Ejecutar y testear una aplicación de Android en Eclipse</i>	78
5.12.2	<i>Configuración de un emulador</i>	78
5.12.3	<i>Configuración de un dispositivo móvil</i>	79
5.12.4	<i>Google USB Driver</i>	80
5.12.5	<i>Descarga del Google USB Driver</i>	80
5.12.6	<i>Uso del dispositivo de hardware</i>	82
5.12.7	<i>Activación de opciones para desarrolladores en el dispositivo</i>	82
5.12.8	<i>Configuración de un dispositivo para el desarrollo</i>	82
5.13	ANIMACIÓN DE APLICACIONES RATIONAL RHAPSODY EN ECLIPSE	85
5.13.1	<i>Preparación para la animación</i>	85
5.13.2	<i>Ejecución de animación</i>	86
5.13.3	<i>La validación de un sistema</i>	86
5.13.4	<i>Verificación del modelo</i>	86
6	ACCELEO	87
6.1.1	<i>Definición</i>	87
6.1.2	<i>Fundamento</i>	87
6.2	HISTORIA Y DESARROLLO	87
6.2.1	<i>La entrada en la fundación Eclipse</i>	88
6.2.2	<i>El código fuente</i>	89
6.2.3	<i>Licencia</i>	89
6.3	COMPATIBILIDAD	89
6.3.1	<i>Los cambios de comportamiento entre las versiones</i>	89
6.3.2	<i>Compatibilidad entre Eclipse y Acceleo</i>	90
6.3.3	<i>Plataformas compatibles</i>	90
6.4	INSTALACIÓN DE ACCELEO	90
6.4.1	<i>Proyecto de Fusión</i>	91
6.4.2	<i>Sitio de actualización</i>	92
6.5	EL LENGUAJE ACCELEO	92
6.5.1	<i>Módulos</i>	92
6.5.2	<i>Plantillas</i>	93
6.5.3	<i>Anulaciones</i>	93

6.5.4	<i>Pre-Condiciones</i>	95
6.5.5	<i>Post-Tratamiento</i>	96
6.5.6	<i>Consultas (Query)</i>	96
6.6	ELEMENTOS DEL LENGUAJE.....	96
6.6.1	<i>Módulos Main y clase Launcher</i>	96
6.6.2	<i>Generación de archivos</i>	98
6.6.3	<i>Estructuras de control</i>	99
6.6.4	<i>Servicios Java</i>	99
6.6.5	<i>Archivos de propiedades</i>	100
6.7	CARACTERÍSTICAS.....	101
6.7.1	<i>Dominio</i>	102
6.7.2	<i>Construido con las herramientas de su elección</i>	102
6.7.3	<i>Tipo de tecnología a generar</i>	102
6.7.4	<i>Resumen de características</i>	102
6.7.5	<i>Características del editor Acceleo</i>	104
6.7.6	<i>El lado oscuro del estándar</i>	107
6.7.7	<i>Depurador</i>	108
6.7.8	<i>Perfilador</i>	108
6.7.9	<i>Trazabilidad (localizar los orígenes de su código generado)</i>	109
6.7.10	<i>Stand-alone</i>	109
6.7.11	<i>Editor reflexivo</i>	110
6.7.12	<i>Bloques de código de usuario</i>	111
6.8	VISTAS DE LA PERSPECTIVA ACCELEO.....	113
6.8.1	<i>Vista Result (muestra la sincronización del modelo, los módulos y el código generado)</i>	113
6.8.2	<i>Vista Overrides (para anular un comportamiento existente en una plantilla)</i> 114	
6.8.3	<i>Vista Generation Patterns (para definir su propia propuesta de finalización)</i> 115	
6.8.4	<i>Vista Interpreter</i>	116
6.9	PROYECTO ACCELEO.....	117
6.9.1	<i>Dependencias</i>	117
6.9.2	<i>Creación de un Proyecto Acceleo "desde cero"</i>	117
6.9.3	<i>La transformación de un proyecto existente en un proyecto Acceleo</i>	119
6.9.4	<i>Creación de un Proyecto Acceleo UI</i>	120
7	COMPARACIÓN DE LAS HERRAMIENTAS	122
7.1	ASPECTOS TEÓRICOS.....	122
7.2	EJEMPLO PRÁCTICO CONCRETO "STOPWATCH".....	125
7.2.1	<i>Implementación con App Inventor 2</i>	125
7.2.2	<i>Implementación con Rational Rhapsody</i>	129
7.2.3	<i>Implementación con Acceleo</i>	132
7.2.4	<i>Conclusiones Prácticas</i>	138
8	CONCLUSIONES Y TRABAJOS FUTUROS	140
8.1	CONCLUSIONES.....	140
8.2	TRABAJOS FUTUROS.....	141
	REFERENCIAS BIBLIOGRÁFICAS	142
	APÉNDICE: GLOSARIO	145

1 Introducción

1.1 Objetivos

El objetivo de la tesina constará en la comparación de varias herramientas de software que permitan generar código fuente Android en forma automática a partir de un modelo definido gráficamente por el usuario. Para esto se trabajará en:

- Adentrarme en el mundo del Desarrollo de Software Dirigido por Modelos (MDD) sus conceptos y particularidades.
- Analizar e investigar herramientas que a partir de modelos generen automáticamente código para aplicaciones Android.
- Obtener las características de estas herramientas.
- Realizar comparaciones entre las herramientas, capturando ventajas y desventajas de las mismas.
- Las herramientas a analizar serán Arctis, ApplInventor, IBM Rational Rhapsody y Acceleo.
- Para algunas herramientas implementar ejemplos, creando modelos, que luego serán transformados a código Android y puedan ser ejecutados tanto en emuladores como en dispositivos móviles.

1.2 Motivación

El proceso de desarrollo de software, como se utiliza hoy en día, es conducido por el diseño de bajo nivel y por el código. En las primeras fases se construyen distintos modelos tales como modelos de requisitos (generalmente escritos en lenguaje natural), los modelos de análisis y los modelos de diseño (frecuentemente expresados mediante diagramas). En la práctica cotidiana, los modelos quedan rápidamente desactualizados debido al atajo que suelen tomar los desarrolladores en el ciclo de vida de este proceso.

El Desarrollo de Software Dirigido por Modelos (MDD) es una alternativa sobresaliente a los métodos convencionales de producción de software, más orientado al espacio de la solución que al espacio del problema.

Mirando hacia atrás podemos ver que ha ocurrido un cambio de foco en el desarrollo de sistemas de software a lo largo de los años. Tempranamente se escribían programas usando lenguajes de bajo nivel (como Assembler). Más tarde se produjo un cambio de perspectiva orientada a escribir programas en lenguajes de alto nivel (como Cobol). El hecho de que estos programas de alto nivel tengan que ser traducidos a programas escritos usando lenguajes de bajo nivel es algo que pasa casi inadvertido.

Los programadores no necesitan preocuparse sobre la generación de código Assembler, ya que esa tarea ha sido automatizada y podemos confiársela al compilador.

Adoptando la propuesta MDD, el foco se desplaza otro nivel más. Hoy en día la actividad principal es desarrollar código, frecuentemente escrito en un lenguaje de programación orientado a objetos. En el futuro, el foco se desplazará a escribir el modelo. La gente olvidará el hecho de que el modelo necesita ser transformado a código ya que la generación de código estará automatizada.

El incremento de los niveles de abstracción y automatización, así como el uso de estándares, realizado correctamente, son todos principios del concepto MDD de innegable utilidad.

Después de muchos años intentándolo, parece que por fin la comunidad de la ingeniería del software acepta que un proceso robusto de producción de software debe estar soportado por modelos conceptuales y dirigido por las transformaciones correspondientes entre modelos definidos de forma precisa. De esta forma, MDD permite reducir los costos de desarrollo de software, adaptarlo rápidamente a los cambios tecnológicos y a cambios en los requisitos, siempre manteniendo la consistencia entre los modelos y el código del software.

Además existen varios ejemplos de aplicaciones exitosas de MDD en grandes proyectos industriales y también herramientas que lo hacen ya realidad a nivel comercial, las cuales intentare describir enfocándome en aquellas que nos permitan crear nuestros propios modelos para luego transformar y generar automáticamente ese modelo al código de la aplicación.

Para acotar y definir aun más el tema me intereso investigar y comparar las herramientas que generen código para aplicaciones que estén destinadas a ejecutarse sobre el actualmente en auge sistema operativo Android.

1.3 Herramientas analizadas

Inicialmente al abordar este trabajo se buscaron herramientas que cumplan con los requerimientos MDD para la generación de código Android. Luego de una tarea de exploración, en la cual se fueron descartando herramientas que han quedado obsoletas o no cumplen los requisitos esperados, se decidió analizar en detalle cuatro herramientas, las cuales se describen en capítulos posteriores: Arctis, App Inventor, IBM Rational Rhapsody y Acceleo.

Además en el capítulo de comparación se decidió desarrollar el mismo ejemplo concreto para algunas de las herramientas de modo que se pueda visualizar claramente sus mecanismos de generación.

2 Conceptos básicos

En este capítulo se describen los conceptos básicos para una mejor comprensión posterior en la descripción y comparación de las herramientas.

2.1 MDD (Model Driven Development)

El desarrollo de software dirigido por modelos MDD (en inglés: model driven software development) se ha convertido en un nuevo paradigma de desarrollo de software, cuyo uso ha venido en creciente aumento durante los últimos años y el mismo pretende mejorar, estandarizar y automatizar la construcción del software basándose en un proceso guiado por modelos y soportado por potentes herramientas. Este paradigma asigna a los modelos un rol central y activo: son al menos tan importantes como el código fuente. Los modelos se van generando desde los más abstractos a los más concretos a través de pasos de transformación y/o refinamientos, hasta llegar al código fuente (dependientes de una plataforma) aplicando una última transformación. La transformación entre modelos constituye el motor de MDD.

Los puntos claves de la iniciativa MDD son los siguientes:

- El uso de un mayor nivel de abstracción en la especificación tanto del problema a resolver como de la solución correspondiente, en relación con los métodos tradicionales de desarrollo de software.
- El aumento de confianza en la automatización asistida por computadora para soportar el análisis, el diseño y la ejecución.
- El uso de estándares industriales como medio para facilitar las comunicaciones, la interacción entre diferentes aplicaciones y productos, y la especialización tecnológica.
- Desarrollo de software se ha convertido en un campo muy diverso donde debe convivir un gran número de plataformas (Windows, Mac OSX, Linux...), dispositivos (PC, teléfonos inteligentes, tabletas...), frameworks (Java, .NET, SAP, Adobe Flash...), canales de distribución (Cloud, Mobile, Browser, lugares de mercado...) y lenguajes (Java, C #, ABAP, Python, C/C++, Objective C...).

Empresas de Software

- Reducir los costos de desarrollo de software, adaptándose rápidamente a los cambios tecnológicos y a cambios en los requisitos, siempre manteniendo la consistencia entre los modelos y el código del software.
- Aumentar la velocidad de desarrollo de software.
- Mejorar la productividad del desarrollador.

Desarrollador

- No hay necesidad de aprender/desaprender una tecnología diferente.

2.2 MDA (Model Driven Architecture)

La arquitectura dirigida por modelos (MDA, del inglés model driven architecture), es una de las iniciativas más conocidas y extendidas dentro del ámbito de MDD. MDA es un concepto promovido por el OMG (Object Management Group) a partir de 2000, con el objetivo de afrontar los desafíos de integración de las aplicaciones y los continuos cambios tecnológicos.

MDA es un conjunto de normas, métodos e ideas pero que en este momento carecen de herramientas eficaces y operativas.

El enfoque MDA está fuertemente centrado alrededor de los modelos y sus transformaciones. Su objetivo es pasar de modelos principalmente documentales a los modelos productivos, realizando transformaciones automáticas de modelos hasta la generación del código que implementa el software.

La idea principal de MDA es utilizar modelos como el núcleo del desarrollo y, de esta forma, separar los aspectos específicos de la plataforma del proceso de desarrollo de software. MDA separa el diseño de la arquitectura y de las tecnologías de construcción, facilitando que el diseño y la arquitectura puedan ser alterados independientemente. El diseño alberga los requerimientos funcionales (casos de uso) mientras que la arquitectura proporciona la infraestructura a través de la cual se hacen efectivos requerimientos no funcionales como la escalabilidad, fiabilidad o rendimiento. MDA se asegura de que el modelo independiente de la plataforma (PIM), el cual representa un diseño conceptual que concreta los requerimientos funcionales, sobrevive a los cambios que se produzcan en las tecnologías de fabricación y en las arquitecturas software.

MDA aporta importantes avances en el control del desarrollo de software y permite, en particular:

- ganancias de productividad
- aumento de la fiabilidad
- mejora significativa de la durabilidad
- mejor capacidad para hacer frente a los cambios

Por lo tanto, el enfoque MDA hace posible que el mismo modelo se implemente en múltiples plataformas a través de las proyecciones estandarizadas, permitiendo a las aplicaciones interoperar con los modelos y apoyar el desarrollo de nuevas plataformas y técnicas.

Los beneficios potenciales de la MDA proviene de la reducción de costos al tener un único código para escribir y mantener, la reducción del tiempo al ser capaz de escribir un código y apuntar a múltiples dispositivos y plataformas, por lo que se promueve la investigación del desarrollo de aplicaciones multiplataforma.

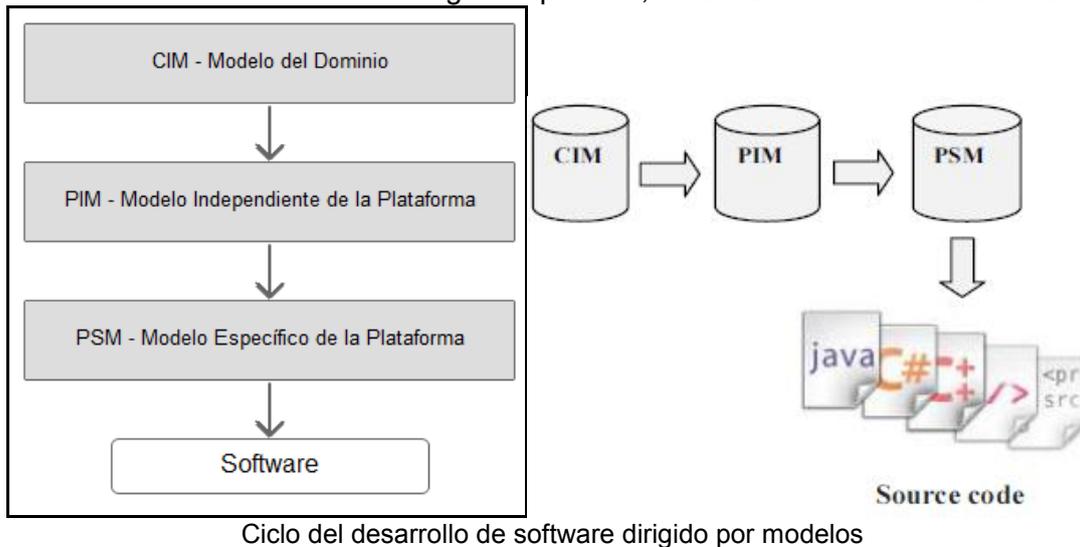
2.3 Modelos

Un modelo es una representación conceptual o física a escala de un proceso o sistema, con el fin de analizar su naturaleza, desarrollar y permitir una mejor comprensión del fenómeno real al cual el modelo representa y permitir así perfeccionar los diseños, antes de iniciar la construcción de las obras u objetos reales.

Con el fin de aclarar los conceptos, el Object Management Group (OMG) ha definido cuatro tipos de modelos:

- **CIM:** El modelo independiente de la computación (en inglés: Computation Independent Model). Un CIM es una vista del sistema desde un punto de vista independiente de la computación. Un CIM no muestra detalles de la estructura del sistema. Usualmente al CIM se lo llama modelo del dominio o de negocio. Se asume que los usuarios a quienes está destinado el CIM (los expertos de dominio) no poseen conocimientos técnicos acerca de los artefactos que se usarán para implementar el sistema.
- **PIM:** El modelo independiente de la plataforma (en inglés, Platform Independent Model). Un PIM es un modelo funcional con un alto nivel de abstracción que es independiente de cualquier tecnología o lenguaje de implementación. Dentro del PIM el sistema se modela desde el punto de vista de cómo se soporta mejor al negocio, sin tener en cuenta cómo va a ser implementado: ignora los sistemas operativos, los lenguajes de programación, el hardware, la topología de red, etc. Por lo tanto un PIM puede luego ser implementado sobre diferentes plataformas específicas.

- **PSM:** El modelo específico de la plataforma (en inglés, Platform Specific Model). Como siguiente paso, un PIM se transforma en uno o más PSMs (Platform Specific Models). Cada PSM representa la proyección del PIM en una plataforma específica. Un PIM puede generar múltiples PSMs, cada uno para una tecnología en particular. Generalmente, los PSMs deben colaborar entre sí para una solución completa y consistente. Por ejemplo, un PSM para Java contiene términos como clase, interfaz, etc. Un PSM para una base de datos relacional contiene términos como tabla, columna, clave foránea, etc.
- **IM:** El modelo de la implementación (en inglés, Implementation Model) es el paso final en el desarrollo es la transformación de cada PSM a código fuente. Ya que el PSM está orientado al dominio tecnológico específico, esta transformación es directa.



2.4 Metamodelos

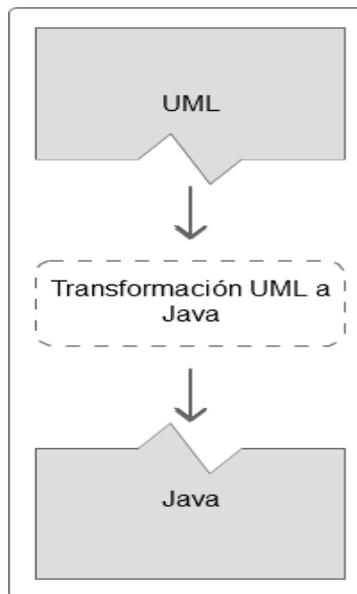
Básicamente, un metamodelo (OMG, 2003) es un modelo que define el lenguaje formal para representar un modelo. En otras palabras, el metamodelo de un modelo describe qué elementos se pueden usar en el modelo y cómo pueden ser conectados. Por ejemplo, el lenguaje UML establece que dentro de un modelo se pueden usar los conceptos clase, atributo, asociación, entre otros.

Como un metamodelo es también un modelo, es necesario que exista un lenguaje para definir metamodelos de manera precisa. En esta instancia aparece el concepto de meta-metamodelo (OMG, 2003) y el mismo es un modelo que define el lenguaje formal para representar un metamodelo. La relación entre un meta-metamodelo y un metamodelo es análoga a la relación entre un metamodelo y modelo.

2.5 Transformación

Una transformación consiste en una colección de reglas, las cuales son especificaciones no ambiguas de las formas en que un modelo, o parte de él, puede ser usado para crear otro modelo. Entonces, decimos que es una herramienta que toma un PIM como entrada y lo transforma en un PSM. La misma herramienta u otra, tomará el PSM y lo transformará a código fuente, por ejemplo. Estas transformaciones son esenciales en el proceso de desarrollo de MDD. Por lo general existen dos tipos de transformaciones:

- Modelo a modelo (M2M): cuyo principal objetivo es el de crear su modelo destino como una instancia de un metamodelo específico. Un ejemplo de esto es la transformación que existe de PIM a PSM.
- Modelo a texto (M2T): el origen es un modelo y su destino es un documento de texto. Un ejemplo de esto es la transformación que se lleva a cabo de PSM a Código Fuente.



Transformación M2T: Modelo UML a Java

2.5.1 OMG (Object Management Group)

El OMG (de sus siglas en inglés Grupo de Gestión de Objetos) es una organización sin fines de lucro, formado en 1989, dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI¹, etc. El grupo está formado por diversas compañías y organizaciones con distintos privilegios dentro de la misma.

2.5.2 MOF y la arquitectura de cuatro capas

El objetivo es el de proveer de un sistema de tipos y una interfaz para que dichos tipos puedan ser creados y manipulados.

En este contexto, el modelo MOF (Meta-Object Facility), se refiere como un meta-metamodelo porque es usado para definir metamodelos como por ejemplo el UML.

MOF está diseñado siguiendo una arquitectura de cuatro capas. Esta es una propuesta de la OMG para organizar y estandarizar los conceptos de modelado, desde los más abstractos a los más concretos.

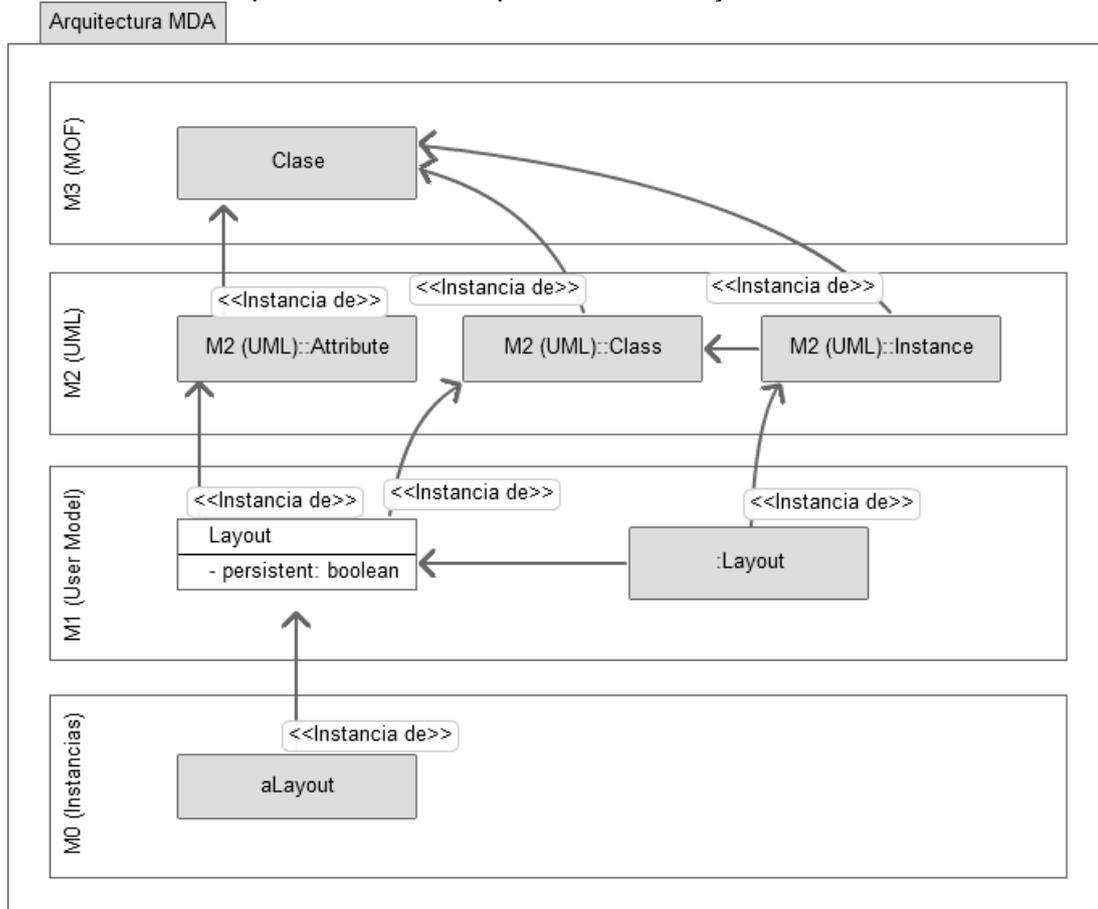
Las arquitectura de cuatro capas se divide en niveles: M3 (MOF), M2 (UML), M1 (User Model), y M0 (Instancias).

- *Meta-metamodelo*: La función principal de esta capa es definir el lenguaje para especificar un metamodelo. Un meta-metamodelo define un modelo en un nivel más alto de abstracción que un metamodelo, y es más compacto que el metamodelo que describe. Un meta-metamodelo puede definir múltiples metamodelos, y puede haber múltiples meta-metamodelos asociados con cada metamodelo.
- Dentro del OMG, MOF es el lenguaje estándar de esta capa. Ejemplos de meta-meta objetos en la capa de meta meta-modelado serían: meta-clase, meta-atributo y meta-operación.
- *Metamodelo*: Un metamodelo es una instancia de un meta-metamodelo. La responsabilidad primaria de esta capa es de definir un lenguaje para especificar modelos. Los metamodelos son típicamente más elaborados que los meta-

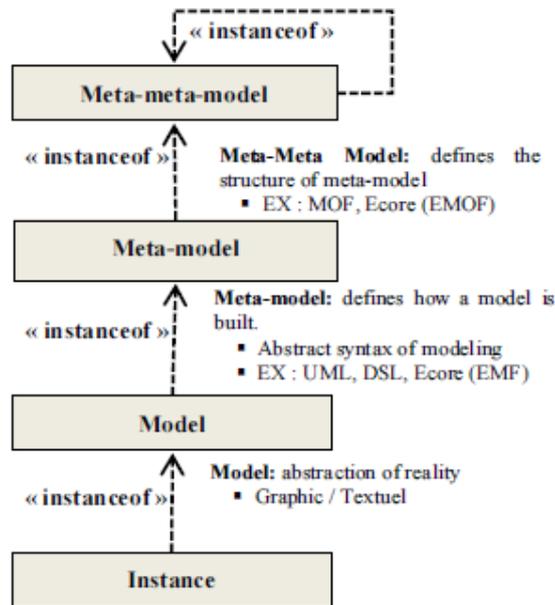
¹ XML Metadata Interchange es una especificación para el intercambio de diagramas. Especifica cómo representar un mode.

metamodelos que los describen, especialmente cuando describen semánticas dinámicas. Ejemplos de meta objetos en la capa de meta-modelado son: clase, atributo, operación y componente.

- **Modelo:** Un modelo es una instancia de un metamodelo. La función principal de la capa de modelo es de definir un lenguaje que describa el dominio de la información. Ejemplos de objetos en esta capa son: Auto (Clase), Modelo (Atributo), arrancar (Operación).
- **Objetos de usuario:** Los objetos de usuario son las instancias de un modelo. La responsabilidad principal de la capa de objetos de usuario es de describir un dominio de información específico. Se utiliza para describir objetos del mundo real.



Arquitectura de cuatro capas



2.5.3 MOFM2T

MOFM2T o Lenguaje de transformación de modelo a texto (MTL) basado en MOF, es una especificación de la OMG para la transformación de modelos a lenguajes de programación. Puede ser usado para expresar transformaciones que transforman un modelo a texto (M2T). Éste reusa algunos conceptos de MOF. El lenguaje utilizado en Aceleo es una implementación de esta norma.

2.6 Sistema Operativo Android

Android es un sistema operativo basado en el Kernel de Linux, diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes o tabletas, y también para relojes inteligentes, televisores, automóviles, etc. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005, compró.

2.6.1 Características

- La plataforma es adaptable a pantallas de distintas resoluciones.
- Cuenta con una base de datos liviana (SQLite) para almacenamiento.
- Soporte para mensajería, formatos multimedia, streaming, multi-táctil, conectividad bluetooth Wi-Fi, etc., videollamadas, puede ejecutar multitareas, búsquedas en Google a través de voz, tethering, contiene navegador web y servicio de Google Play.
- Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma. El bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y corre en la Máquina Virtual Dalvik. Dalvik es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.
- Soporte hardware para cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, sensores de proximidad y de presión, sensores de luz, termómetro, aceleración por GPU 2D y 3D, etc.
- Inicialmente el IDE utilizado era Eclipse con el plugin ADT. Ahora se considera como entorno oficial Android Studio. Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software.

2.6.2 Arquitectura

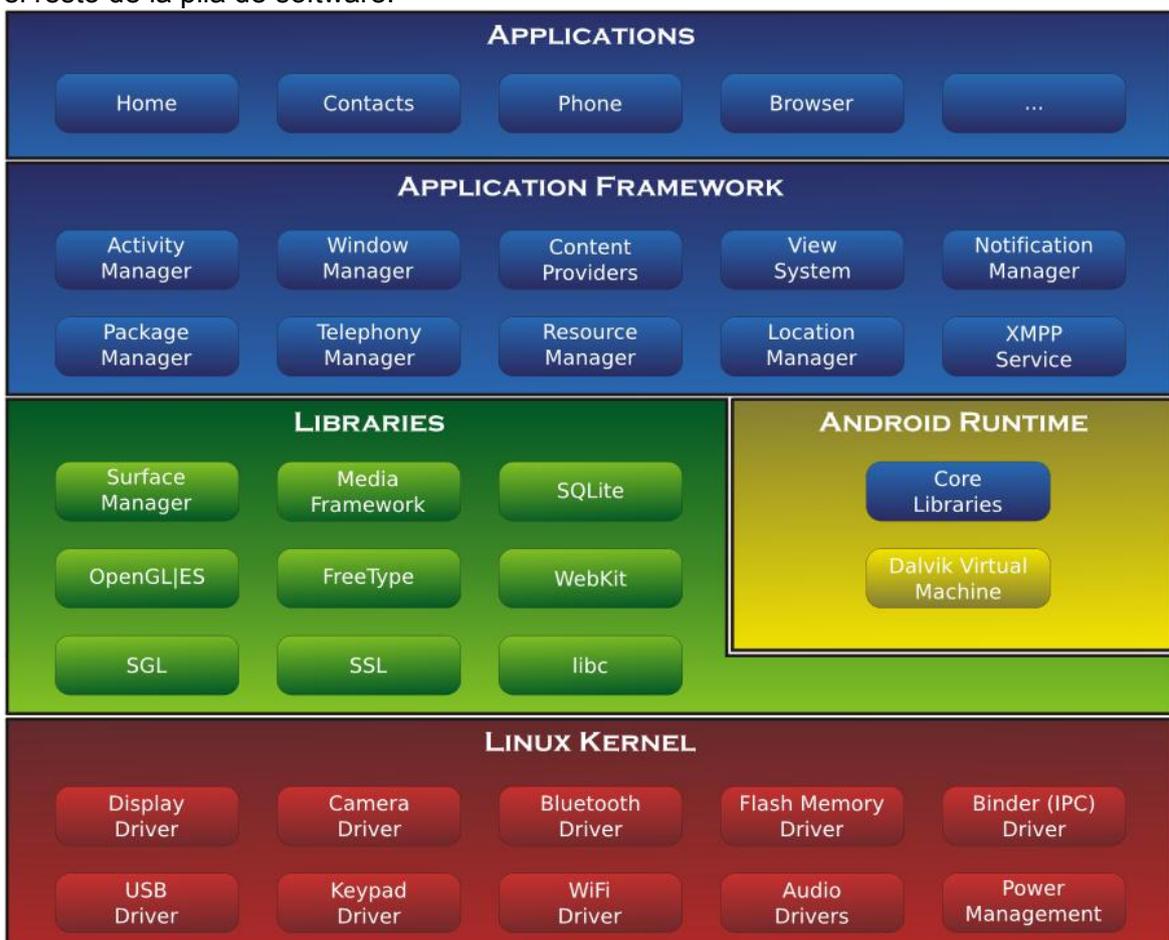
Aplicaciones: Las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.

Framework de aplicaciones: Los desarrolladores tienen acceso completo a los mismos APIs del framework usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.

Bibliotecas: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema.

Runtime de Android: Android incluye un conjunto de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual. Esta máquina virtual denominada Dalvik ha sido escrita de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecuta archivos en el formato Dalvik Executable (.dex), el cual está optimizado para memoria mínima.

Kernel Linux: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El kernel también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.



Arquitectura de SO Android

2.6.3 SDK de Android

Desarrollador	Google
Última versión estable	R24.0.2
Programado en	Java
Sistema operativo	Multiplataforma
Licencia	Gratuita. Acuerdo con Google

El SDK de Android es un kit de desarrollo de software que incluye todas las bibliotecas y los archivos necesarios para los desarrolladores para hacer aplicaciones para Android. El kit también incluye un depurador de código, un simulador de teléfono basado en QEMU, documentación, ejemplos de código, tutoriales y el Administrador de dispositivos virtuales de Android, que permite probar la aplicación en cualquier versión de Android (o Android Wear) que desee.

La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente es Android Studio junto con el complemento ADT (Android Development Tools plugin).

Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad. Una aplicación Android está compuesta por un conjunto de ficheros empaquetados en formato .apk y guardada en el directorio /data/app del sistema operativo Android (el cual necesita permisos de superusuario, root, por razones de seguridad). Un paquete APK incluye ficheros .dex (ejecutables Dalvik, un código intermedio compilado), recursos, etc.

2.6.4 Android Development Tools (ADT)

Tenemos dos programas que son individuales, nada vinculados, por un lado el SDK de Android y por otro el entorno de desarrollo Eclipse, que si lo abrimos una vez instalado el SDK, veremos que no hay nada nuevo. Lo que quiere decir que tenemos que vincularlo para que se reconozcan entre ellos. Esto se realiza mediante la instalación de un plugin para Eclipse denominado ADT. Al instalar el plugin ADT, se vinculan Eclipse con el SDK de Android para poder instalar herramientas (como Google USB Driver) y las diferentes versiones de Android en Eclipse. Esto nos permite interactuar con dispositivos móviles y crear una gran variedad de emuladores con diferentes características a través del AVD Manager (Android Virtual Device Manager).

2.6.5 Android API Level

API Level (Nivel de API) es un valor entero que identifica de forma exclusiva una revisión de la API que ofrece una versión de la plataforma Android. La plataforma Android proporciona una API que las aplicaciones pueden utilizar para interactuar con el sistema Android subyacente.

A continuación se mencionan las versiones de Android más destacadas, junto con su versión y API Level:

Versión	API Level	Código o Nombre
Android 1.6	4	Donet
Android 2.3.2	9	GingerBread
Android 3.0	11	HoneyComb
Android 4.0	14	Ice Cream Sándwich
Android 4.1	16	Jelly Bean

Android 4.4	19	KitKat
Android 5.0/5.1	21/22	Lollipop
Android 6.0	23	Marshmallow
Android 7.0	24	Nougat

3 Arctis

Arctis es el resultado de la investigación en el Departamento de Telemática de la Universidad Noruega de Ciencia y Tecnología (NTNU), y se desarrolla por un proyecto nacido como extensión del equipo Bitreactive. Arctis es de código cerrado, pero gratuito para uso no comercial.

Arctis es una herramienta que le ayuda a desarrollar, analizar y generar aplicaciones Java reactivas gráficamente, conectando conjuntamente bloques de construcción.

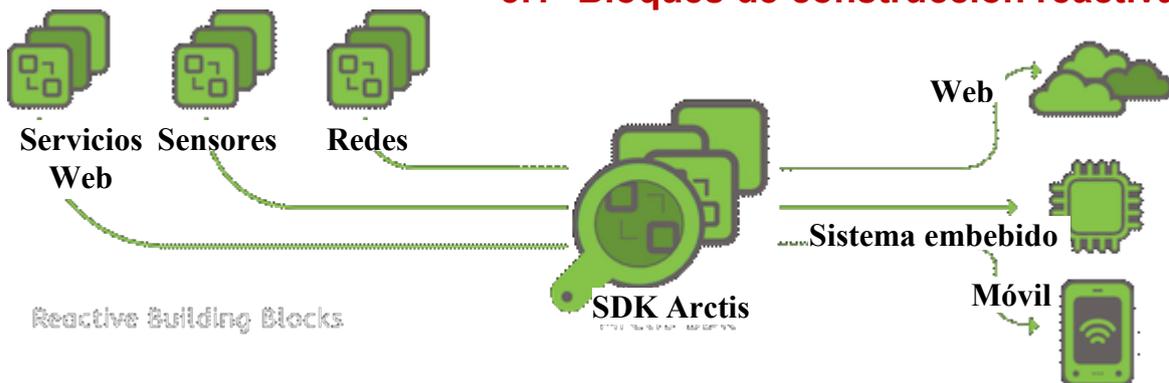
Los bloques de construcción son una forma de agrupar y organizar código Java que se puede conectar entre sí.

Arctis contribuye a la construcción de bloques de construcción de módulos reutilizables. Varias soluciones han sido desarrolladas en las herramientas Arctis que cubren varios ámbitos, como los servicios móviles, sistemas embebidos, domótica o automatización del hogar, gestión de confianza y servicios Web.

Arctis es un SDK que consta de tres herramientas que están estrechamente integradas entre sí como plugins de Eclipse:

- **Editor de Arctis:** es un editor gráfico y textual combinado para crear y combinar bloques de construcción. Junto con el analizador, el editor también puede animar el comportamiento de las especificaciones.
- **Analizador de Arctis:** toma bloques de construcción y comprueba los errores y defectos de diseño. Se hace un diagnóstico sobre la situación y sugiere algunas mejoras al caso.
- **Compilador de Arctis:** toma bloques de construcción del sistema y crea proyectos de Eclipse con código ejecutable para diferentes plataformas, dependiendo del compilador que se utiliza. Una vez más, esto funciona completamente automatizado, y usted no tiene que cambiar nada en el código generado.

3.1 Bloques de construcción reactiva



El uso de los servicios de Android significa que las aplicaciones tienen que ser reactivas. La interfaz de usuario, por ejemplo, organiza eventos sobre toda acción por parte del usuario, el servicios de localización puede actualizar un estado en cualquier momento, o las aplicaciones pueden ser notificadas por otros servicios disponibles en el teléfono.

Un desarrollador de aplicaciones se enfrenta a la compleja tarea de coordinar los comportamientos que todas estas interfaces tienen. Hay aplicaciones que son difíciles de codificar correctamente, sobre todo cuando son reactivas y tienen que manejar muchos eventos al mismo tiempo. Incluso si te las arreglas programando, el código de estas

aplicaciones es a menudo difícil de entender para sí mismo y para los demás, por no hablar de mantener y actualizar más tarde.

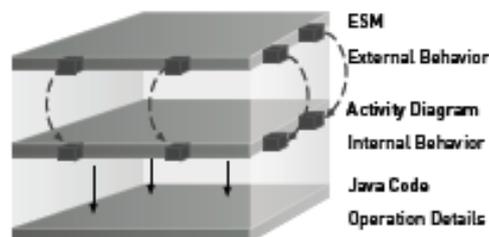
En Arctis, las aplicaciones reactivas se modelan gráficamente, utilizando los flujos de datos bien conocidos (actividades UML). Los modelos de flujo de datos se organizan en bloques especiales, que pueden ser seleccionados a partir de librerías que soportan diversas tecnologías y dominios de aplicación. Hasta el 70% de un sistema puede venir de estas librerías reutilizables.

Los bloques reactivos SDK son un nuevo tipo de módulo de software, ellos generan automáticamente código optimizado, orientado a eventos, altamente concurrente por parte de la construcción, y se ejecutan de manera eficiente a partir de la combinación de bloques de construcción.

Además, estos bloques reactivos permiten la verificación automática. Son formados por el código del programa, integrado en un lenguaje gráfico que se ocupa de la conducta concurrente, sincronizaciones, los flujos de datos y conexiones a otros bloques. Con esta claridad, también la adaptación y mantenimiento de aplicaciones se vuelve mucho más fácil.

Los bloques reactivos se encapsulan y acoplan estrechamente:

- **código Java** para describir con detalle las operaciones.
- un **diagrama de actividad** para representar gráficamente el comportamiento interno de un bloque de construcción.
- una **máquina de estado externo** (ESM) que será descrita más adelante.



El editor gráfico crea bloques de construcción y aplicaciones completas utilizando los flujos de datos basados en eventos. Este editor Arctis gestiona todos los diagramas y archivos de un bloque de construcción.

Usted puede editar el contenido de las operaciones en el entorno familiar de Eclipse, incluyendo todo el soporte para Java que está acostumbrado. Modelos y códigos se mantienen sincronizados automáticamente.

Los bloques combinan los flujos de datos gráficos y de código. El flujo de datos se encarga del comportamiento concurrente y sincronizaciones.

Arctis, una herramienta basada en Eclipse utilizada para los sistemas y servicios de diseño, utiliza máquinas de estado en tiempo de ejecución como un medio para manejar la concurrencia. Estas máquinas de estado se generan automáticamente a partir de bloques de construcción.

La coordinación del comportamiento concurrente es uno de los puntos fuertes de la herramienta Arctis, en el que las aplicaciones se especifican por medio de diagramas de actividad UML. Estas actividades se construyen por la composición de bloques de construcción que encapsulan ciertas funcionalidades. El código ejecutable se genera de forma automática a través de una transformación dentro de máquinas de estado.

3.2 Máquina de estado externo (ESM)

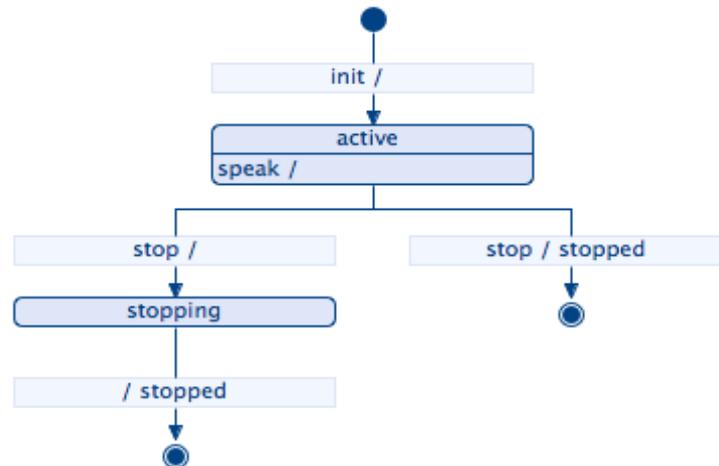
Las máquinas de estado externos se utilizan para describir el contrato entre un bloque reactivo y su entorno asegurando el uso correcto de este bloque. Una ESM describe la secuencia en la que las características de un bloque reactivo se pueden utilizar y por lo tanto proporcionan una API fuerte para el bloque.

La ESM de un bloque se programa mediante la cumplimentación de una tabla con todas las transiciones legales.

External Behavior (ESM)		
Source State	Parameter Nodes	Target State
active	stop /	stopping
stopping	/ stopped	<finals3>
<initial>	init /	active
active	stop / stopped	<finals1>
active	speak /	active

Estas interacciones legales se denominan transiciones e incluyen:

- el estado actual del bloque (estado original).
- la entrada en cuestión y los pines de salida (nodos parámetros).
- el siguiente estado del bloque (estado del objetivo).



3.3 Tipos de bloques

■ Bloques del Sistema

Especifican una aplicación en el nivel de descomposición más alta. El sistema no tiene ninguna entrada o parámetro de salida, pero si nodos iniciales para comenzar y nodos de fin de actividad para la terminación. Para implementar sistemas, es necesario este tipo de bloque como elemento de nivel superior.

■ Bloques locales

Son similares a las funciones que se ejecutan en un solo componente para hacer algún trabajo. La mayoría de los bloques de construcción de una aplicación son de este tipo. “Local” es relativo con el diagrama de actividad: las operaciones también pueden invocar código en otras máquinas. Una llamada HTTP a una máquina remota puede ser representada por un bloque local.

■ Bloques superficiales

Se utilizan para coordinar los flujos en los diagramas de actividad de una manera más potente de lo que se permite con la unión de los nodos de actividad. Formalmente se comportan como máquinas de estado. En contraste con los bloques locales, que constan de un ESM y una actividad que sólo tiene nodos parámetros.

■ Bloques de colaboración

Especifican el comportamiento de varios de los participantes con el fin de cumplir con una tarea determinada. La consulta de algunos datos de algún servidor, por ejemplo, puede ser descrito como bloques de colaboración.

Tenga en cuenta que las colaboraciones son una característica experimental no se utiliza para la producción. Las colaboraciones deben estar activadas en las preferencias.

	Collaboration (Structure)	Activity (Internal Behavior)	ESM (External Behavior)	Java Code (Operations)
System Block	yes	yes	no	yes
Collaborative Block	yes	yes	yes	yes
Local Block	no	yes	yes	yes
Shallow Block	no	frame only	yes	no

3.4 Beneficios

★ Crear aplicaciones mediante la combinación de bloques de construcción:

Una aplicación consiste en una jerarquía de bloques reactivos, que están anidados unos dentro de otros representando diferentes niveles de detalle. En cada nivel, los bloques están conectados unos con otros (y con nodos de actividad, los elementos utilizados en los diagramas de actividad) por los flujos para representar el flujo de datos dentro del sistema, que puede ser secuencial, ramificado o concurrente. Un sistema está diseñado mediante el modelado de su flujo de datos gráficamente. Todo esto hace que sea posible para entender y resolver un problema de una vez y para siempre, donde la solución estará disponible como un bloque de construcción encapsulado dedicado y listo para ser reutilizado.

★ Reutilizar, Reciclar, Reutilizar:

Los bloques son como una interfaz de programación (API), encapsulan soluciones a los problemas en forma independiente y además describen las características de secuencia en la que tienen que ser utilizados. Esto los convierte en unidades ideales de reutilización. Según la aplicación, se puede esperar que hasta el 70% de su sistema se originen en las librerías existentes.

★ Use una poderosa combinación de gráficos y códigos:

Las aplicaciones complejas se descomponen gráficamente en bloques de construcción, los cuales permiten comprender de forma fácil y rápida lo que está pasando. Los gráficos y el código están alineados sin redundancia, y las tareas que se resuelven mejor mediante la programación son codificadas.

★ Muestre sus sistemas a los demás

Las especificaciones de Arctis muestran lo que sucede en un sistema, y es una base ideal para discusiones técnicas. Además estas especificaciones pueden ser entendidas por personas no técnicas o utilizarlas para mostrar y explicar a los jefes de equipo o incluso clientes lo que se está trabajando.

★ Mantenerse organizado

Los bloques de construcción son unidades independientes de trabajo que combinan todos los artefactos en un solo lugar, con su interfaz, la documentación y los datos necesarios para su implementación.

★ **La labor se centró, bloque por bloque**

Los bloques de construcción son la unidad fundamental de desarrollo. Esto significa que los problemas de diseño son tratados por bloques separados, y de esta manera poder concentrarse en una sola tarea.

★ **Comprobar su trabajo con animación y análisis automático**

Con los diagramas de Arctis puede detectar errores de forma automática, antes que el código esté escrito. En el análisis automático se encuentran todo tipo de defectos de diseño intrincado que un compilador tradicional para la programación se perdería. Encontrar bloqueos mutuos, errores en las sincronizaciones, las condiciones de ejecución y un montón de otras conductas extrañas. Los bloques de construcción que se utilizan están protegidos por contratos, por lo que realmente no se pueden utilizar de manera incorrecta. Además los tipos de errores en Arctis son vistos, especialmente en cuanto a la concurrencia y las interacciones constantes, un área difícil de analizar con otras herramientas. Para asegurarse de que su sistema está haciendo lo que usted pretende, también puede simular los bloques por animación interactiva para buscar situaciones que nunca pensó.

★ **Colaborar como un equipo de ingeniería**

Utilice diferentes bloques para distribuir las tareas entre un grupo de ingenieros, cada uno responsable de su propio conjunto de bloques. Debido a sus interfaces, es posible integrar los bloques de los demás, incluso antes de que estén completamente terminadas, y todo funcionará como se espera.

★ **Deje a Arctis hacer la aplicación**

Arctis crea automáticamente implementaciones de bloques de construcción, el cuidado de la programación eficiente y un montón de cosas aburridas. Dependiendo del tipo de aplicación, se puede ahorrar hasta un 60 % del código escrito, y al hacer uso de bloques de construcción de las librerías existentes, esta ración se puede ir hasta el 90 %. Esto significa que sólo se tiene que escribir el 10 % del código de su aplicación.

3.5 Instalando Arctis en Eclipse

Arctis se puede instalar desde el servidor de actualizaciones:

1. Abrir el Administrador de software a través de Help | Install New Software...

2. Agregar un nuevo sitio de actualizaciones y autenticarse:

<http://www.item.ntnu.no/~kraemer/arctis/updates-alpha>

3. El sitio de actualización Arctis es un repositorio de material compuesto, que une también a otros sitios de actualización.

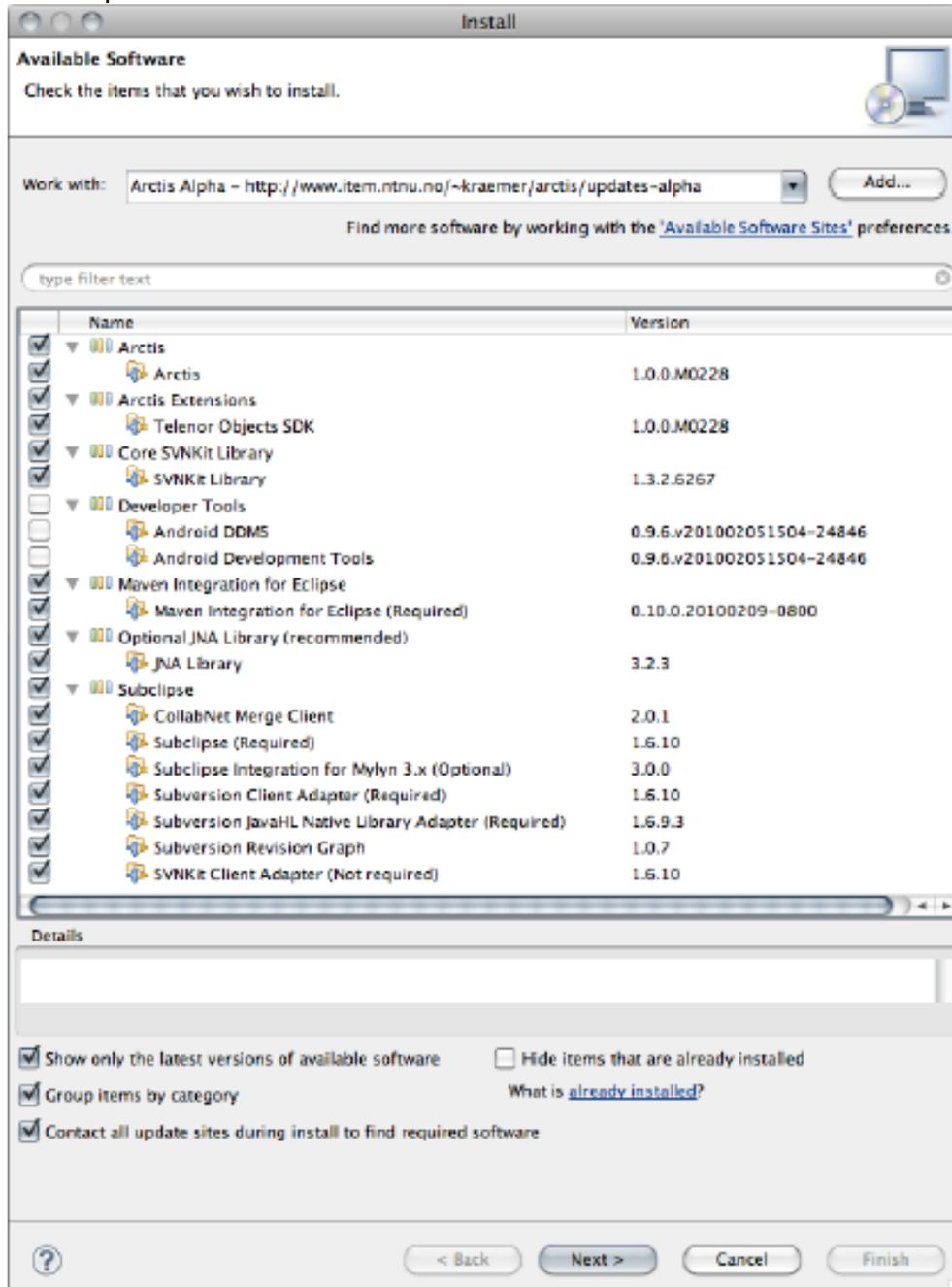
Dependiendo de su plataforma de desarrollo, puede seleccionar diferentes características:

- Arctis (obligatorio).
- Soporte SVN Subclipse (obligatorio).
- Arctis para conectar objetos (opcional).
- Maven para Eclipse (necesario para objetos conectados).
- Arctis para Android (opcional).
- Android Development Tools (opcional, requiere la instalación del SDK de Android).

Si estas características no aparecen, deseleccionar el check "Group items by category".

Si algunas dependencias no se encuentran, asegúrese de que el check “Contact all update sites” esta seleccionado.

4. Reinicie Eclipse.



3.6 Construyendo una simple aplicación

La construcción de una primera aplicación:

1. Comprobar las librerías existentes

Navegar a través de las librerías públicas que figuran en www.arctis.item.ntnu.no y encontrar bloques con funciones útiles para la aplicación. Basta con arrastrar todas las librerías que necesita en la Vista de Bloques de Construcción.

2. Crear un nuevo sistema

Utilice el Asistente Arctis para crear un nuevo sistema.

Dependiendo de su dominio, se puede seleccionar una plantilla existente.

3. Arrastre los bloques importados

Arrastrar los bloques que usted necesita hacia su nuevo sistema creado. Algunos de ellos tienen los parámetros de instancia que hacen posible su adaptación.

4. Agregar un poco de lógica para enlazar los bloques

Conectar bloques con bordes y otros elementos. En algunos casos, es posible añadir operaciones Java para trabajar con los datos o hacer otras cosas.

5. Deje a Arctis analizar el sistema

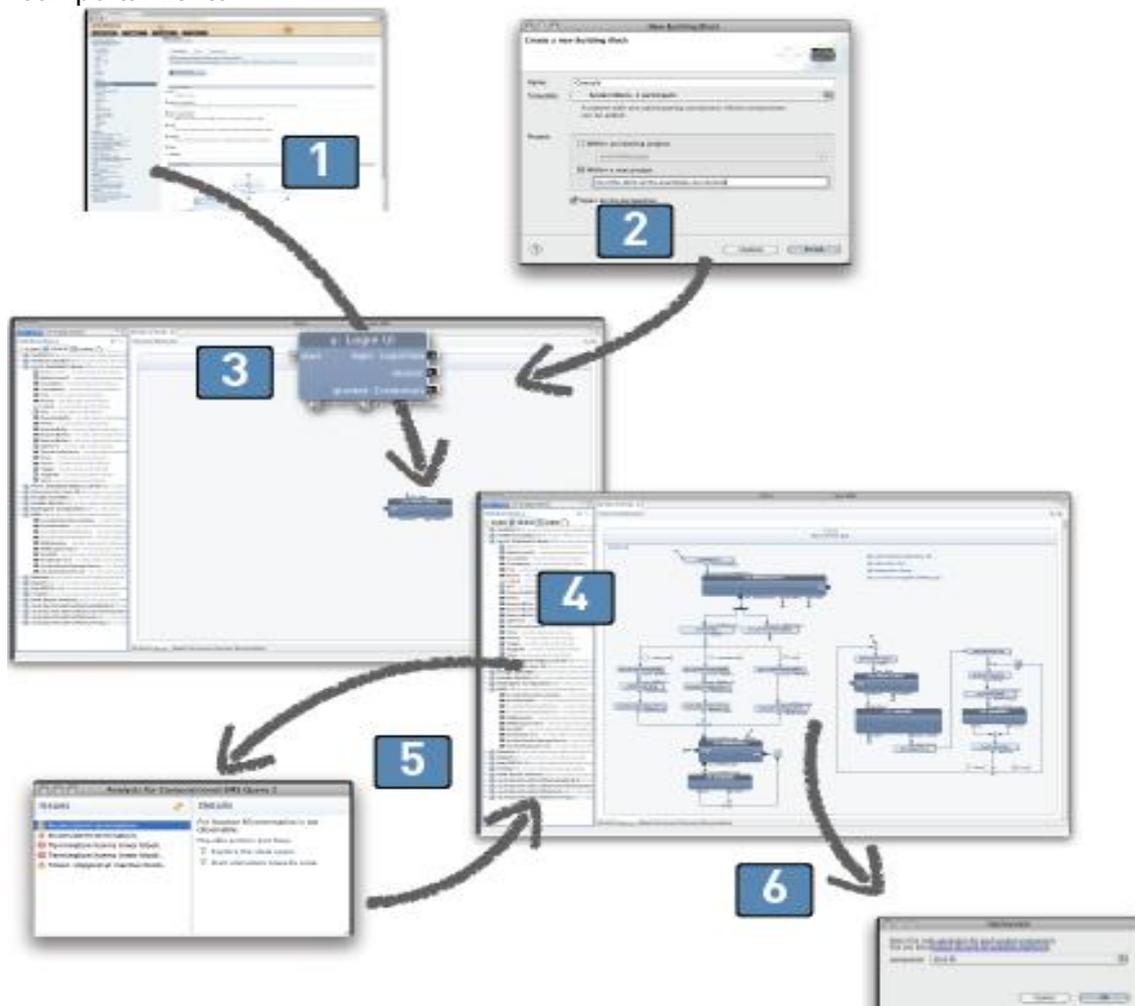
Arctis analiza su sistema para detectar problemas y errores, para después sugerir algunas mejoras al caso.

Vuelva al paso 4 si los cambios deben hacerse.

6. Implementar automáticamente

Generar código de la aplicación que conecta los bloques de construcción importados.

Los casos más avanzados incluyen crear sus propios bloques de construcción y dominar el comportamiento.

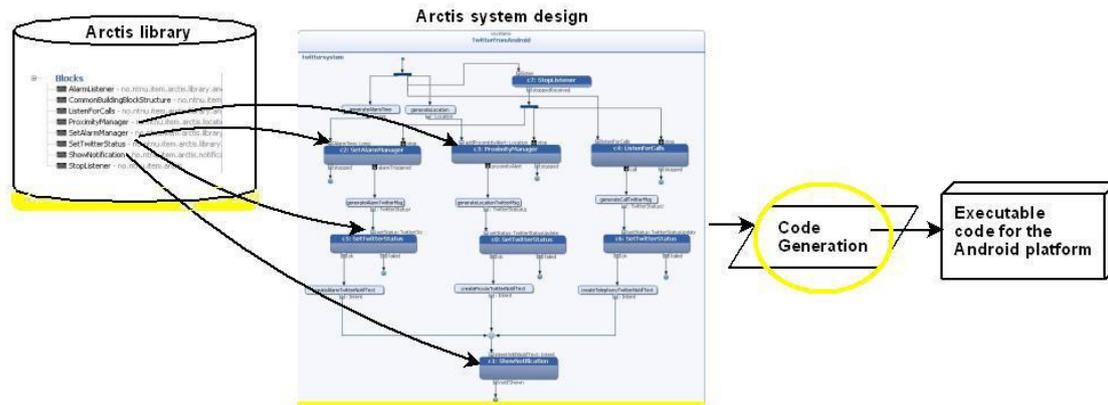


3.7 Tesis: “Desarrollo de aplicaciones Android con Arctis”

A continuación se describe una tesis desarrollada en la NTNU, sobre la herramienta Arctis, donde se propuso como objetivo principal el diseño de aplicaciones en Arctis para la plataforma Android.

En este trabajo se utilizan los bloques de construcción de Arctis y un generador de código en el desarrollo de aplicaciones para Android, como puede verse en la siguiente figura.

Una biblioteca Arctis contiene bloques de construcción que acceden a los servicios Android o realizan otras tareas que se utilizan en el desarrollo de diseños de sistemas para aplicaciones de Android. A partir de aquí, un generador de código se utiliza para generar código Android ejecutable para la aplicación diseñada.



Desarrollo de aplicaciones con Arctis para Android

Arctis disponía de una opción de implementación para Java SE, que permitía la implementación de diseños de sistemas Arctis en código Java ejecutable. Este código es capaz de ejecutar en un dispositivo Android, pero como el proyecto generado aún no es un proyecto Android, son necesarias adaptaciones, las cuales fueron documentadas.

Para evitar tener que hacer adaptaciones manuales cada vez que una aplicación Android se va a generar, se utilizaron las adaptaciones documentadas en la creación de un generador de código Java SE para Android.

3.7.1 Descripción abstracta

Se ha realizado una tesis donde analizan cómo Arctis se pueden utilizar para el desarrollo de aplicaciones Android. El generador de código existente para Java fue ampliado para producir aplicaciones de Android. Se han investigado y propuesto los mecanismos para encapsular los servicios de Android proporcionados por la API del teléfono, como bloques de construcción en Arctis.

Se presenta una discusión sobre los problemas, las soluciones y arquitecturas para el diseño e implementación de aplicaciones Android usando Arctis. Se han rediseñado aplicaciones utilizando bloques de construcción Arctis y desplegado como un proyecto Java utilizando el generador de código existente para Java SE. Las adaptaciones necesarias para convertir el proyecto Java en un proyecto Android ejecutable se estudió en detalle y sirvió como base para el desarrollo del generador de código. Después de describir el generador de código, varios bloques de construcción están diseñados para una biblioteca de bloques de construcción de Android.

3.7.2 Contribución

Las partes resaltadas de la figura anterior representa la contribución del trabajo realizado con Arctis. El editor de Arctis fue equipado con una librería de bloques de construcción

Android las cuales contienen todos los bloques de desarrollo para su uso por aplicaciones de Android.

Dos aplicaciones de ejemplo están diseñadas usando actividades UML, bloques de construcción existentes y bloques de construcción desarrollados para utilizar por Android. Un nuevo generador de código es desarrollado, usando piezas del generador SE Java existentes en la actualidad, y la adición de la funcionalidad para la generación de proyectos de Android. Se añadió una nueva opción de despliegue para Arctis, Java SE para Android, que utiliza el generador de Android en la creación de un proyecto Android. Este proyecto es capaz de ejecutarse en el emulador de Android o dispositivo móvil.

3.7.3 Conclusión

El principal problema de discusión fue la forma de resolver problemas relacionados con el planificador de tiempo, que se ejecuta en un subproceso independiente de los componentes de Android.

Hay algunas desventajas para la arquitectura propuesta en cuanto a las aplicaciones que utilizan estos nuevos bloques de construcción los cuales acceden a subprocesos que dependen de los servicios Android. Las soluciones a los problemas se discutieron y propusieron. Los bloques de construcción que acceden por defecto a los servicios de Android no exigen de un subproceso Android que puede ser diseñado sin alternancia del generador de código creado.

El método para ajustar el generador de código fue diseñar primero algunos bloques de construcción que componen una aplicación de Android en Arctis e implementarlo utilizando el generador de SE Java existente. Las adaptaciones necesarias para permitir a la aplicación generada ejecutarse en un dispositivo Android se estudiaron y observaron. El proceso de adaptación a lo largo de la discusión arquitectónica hizo la base para el desarrollo del nuevo generador de código Java SE para Android.

Este trabajo ha dado como resultado el editor Arctis equipado con una opción de despliegue para aplicaciones de Android llamada Java SE para Android. Se desarrollo y desplegó con éxito dos aplicaciones para la plataforma Android usando Arctis y el Android para el generador de código de Java SE.

HelloLocationWorld es una aplicación de seguimiento de localización simple que se utiliza en la descripción de la plataforma Android y el descubrimiento de ajustes. TwitterFromAndroid es una aplicación más completa que utiliza bloques de construcción de la nueva biblioteca y las habilidades de modelado proporcionadas por Arctis en la creación de una aplicación de Twitter para el dispositivo Android. Las actualizaciones de Twitter para una cuenta determinada automáticamente se realizan como resultado de los acontecimientos en el dispositivo Android.

4 App Inventor

Desarrollador	Google Labs y MIT
Última versión estable	App Inventor 2 (06/12/2013)
Licencia	Freeware
Estado actual	Actualizando mensualmente

Google App Inventor es una aplicación de Google Labs y mantenida ahora por el Instituto de Tecnología de Massachussets (MIT) para crear aplicaciones de software para el sistema operativo Android. Utiliza una interfaz gráfica, que permite a los usuarios arrastrar y soltar objetos visuales para crear una aplicación. De forma visual y a partir de un conjunto de herramientas básicas, el usuario puede ir enlazando una serie de bloques para crear la aplicación. Está dirigida a personas que no están familiarizadas con la programación informática. Esto se debe a que en lugar de escribir código, se diseña visualmente la forma en que la aplicación se ve y utiliza bloques para especificar el comportamiento de la aplicación.

El sistema es gratuito y se puede descargar fácilmente de la Web. Las aplicaciones de App Inventor están limitadas por su simplicidad, aunque permiten cubrir un gran número de necesidades básicas en un dispositivo móvil.

App Inventor le permite desarrollar aplicaciones para los teléfonos Android con un explorador Web y, o bien un teléfono o un emulador conectado. Los servidores de App Inventor almacenan su trabajo y ayudan a mantener un registro de sus proyectos.

Con App Inventor, se espera un incremento importante en el número de aplicaciones para Android debido a dos grandes factores: la simplicidad de uso, que facilitará la aparición de un gran número de nuevas aplicaciones; y Google Play, el centro de distribución de aplicaciones para Android donde cualquier usuario puede distribuir sus creaciones libremente.

4.1 Historia

La aplicación se puso a disposición de los usuarios, mediante invitación, el 12 de julio de 2010, el 15 de diciembre de 2010 se puso a disposición de usuarios registrados.

Google puso fin al desarrollo el 31 de diciembre de 2011 cediéndole el código al MIT, quién lo ha puesto a disposición de todos en su versión Classic en marzo de 2012.

En la creación de App Inventor, Google se basó en investigaciones previas significativas en informática educativa. Fue creada a mediados del 2009 por el profesor Harold Abelson del MIT. Antes de salir al mercado se ha probado en diferentes centros educativos y la han utilizado desde niños de 12 años hasta licenciados universitarios sin nociones de programación.

El 6 de diciembre de 2013 MIT lanzó App Inventor 2.

MIT App Inventor 1 (Classic) ya no está disponible para la creación de aplicaciones. Puede seguir teniendo acceso a los antiguos proyectos de la aplicación en el sitio Web App Inventor. Se pueden descargar pero no se pueden modificar o crear nuevos proyectos Classic App Inventor. Para trabajar con App Inventor, se utilizará el MIT App Inventor 2. App Inventor clásico se cerró el 15 de julio de 2015. Para migrar los proyectos desde App Inventor Classic a App Inventor 2 se usa el convertidor del siguiente sitio web: <http://convert.appinventor.mit.edu/>

4.2 Estructura

Al construir aplicaciones se esta trabajando con:

El App Inventor Diseñador, donde se seleccionan los componentes para su aplicación. El App Inventor Editor de Bloques, donde se ensamblan bloques de programa que especifican cómo los componentes deben comportarse. Se ensamblan los programas en forma visual, piezas de montaje como piezas de un rompecabezas.

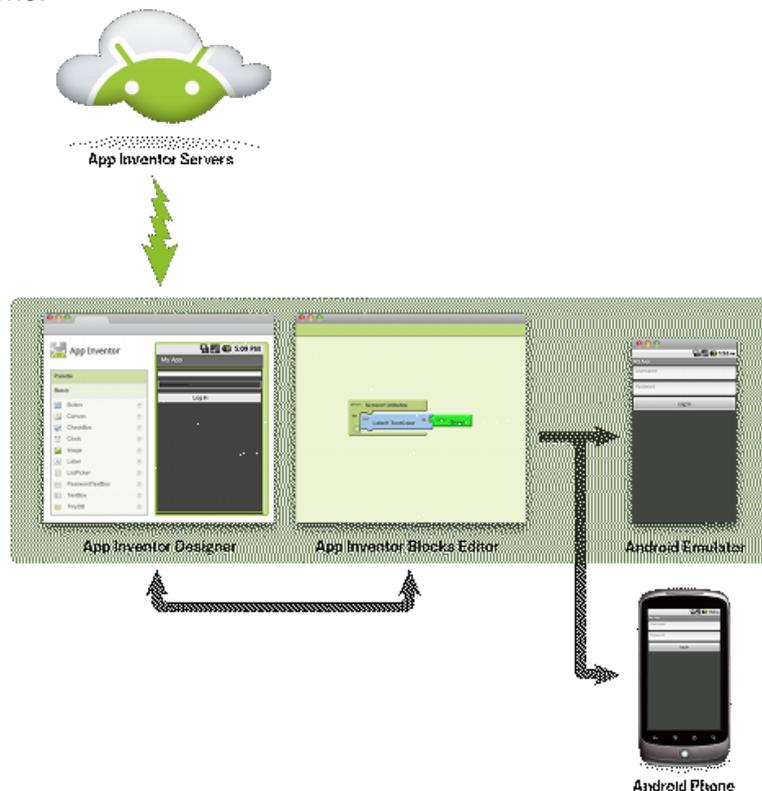
El editor de bloques de la aplicación utiliza la librería Open Blocks de Java para crear un lenguaje visual a partir de bloques. Estas librerías están distribuidas por MIT bajo su licencia libre (MIT License). El compilador que traduce el lenguaje visual de los bloques para la aplicación en Android utiliza Kawa como lenguaje de programación, distribuido como parte del sistema operativo GNU de la Free Software Foundation.

4.3 Más sobre MIT App Inventor

La creación de una aplicación de App Inventor empieza en su navegador, en el que diseño el aspecto que tendrá la aplicación. Entonces, como encajando las piezas del rompecabezas, se establece el comportamiento de su aplicación. Al mismo tiempo, a través de una conexión en directo entre el ordenador y el teléfono, la aplicación aparece en el teléfono.

Su aplicación aparece en el teléfono paso a paso a medida que añada piezas a la misma, por lo que puede poner a prueba su trabajo a medida que construye. Cuando haya terminado, usted puede empaquetar la aplicación y producir una aplicación independiente para instalar.

Si usted no tiene un teléfono con Android, puedes construir tus aplicaciones utilizando el emulador de Android, el software que se ejecuta en su computadora y se comporta como el teléfono.



Figura

4.4 Requisitos del sistema para App Inventor

Ordenador y sistema operativo

Macintosh (con procesador Intel): Mac OS X 10.5 o superior.

Windows: Windows XP, Windows Vista, Windows 7 o superior.

GNU/Linux: Ubuntu 8 o superior, Debian 5 o superior.

Navegador

Mozilla Firefox 3.6 o superior.

Nota: Si está utilizando Firefox con la extensión NoScript, usted necesita deshabilitar la extensión.

Apple Safari 5.0 o superior.

Google Chrome 4.0 o superior.

Microsoft Internet Explorer 7 o superior.

Nota: Para App Inventor 2 no es soportado todavía.

Teléfono o tablet (o el emulador en pantalla)

Sistema operativo Android 2.3 ("Gingerbread") o superior.

4.5 Configuración de App Inventor 1

Prepararse para programar con App Inventor es un proceso sencillo que varía en función de si está desarrollando aplicaciones en un dispositivo Android o en la función de emulador teléfono Android (que se instala junto con App Inventor).

Los pasos para la instalación de App Inventor son:

1. Instalar Java de Oracle desde

http://java.com/en/download/help/download_options.xml.

2. Instalar App Inventor desde

<http://appinventor.mit.edu/explore/install-app-inventor-software.html>.

El software necesario se suministra en un paquete llamado Setup App Inventor. Siga las instrucciones según su sistema operativo para hacer la instalación.

3. Iniciar App Inventor por primera vez.

Antes de empezar, asegúrese de que tiene acceso a lo siguiente:
Internet.

Una cuenta de Gmail (así es como se inicia sesión en App Inventor).

a. Iniciar el Diseñador y crear un nuevo proyecto

En el explorador Web, vaya al sitio Web de App Inventor en <http://beta.appinventor.mit.edu/>. Si esta es la primera vez que utiliza App Inventor, verá una página de proyectos en blanco.

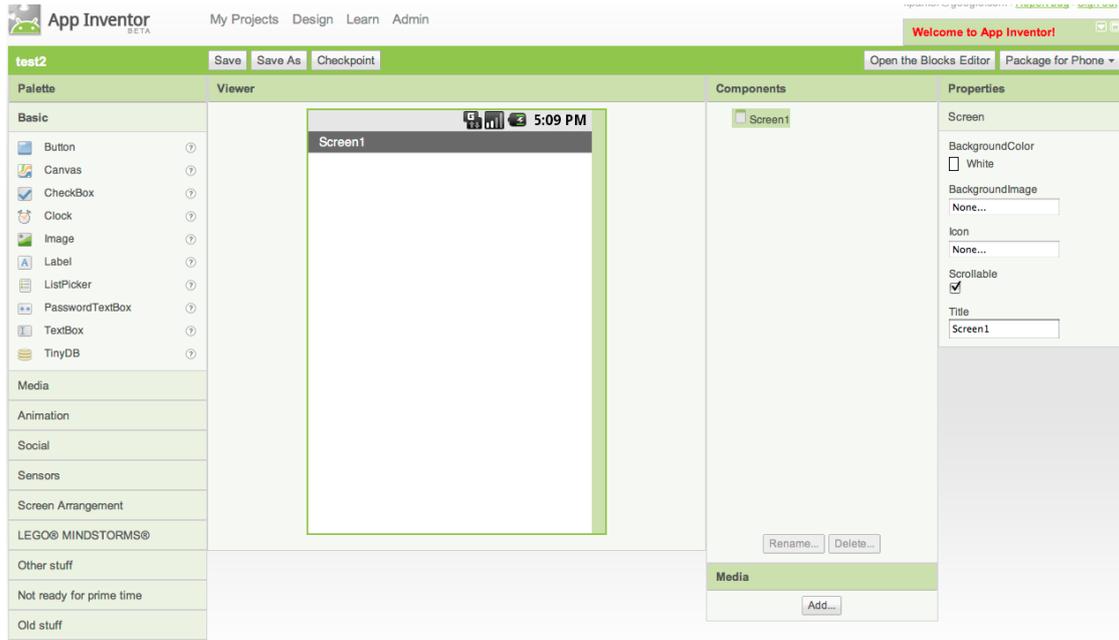
Haga clic en New en la parte izquierda, superior de la página.

Introduzca el nombre del proyecto HelloPurr (una sola palabra, sin espacios) en el cuadro de diálogo que aparece y haga clic en Aceptar.



El navegador abrirá la página Web llamada el Diseñador, el lugar donde mediante la organización, se seleccionan los componentes que se ubicarán tanto dentro como fuera de la pantalla para su aplicación y se diseña la interfaz de usuario.

La página debería tener este aspecto:



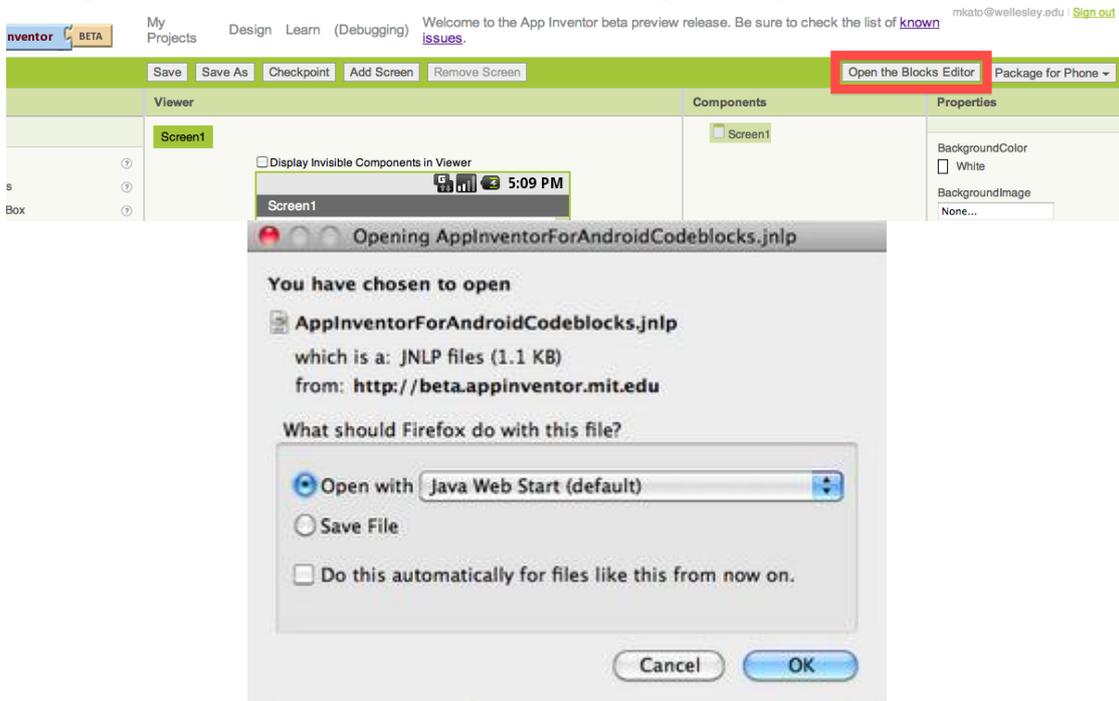
©2010 Google - [About](#) - [Privacy](#) - [Terms](#)

Build: Tue Dec 7 15:39:56 2010 (1291765196) -- 18508103

Además del Diseñador, es necesario iniciar el Editor de bloques, el lugar donde se configura el comportamiento de la aplicación. Se trata de una aplicación independiente con su propia ventana, y por lo tanto tenemos que abrir dos ventanas para diseñar una aplicación. Estas ventanas están relacionadas: los cambios realizados en el Diseñador se reflejarán inmediatamente en el Editor de bloques.

b. Iniciar el Editor de Bloques

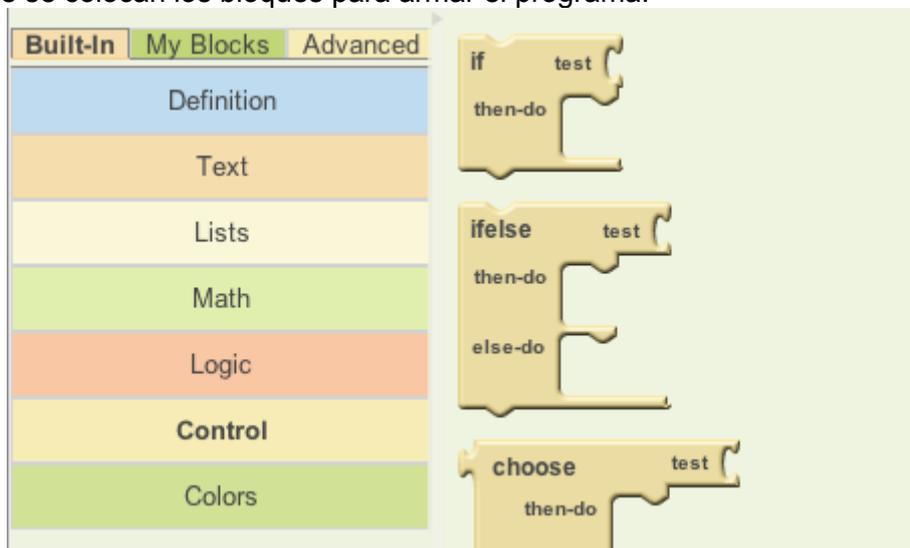
Hacer clic en el botón **Open the blocks editor** en la ventana del diseñador, el archivo de programa del Editor de Bloques debería descargarse y ejecutarse.



Este proceso puede tardar 30 segundos o más. El Editor de Bloques debería tener este aspecto:



El gran espacio vacío "canvas" en el lado derecho se conoce como la zona de trabajo, en el que se colocan los bloques para armar el programa.



En la parte izquierda, tiene tres paletas (Built-In, My Blocks, Advanced), cada paleta tiene cajones de almacenamiento de conjuntos de bloques. Al hacer clic en un cajón, se puede ver todos los elementos almacenados en el cajón.

La paleta **Built-In** contiene el conjunto estándar de los bloques que están disponibles para todas las aplicaciones a construir (por ejemplo, definición, texto, listas, etc.). Los bloques debajo de la paleta **My Blocks** contienen bloques específicos que están relacionados con el conjunto de componentes que ha seleccionado para su aplicación. La paleta **Advanced** contiene bloques para inventar aplicaciones intermedias y avanzadas con lógica más compleja.

El **Diseñador** se ejecuta desde el navegador y el **Editor de Bloques** desde Java, sin embargo, están vinculados. Por lo tanto, incluso cuando se cierra la ventana del Editor de bloques, toda la información en el Editor de bloques se almacena en el diseñador. Al hacer clic en el botón "Open the Blocks Editor", se abre un nuevo Editor de bloques, que contendrá todos los elementos que haya programado antes de cerrarlo.

A medida que construye su aplicación, se puede hacer "pruebas reales" en un emulador o en un dispositivo Android.

4.6 Desventajas

Requiere acceso a Internet.

Se requiere de un navegador Web.

Requiere cuenta en Google → Gmail.

Se accede vía Web → una vez instalado App Inventor, la URL de acceso es <http://beta.appinventor.mit.edu/>

4.6.1 Desventajas solucionadas en App Inventor 2

Se debe descargar un Jar que es el editor de bloques.

Sólo devuelve el código compilado APK, sin posibilidad de ver el código fuente.

Nota: Un archivo con extensión .APK es un paquete para el sistema operativo Android. Este formato es una variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android para smartphones y tablets.

Sólo es ejecutable en plataformas Windows.

4.7 Componentes

Los componentes que se pueden utilizar en App Inventor para crear aplicaciones se dividen en diferentes tipos:

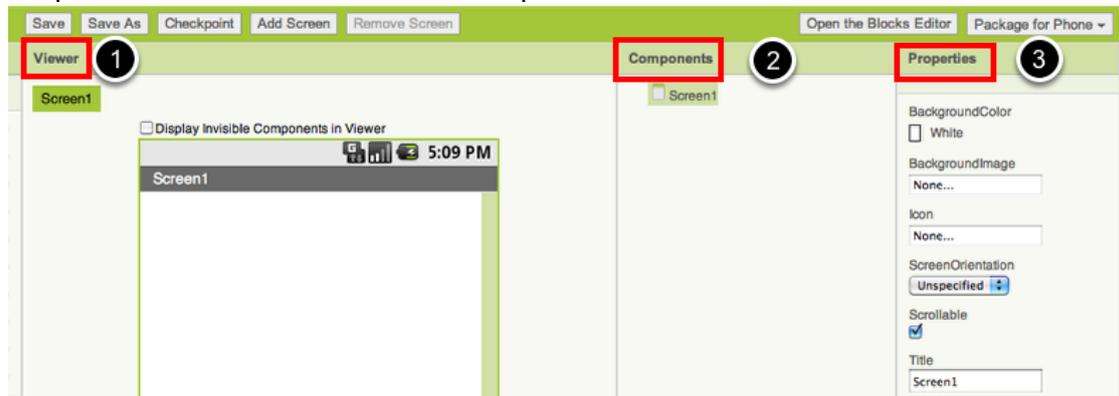
- Componentes básicos: Button, Canvas (pueden almacenar imágenes fijas o animaciones, permite dibujar en él detectando toques y arrastres), Clock, etc.
- Componentes multimedia: Camera, Sound, etc.
- Componentes de animación: Ball, ImageSprite (imagen sensible al tacto que se pueden programar para moverse en un canvas).
- Componentes sociales: ContactPicker, PhoneCall (hace llamadas telefónicas), PhoneNumberPicker (obtiene información de la lista de contactos del teléfono).
- Componentes del sensor: AccelerometerSensor (que funciona como un controlador de Wii y detecta al mover o agitar el teléfono), LocationSensor, OrientationSensor.
- Componentes para la disposición en pantalla: HorizontalArrangement y VerticalArrangement se utilizan para crear diseños verticales u horizontales simples. Usted puede crear diseños más complejos anidando componentes y crear un arreglo en pantalla o usar el componente TableArrangement.
- Componentes LEGO® MINDSTORMS® (son marcas registradas del Grupo LEGO): Estos componentes proporcionan un control de robots LEGO® MINDSTORMS® NXT utilizando Bluetooth. NxtDirectCommands, NxtSoundSensor, NxtDrive.
- Otros componentes: Notifier, BarcodeScanner, Web.
- Componentes que no están listos para la audiencia: Estos componentes son experimentales, y la documentación no está presente. GameClient, SoundRecorder, WebViewer.

Cada uno de estos componentes puede tener métodos, eventos y propiedades. La mayoría de las propiedades se pueden modificar en las aplicaciones, estas propiedades tienen bloques que puede utilizar para obtener y establecer los valores. Algunas propiedades no pueden ser modificados por aplicaciones, estos componentes solo tienen bloques que se pueden usar para obtener los valores, no para setearlos. Algunas propiedades sólo están disponibles en el Diseñador.

4.8 Seleccionar los componentes para diseñar la aplicación

Para utilizar un componente en la aplicación, tendrá que hacer clic y arrastrarlo hacia el visor en medio del Diseñador. Cuando se agrega un componente al Visor (# 1), también aparecerá en la lista de los componentes al lado derecho del visor.

Los componentes (# 2) tienen propiedades que se pueden ajustar para cambiar la forma en que el componente aparece o se comporta dentro de la aplicación. Para ver y cambiar las propiedades de un componente (# 3), primero debe seleccionar el componente deseado en la lista de componentes.



4.9 Bloques

Los diferentes tipos de bloques que se pueden utilizar al crear sus aplicaciones de App Inventor son:

- Bloques de definición: procedure, name, variable (global o local al bloque), etc.
- Bloques de texto: text, join, length, etc.
- Bloques de lista: make a list, select list item, replace list item, etc.
- Bloques matemáticos: equals, number, greater than, etc.
- Bloques lógicos: false, not, =, etc.
- Bloques de control: if, for each, while, etc.
- Bloques de color: Se pueden conectar bloques de color en las tomas de corriente de los bloques que requieren colores, como en un bloque de texto o el fondo de un componente. Existen bloques de colores predefinidos, pero también se pueden definir y crear nuestros propios colores. Los colores se crean utilizando cuatro números, cada uno que va desde 0 a 255. Los tres primeros números significan la cantidad de rojo, verde, y azul en el color, correspondientes a *R*, *G*, *B*. El cuarto número especifica *la opacidad* que entra en juego cuando las regiones se solapan. Opacidad 255 es totalmente opaco, mientras que, opacidad 0 es completamente transparente. El componente de opacidad es opcional, si se omite, se establece en 255.

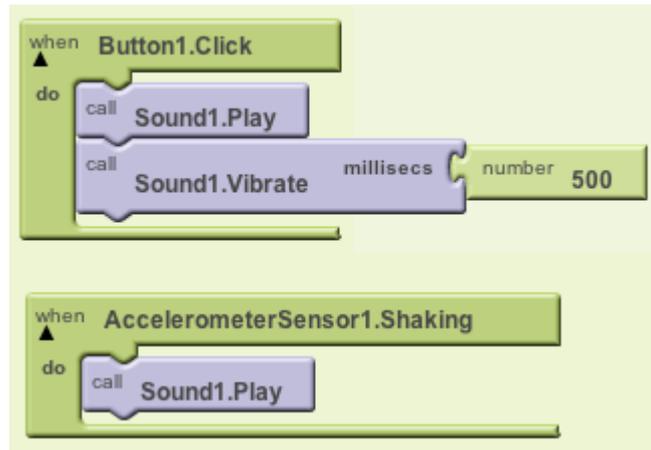
El equipo de la aplicación Inventor ha creado bloques para casi todo lo que puedes hacer con un teléfono Android, así como bloques para programar, para almacenar información, para la repetición de las acciones, y para realizar acciones bajo ciertas condiciones. Hay incluso bloques para hablar con servicios como Twitter.

4.10 Comprensión de bloques y cómo funcionan

4.10.1 Bloques controladores de eventos

Programas de App Inventor describen cómo el teléfono debe responder a ciertos acontecimientos: se ha pulsado un botón, el teléfono se sacudió, el usuario está arrastrando su dedo sobre un lienzo, etc. Esto se especifica mediante bloques de **control de eventos**, que utilizan la palabra *when*.

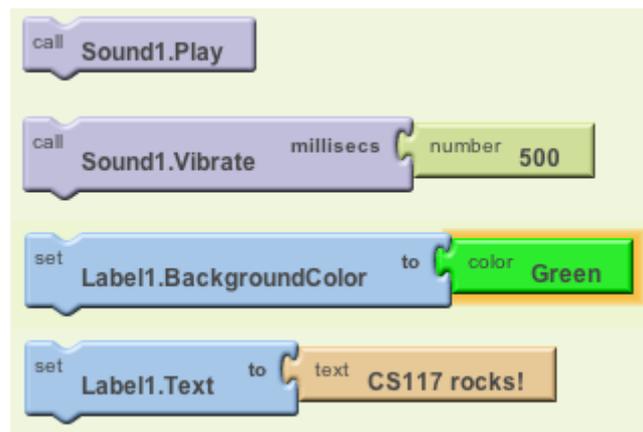
Ejemplos de controladores de eventos:



4.10.2 Comandos y expresiones

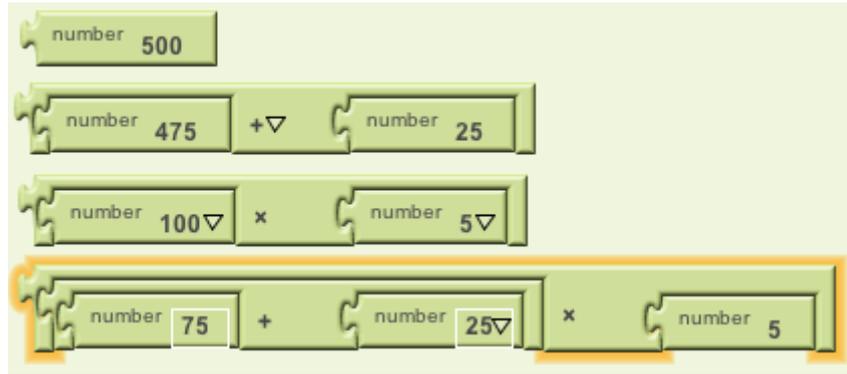
Cuando un controlador de eventos se dispara, se ejecuta una secuencia de comandos en su cuerpo. Un comando es un bloque que especifica una acción a realizar en el teléfono (por ejemplo, la reproducción de sonidos). La mayoría de los bloques de comandos son de color púrpura o azul.

Ejemplos de algunos comandos:

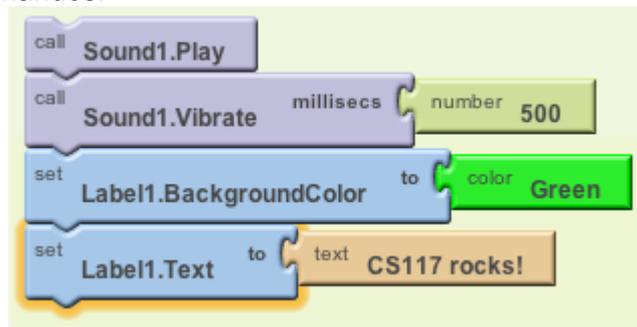


Algunos comandos requieren uno o más valores de entrada (también conocido como parámetros o argumentos) para especificar completamente su acción. Estos valores de entrada se muestran en tomas de corriente en el borde derecho del comando.

Estas tomas pueden ser llenados con expresiones que son bloques que denotan un valor. Los bloques de expresión tienen tapones que apuntan hacia la izquierda, por ejemplo, todas las siguientes expresiones denotan el número 500:



Los comandos se componen verticalmente en una pila de comandos. Aquí está una pila con cuatro comandos:

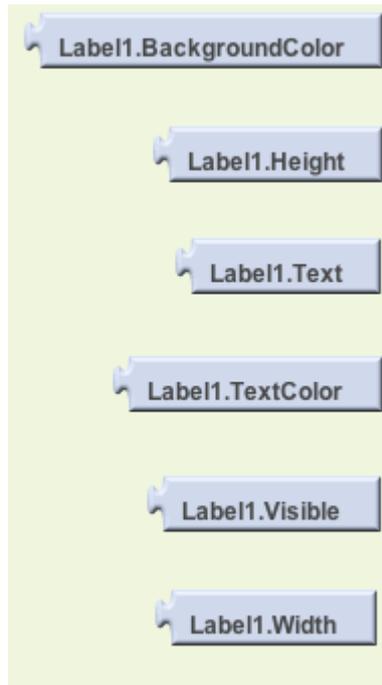
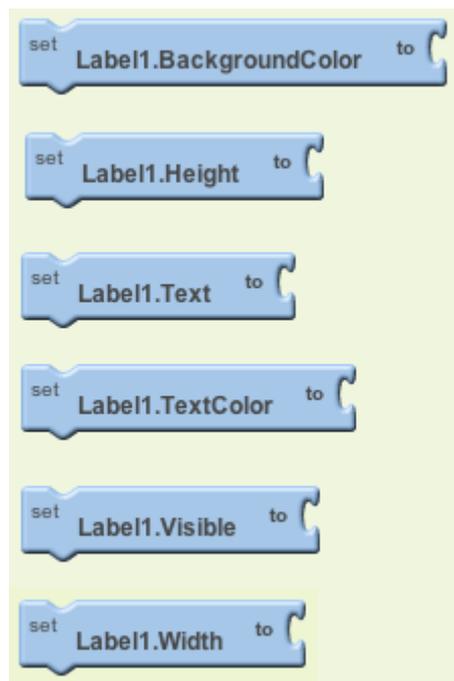


Cuando se coloca la pila de comandos en el cuerpo de un controlador de eventos (por ejemplo, el controlador de eventos `when.Button1.Click`), el comando se ejecutará desde la parte superior hasta la inferior.

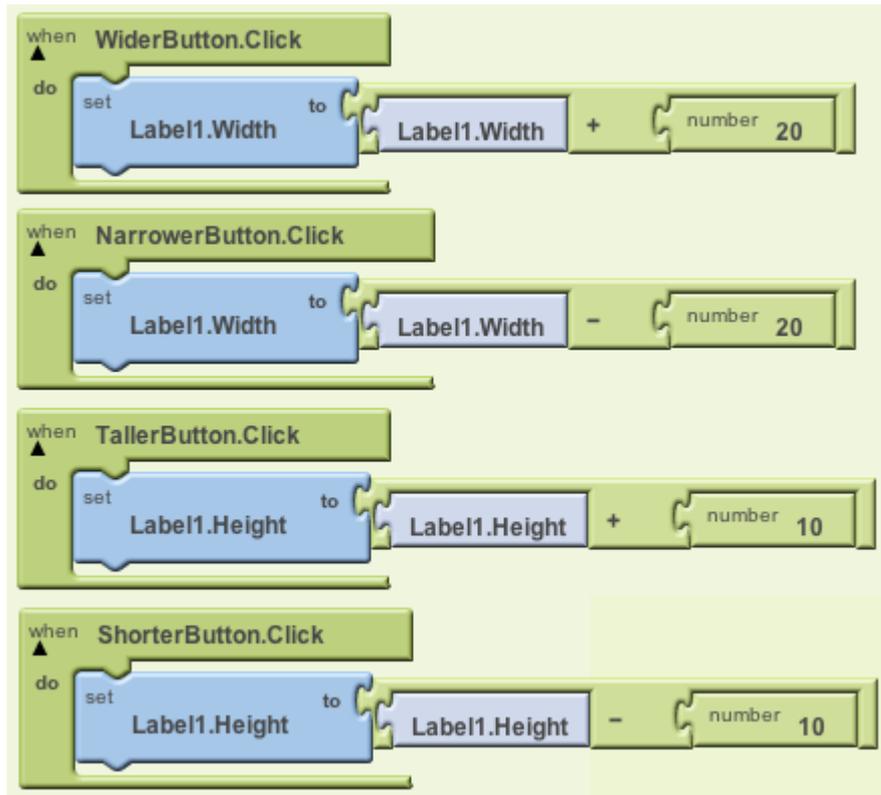
4.11 Manipular el estado de los componentes

Los valores actuales de las propiedades describen el estado del componente. Los programas de App Inventor pueden obtener y establecer la mayoría de las propiedades de los componentes a través de bloques. Por ejemplo, aquí están los bloques para manipular el estado de `Label1`.

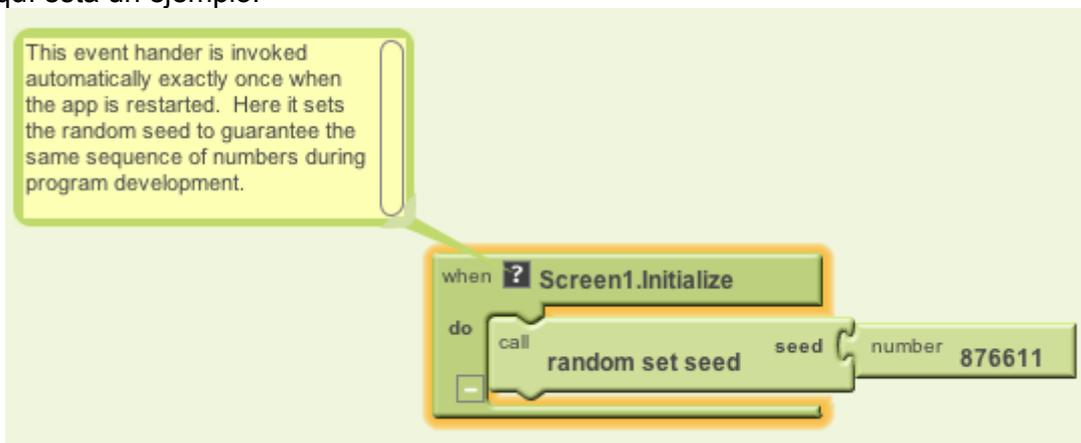
Bloques getter

**Bloques setter**

Los Bloques Getter son expresiones que 'obtienen' o que contienen el valor actual de la propiedad. Los Bloques Setter son comandos que cambian el valor asociado a la propiedad.



En el Editor de Bloques, se puede dejar un comentario a cualquier bloque de código, aquí está un ejemplo:



4.12 Novedades de App Inventor 2

App Inventor 2 es diferente de la App Inventor 1 en varios temas:

- Dropdowns
- Mutadores
- Variables globales y locales
- El emulador
- Colores

4.12.1 Dropdowns (menú desplegable)

Algunos bloques de App Inventor se pueden cambiar para hacer otra cosa con sólo hacer clic en el nombre del bloque.

Algunos bloques de App Inventor tienen una pequeña flecha hacia abajo a la derecha del nombre del bloque. Esta flecha indica que el bloque es un desplegable. Al hacer clic en el nombre del bloque, un menú desplegable aparecerá con opciones de diferentes bloques a los que el bloque puede cambiar.

4.12.2 Mutadores



Cualquier bloque que tiene un cuadro azul con un signo más blanco en la parte superior que coincide con la imagen se considera un bloque mutador.

Los Mutators cambian de forma. Al hacer clic en el icono azul, el usuario puede arrastrar bloques más pequeños adicionales en el bloque más grande mutador cambiando así la forma y la funcionalidad del bloque mutador.

Explicación del bloque mutador min: el usuario quiere encontrar el número mínimo de una lista de 3 valores. Actualmente sólo hay espacio para 2 para conectar con el bloque min. Entonces agrega un pequeño bloque adicional al bloque más grande mutador.

4.12.3 Variables globales y locales

Una **variable global** es una variable que se puede acceder en múltiples ámbitos: obtener su valor actual o establecer su valor en algo más. Las variables globales se crean mediante el bloque “initialize global name to” encontrado en el cajón Variables.

Una **variable local** es una variable que se declara dentro de una función o es un argumento pasado a la función. Esto significa que sólo se puede acceder a estas variables en esa función específica en la que se declaran o se pasan como argumento. Se crean variables locales cuando:

- los argumentos se pasan a un procedimiento o evento.
- utilizando el bloque “initialize local name to”.
- utilizando un bloque “for each in list” o “for each from to” (estos bucles crearán una variable local para la letra *i*).

4.12.4 AI2 colores

Hay tres tipos principales de bloques de color:

- a color box
- make color
- split color

- Bloques de colores básicos



Este es un bloque de color básico. Si hace clic en el color en el centro, una ventana emergente aparece en la pantalla con una tabla de 70 colores que usted puede elegir.

- Make color



“make color” toma en una lista de 3 o 4 números. Estos números representan los valores en un código de RGB. El cuarto valor es opcional y representa el valor alfa o la saturación del color. El valor alfa por defecto es 100.

- Split color

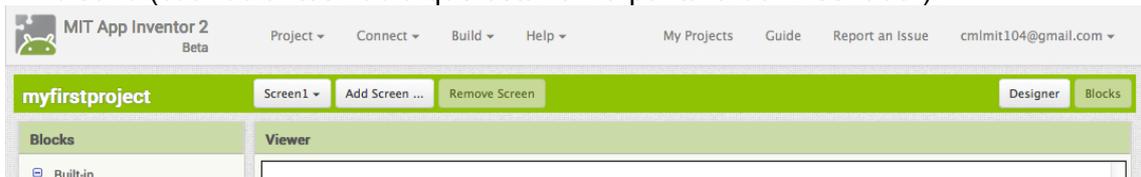
“splits color” hace lo contrario que “make color”. Se necesita de un color: un bloque de color, variable que contiene un color, o la propiedad de uno de



los componentes que representan un color y devuelve una lista de los valores RGB en el código RGB de ese color.

4.12.5 Otras características modificadas

- App Inventor es ahora completamente en el navegador. Previamente al ejecutar App Inventor necesitabas instalar y ejecutar un archivo de Java llamado el Editor de bloques. Ahora, el Editor de bloques es sólo un modo diferente en su proyecto que se ve desde el navegador.
- Un archivo de código fuente ahora tiene la extensión ".aia" en lugar de un archivo ".zip".
- Puede añadir pantallas adicionales, tanto en los bloques como en el modo de diseño (cuando antes había que estar en la pantalla del Diseñador).



- Observe cómo el usuario se encuentra en modo de bloques ya que se deshabilita el botón "Blocks". Sin embargo, el botón "Add Screen..." sigue estando disponible.
- Todos los bloques están disponibles en el lado izquierdo. Ya no hay necesidad de cambiar entre Built-In, Mis bloques y cajones avanzadas.
- Los procedimientos (anteriormente en el cajón Definiciones) también son diferentes. Ahora todos los procedimientos que se escriben se pueden encontrar aquí.
- Hay dos nuevos bloques en el cajón de control. En primer lugar es un bloque llamado "do then return" que se encuentra en el cajón de control. En segundo lugar, está el bloque "evaluate but ignore result", que sustituye el bloque "evaluate".
- Ya no hay un bloque "make text". En su lugar, puede simplemente usar "join".

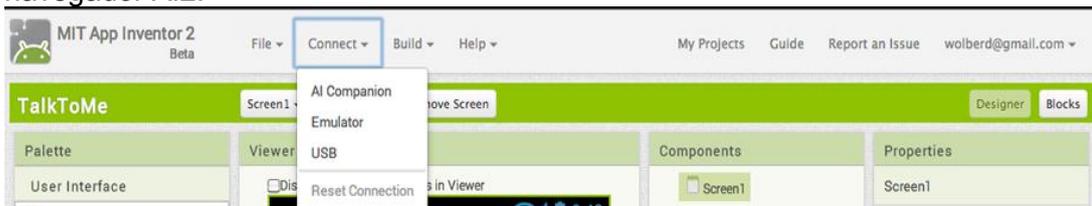
4.13 Configuración de App Inventor 2

En esta nueva versión de la herramienta el Diseñador y Editor de bloques se ejecutan por completo en el navegador. El software de la aplicación App Inventor 2, es accesible desde ai2.appinventor.mit.edu.

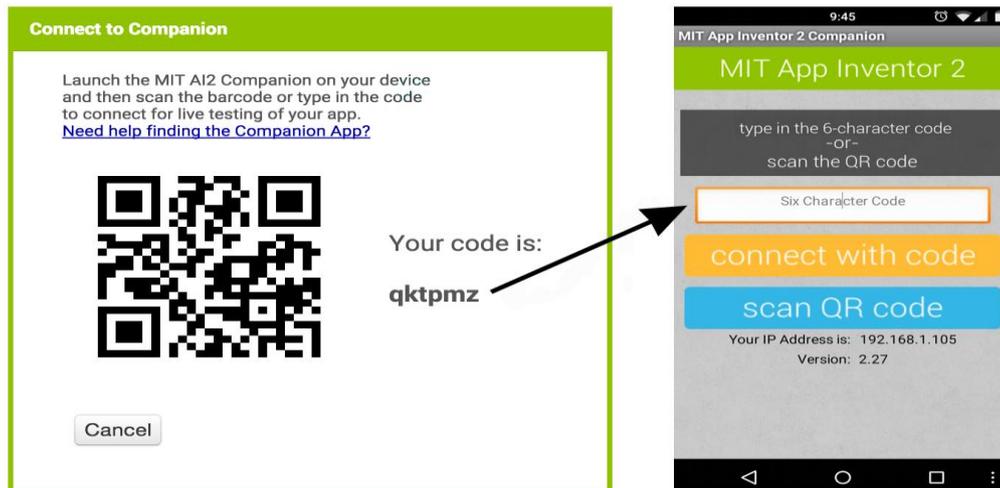
Para ver su aplicación en un dispositivo mientras lo construyes (también llamada "Pruebas en vivo"), existen tres opciones:

- **Si está utilizando un dispositivo Android y tiene una conexión inalámbrica a Internet**, puede comenzar la creación de aplicaciones sin necesidad de descargar ningún software en su ordenador, solo siguiendo los siguientes pasos:
 1. Descargar e instalar el MIT AI2 Companion en su teléfono.
 2. Conectar la computadora y el dispositivo a la misma red Wi-Fi.
 3. Abrir un proyecto de App Inventor y conectarlo a su dispositivo.

A continuación, seleccione "Connect" y "AI Companion" en el menú superior del navegador AI2:

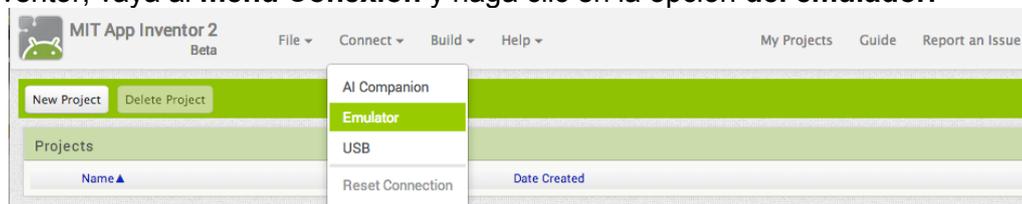


Un diálogo con un código QR aparecerá en la pantalla del PC. En el dispositivo, inicie la aplicación MIT AI2 Companion. A continuación, haga clic en el botón "Scan QR code" y escanea el código en la ventana de App Inventor para acceder:



Al cabo de unos segundos, debería ver la aplicación que se está construyendo en el dispositivo.

- **Si no tiene un dispositivo Android**, podrá instalar y ejecutar el emulador en AI2 siguiendo los pasos a continuación:
 1. Instalar el software App Inventor Setup en su ordenador para que pueda utilizar el emulador de Android en la pantalla.
 2. Iniciar el programa aiStarter (Windows y Linux solamente). Este programa es el auxiliar que permite que el navegador se comunice con el emulador. El programa aiStarter se instaló al instalar el paquete de instalación de App Inventor. El aiStarter se iniciará automáticamente en un Mac, y ejecuta de forma invisible en segundo plano.
 3. Abrir el proyecto App Inventor y conectar el emulador, desde el menú de App Inventor, vaya al **menú Conexión** y haga clic en la opción **del emulador**.



Deberá esperar hasta que el teléfono emulado ha terminado de preparar su tarjeta SD. Cuando conecte, el emulador se iniciará y mostrará la aplicación que tenga abierta en App Inventor 2.

- **Si no tiene una conexión inalámbrica a Internet**, podrá conectar el dispositivo con un cable USB siguiendo estos pasos:
 1. Instalar el software App Inventor Setup en su ordenador.
 2. Descargar e instalar el MIT AI2 Companion en su teléfono.
 3. Iniciar el programa aiStarter (Windows y Linux solamente).
 4. Configurar el dispositivo para USB (Activar depuración USB ON).
 En su dispositivo Android, vaya a Configuración, Opciones de desarrollo y asegúrese de que está permitida la "Depuración USB".
 En la mayoría de los dispositivos con Android 3.2 o mayor, puede encontrar esta opción en Ajustes> Aplicaciones> Desarrollo. En Android 4.0 y más reciente, se encuentra en Ajustes> Opciones del desarrollador.
 Nota: En Android 4.2 y versiones posteriores, Opciones del desarrollador se oculta de forma predeterminada. Para que esté disponible, vaya a Ajustes> Acerca del teléfono y toque el número de compilación siete veces. Volver a la pantalla anterior para encontrar opciones del desarrollador, incluido "Depuración USB".

5. Conecte su dispositivo Android a la computadora usando el cable USB y autentifíquese si es necesario.

4.14 Diseñador y editor de bloques

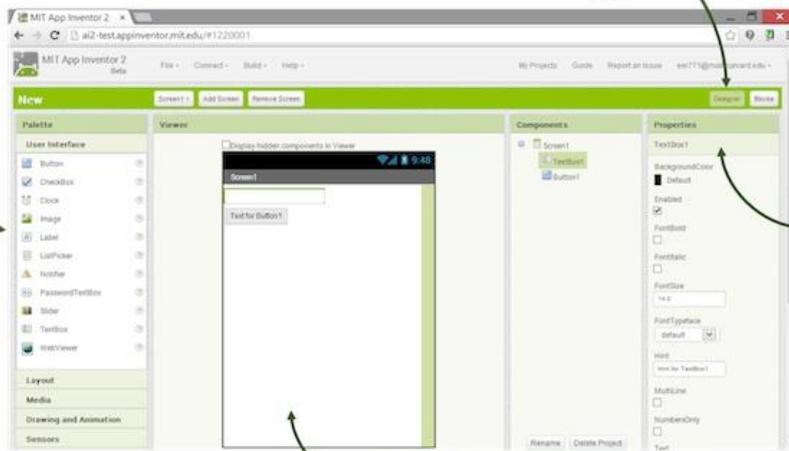
App Inventor consiste en el **diseñador** y el **editor de bloques**. Éstos se describen en detalle a continuación.

4.14.1 Diseñador de App Inventor 2

Diseño de la interfaz de usuario de la aplicación mediante la organización, tanto dentro como componentes fuera de la pantalla.

Palette: Find your components and drag them to the Viewer to add them to your app.

Designer Button: Click from any tab to go to the Designer tab.



Properties: Select a Component in the Components List to change its properties (color, size, behavior) here.

Viewer: Drag components from the Palette to the Viewer to see what your app will look like.

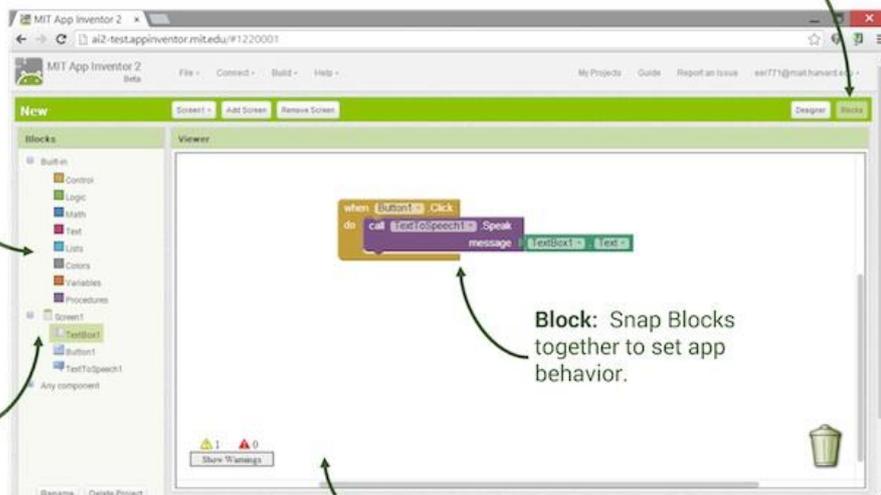
4.14.2 Editor de bloques de App Inventor 2

Programar el comportamiento de la aplicación, poniendo bloques juntos.

Built-In Drawers: Find Blocks for general behaviors you may want to add to your app and drag them to the Blocks Viewer.

Blocks Button: Click from any tab to go to the Blocks tab.

Component-Specific Drawers: Find Blocks for behaviors for specific Components and drag them to the Blocks Viewer.



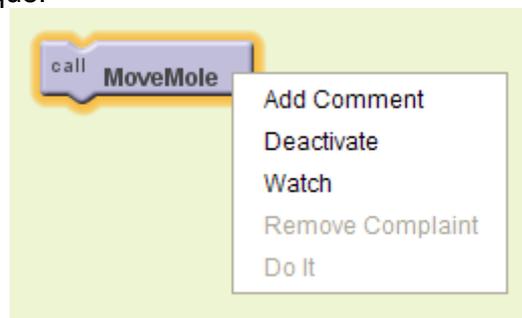
Block: Snap Blocks together to set app behavior.

Viewer: Drag Blocks from the Drawers to the Blocks Viewer to build relationships and behavior.

4.15 Desarrollo incremental con el editor de bloques

Al crear aplicaciones con App Inventor, se está trabajando directamente en el teléfono: arrastrar un botón en la vista del Diseñador, asignarle un color y aparecerá en el teléfono; arrastrar un bloque que hace que algo suceda cuando se pulsa el botón y luego pulsarlo en el teléfono para ver que sucede. Este tipo de información es de gran utilidad, ya que le permite desarrollar y probar sus aplicaciones de forma incremental, definiendo cada nuevo comportamiento y probarlo a medida que avanza. Programadores inexpertos suelen cometer el error de construir una gran cantidad de cosas antes de probar cualquiera de ellas. Entonces, cuando encuentran errores, están enfrentados con una enorme maraña para resolver, en los que no saben qué piezas están funcionando y cuáles no. El desarrollo incremental permite aislar errores más rápidamente y corregirlos más fácilmente.

Además, App Inventor incluye características para ayudar en las pruebas y la depuración mientras esté usando el editor de bloques. Se puede ver un menú si hace clic derecho en un bloque.

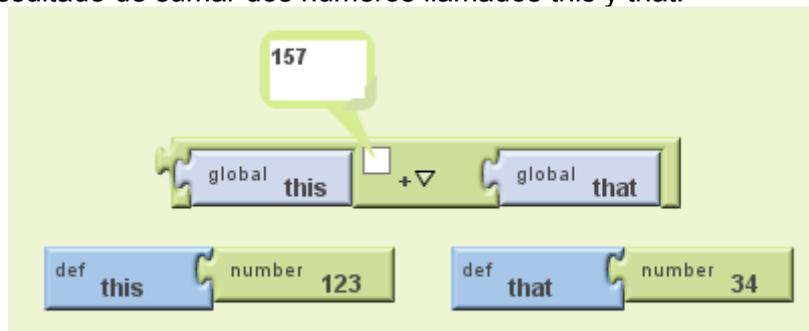


No todos los bloques ofrecen la misma opción en el menú.

4.15.1 Do it

En la depuración de un programa como el MoleMash, ver si el Mole se mueve en el teléfono al seleccionar Do It.

Do It no sólo hace la acción del bloque, sino que también pone un globo que muestra el valor devuelto. La siguiente figura muestra el uso de Do It en un bloque, además de mostrar el resultado de sumar dos números llamados this y that.



4.15.2 Observación de las variables

A veces, sobre todo en la depuración, al hacer clic en Do It lo que realmente se quiere ver son los cambios en el valor de una variable a medida que se ejecuta el programa. Al hacer clic en "watch" se abre un globo cuyo resultado es un seguimiento constante del valor de la variable.

4.16 Advertencias y errores

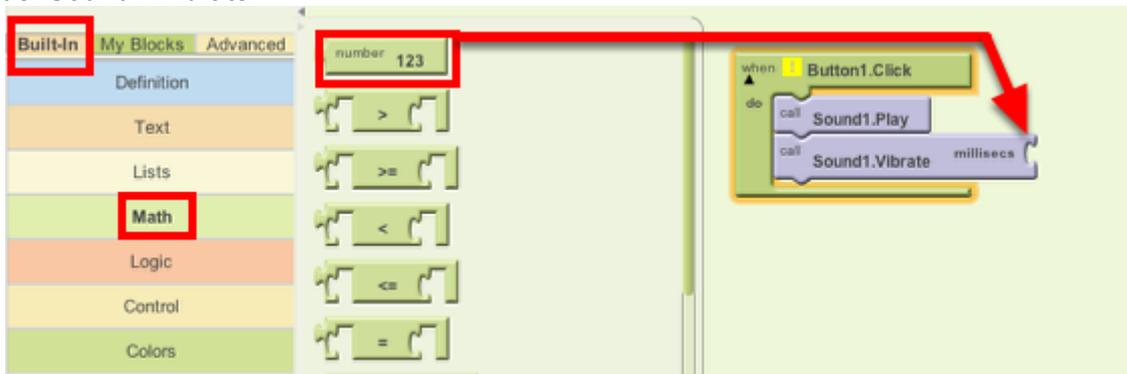
Ahora agregaremos vibración al ronroneo del gato. Ir al editor de bloques y abrir el cajón **Sound1** y arrastre el bloque Sound1.Vibrate y colocarlo bajo el bloque

Sound1.Play. Usted verá un icono de advertencia de color amarillo en la esquina izquierda del bloque, lo que significa que el bloque tiene un componente que falta.

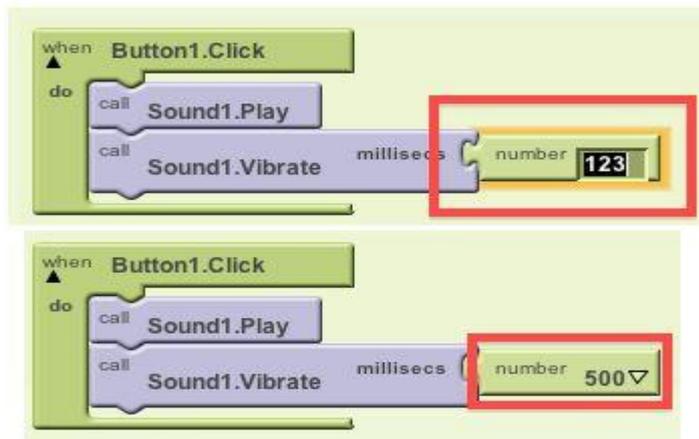


El bloque Sound1.Vibrate tiene una ranura abierta, lo que significa que hay que enchufar algo en él para especificar más sobre cómo debería funcionar el comportamiento. Aquí, queremos especificar la duración de la vibración.

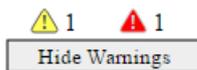
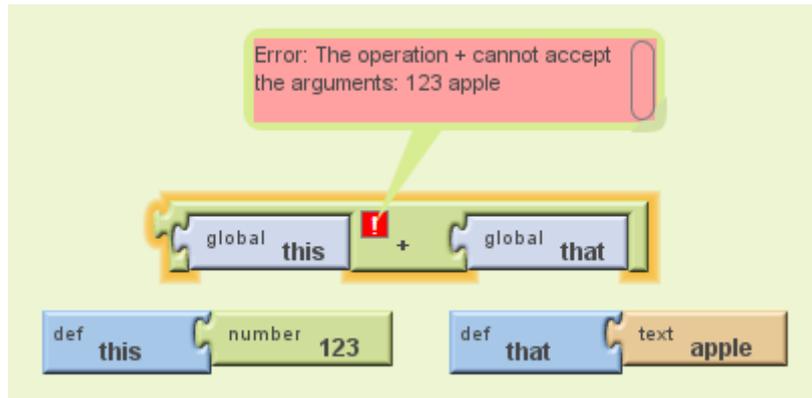
Ir a la paleta, al cajón **de Math**, arrastre el bloque de números y colocarlo en el socket del Sound1.Vibrate.



Después de colocar el bloque de números, haga clic en el número "123". Resalta el número en negro: escriba "500" con el teclado.



A veces el resultado de un bloque será un error. Por ejemplo en la imagen de abajo, donde hemos cambiado el valor del texto that a "apple" causando un pequeño signo de exclamación.



En la esquina del editor se puede observar un contador.

4.17 Compartir y empaquetar aplicaciones

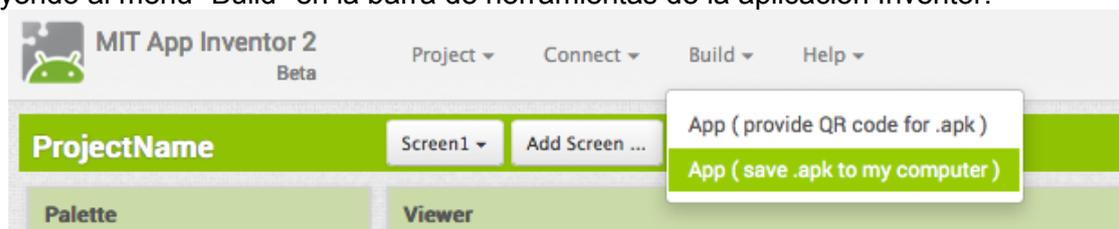
Usted puede compartir su aplicación en un formato *ejecutable* (.apk) que se puede instalar en un dispositivo, o en forma de código *fuentes* (.aia) que se pueden cargar en App Inventor y editarlo.

4.17.1 Compartir su aplicación para que otros puedan editarlo (.aia)

Seleccione **Archivo | Exportar proyecto** para exportar el código fuente (bloques) para su proyecto. El código fuente se descarga en un archivo .aia. Si lo envía a un amigo, pueden abrirlo con **Archivo | Importar proyecto**.

4.17.2 Empaquetar su aplicación para su uso por otros (.apk)

Para tener una aplicación en ejecución sin estar conectado a App Inventor, debe **"empaquetar"** la aplicación para producir un paquete de aplicación (archivo .apk) yendo al menú "Build" en la barra de herramientas de la aplicación Inventor.



Una vez que tenemos la aplicación (archivo ".apk") se puede enviar por mail para instalarlo abriendo el correo electrónico desde su teléfono. Si desea distribuir de forma más amplia, puede subirlo a un sitio web donde se pueda acceder o también puede distribuir su aplicación en la tienda de Google Play.

La otra opción (proporcionar el código QR para .apk) produce un código QR susceptible de ser analizada que descargar la aplicación durante dos horas. Puede compartir el código con los demás, pero tienen que usarla dentro de 2 horas de su generación.

NOTA: Cualquier instalación de la aplicación (que es un archivo ".apk") tendrá que cambiar la configuración de su teléfono para permitir la instalación de aplicaciones que no son de mercado: para encontrar esta opción en las versiones de Android anteriores a 4.0, vaya a "Configuración> Aplicaciones" y luego seleccione la casilla junto a "Orígenes desconocidos". Para los dispositivos con Android 4.0 o superior, vaya a "Configuración> Seguridad" o "Configuración> Seguridad y bloqueo de pantalla" y luego marque la casilla junto a "Orígenes desconocidos" y confirme su elección.

4.18 Consejos y trucos de App Inventor 2

- **Hacer comentarios**
Se pueden añadir comentarios haciendo clic derecho sobre un bloque y seleccionando la opción Añadir un comentario. Tu comentario se guardará y se puede acceder haciendo clic en el signo de interrogación en la parte superior derecha del bloque.
- **Condensando bloques**
Haga clic en un grupo de bloques para condensarlos y le permitirá tener más espacio disponible en la pantalla. La condensación se puede hacer haciendo clic derecho y seleccionando la opción "Collapse Blocks".
- **Typeblocking**
Haga clic en la pantalla y comience a escribir su nombre. Esto debería provocar una lista de bloques con los títulos que coinciden con la palabra escrita. Para más ayuda sobre este truco, visite la página <http://appinventor.mit.edu/explore/tips/typeblocking.html>.
- **Eliminación rápida**
En lugar de arrastrar el bloque o bloques a la papelera en la esquina inferior izquierda, seleccionarlos haciendo clic con el ratón y pulsa la tecla de borrar.
- **Copiar y pegar**
Si hay ciertos bloques que va a necesitar de nuevo, sólo tienes que copiar y pegar los bloques.

5 IBM Rational Rhapsody

Los ingenieros de sistemas y desarrolladores de software utilizan IBM Rational Rhapsody Developer en todas las fases de análisis y diseño de sistemas, cuya línea de productos es un diseño colaborativo en un entorno de desarrollo visual, para crear, probar y documentar el desarrollo de software dirigido por modelos en sistemas técnicos, integrados y/o de tiempo real basado en UML y SysML.

Ayuda a mejorar su productividad durante todo el ciclo de vida de desarrollo del software incluido: desde la captura de requisitos a la implementación, prueba y desarrollo. Se puede reducir el tiempo de comercialización de nuevos productos con procesos de desarrollo de software automatizados, una pronta validación y pruebas de modelo mejoradas.

El poder de modelado gráfico transforma el proceso de generación de código, de modo que sea capaz de describir el comportamiento del sistema y validar la consistencia mediante el uso de la simulación y testeo de pruebas automatizadas para identificar problemas en el desarrollo temprano.

Rational Rhapsody implementa la solución de diagramas de diseño y generación automática de código compatible con ANSI que está optimizado para los entornos más utilizados.

5.1 Historia

Rhapsody fue lanzado por primera vez en 1996 en la compañía de software I-Logix Inc. de Israel. Rhapsody fue desarrollado como una herramienta orientada a objetos para el modelado y ejecución de Statecharts (diagramas de estado), basado en el trabajo realizado por David Harel en el Instituto de Ciencia Weizmann en Rehovot, Israel, que era el primero en desarrollar el concepto de Statecharts.

En 2006, I-Logix vendió la compañía a la compañía sueca de software Telelogic AB. Rhapsody se convirtió en un producto Rational Software después de la adquisición de Telelogic AB en 2008, al igual que todos los anteriores productos de Telelogic. Desde el cambio de marca, Rational Rhapsody se ha integrado con los sistemas de IBM Rational.

5.2 Características

Dependiendo de la edición que se utiliza, Rational Rhapsody tiene las siguientes características:

- El apoyo de todas las ediciones tiene la capacidad de analizar el comportamiento previsto mediante el modelado y generación de código a partir de diagramas UML, SysML o lenguajes específicos del dominio DSL y probar la aplicación mientras la crea.
- Genera código de aplicación para los lenguajes C, C#, C++, Java y Ada, incluyendo vistas de arquitectura y de comportamiento (gráficos de estados, diagramas de actividad).
- El software tiene una licencia shareware, la cual permite evaluar el producto por 30 días.
- Sincroniza los cambios de diseño o de código para mantener ambos sincronizados.
- Comprobación o chequeo estático para asegurarse de que el diseño es consistente ayudando a mejorar la coherencia e integridad de los modelos.
- Integración con la plataforma Eclipse (sólo Windows) en un entorno integrado de

modelado, código y depuración. El plugin Rational Rhapsody para Eclipse integra un modelo en el entorno Eclipse, permitiendo a los desarrolladores de software trabajar en el código o modelo en un solo entorno de desarrollo. Se pueden utilizar las capacidades del modelado Rational Rhapsody o modificar el código usando el editor de Eclipse, mientras que mantiene la sincronización entre ambos y navegar fácilmente de una a otra.

- Creación de workspaces personalizados: cuando se abre un proyecto en Rational Rhapsody, puede abrirlo sin cargar las unidades del proyecto. De esta manera, se puede crear un workspace personalizado que tiene sólo las unidades que desea, lo que reducirá el tiempo necesario para las operaciones de rutina, tales como el guardado y la generación de código.
- La función de librerías de modelos permite cargar sólo las clases que están referenciadas en el modelo. El paquete que contiene la clase de referencia es cargado, mientras que las librerías de clases restantes solo se muestran y tienen el símbolo U (unloaded) al lado de ellos.
- Generación de código de forma incremental y en paralelo. Después de la generación de código inicial para una configuración, Rational Rhapsody genera código solamente para los elementos que se han modificado desde la última generación. Aunque siempre está la opción de regenerar todo el código desde cero. Igualmente, a partir de la versión 8.0, la generación de código por defecto es usar el procesamiento en paralelo.
- Se puede configurar Rational Rhapsody con propiedades. Las propiedades del proyecto son pares nombre-valor que se puede utilizar para personalizar muchos aspectos de la interacción del ambiente y la generación de código.
- Las entradas al generador de código son el modelo y las propiedades de generación de código. Las salidas del generador de código son archivos de código fuente en el lenguaje de destino.
- Roundtripping es un método utilizado para actualizar el modelo rápidamente con pequeños cambios introducidos al código generado previamente. Puede activar un roundtripping mediante la actualización del código, para luego sincronizar de forma explícita el modelo. Sin embargo, no utilice roundtripping para grandes cambios que requerirían la reconstrucción del modelo.
- Generación automática de código debido a la asociatividad dinámica de código y modelo (DMCA), donde los cambios realizados en el modelo se reflejan en el código y los cambios realizados al código puede ser fácilmente roundtripping de nuevo en el modelo.
- Diseño de aplicaciones Android: Rational Rhapsody incluye un perfil para el desarrollo de aplicaciones para dispositivos basados en Android, utilizando el Android plugin para Eclipse (Herramientas de Desarrollo de Android - ADT).
- Rational Rhapsody permite visualizar el modelo a través de la simulación. Una vez que se simula el modelo, puede abrir diagramas simulados que le permiten observar el modelo y llevar a cabo la depuración a nivel de diseño. El soporte de animación le permite controlar el diseño, pero a un nivel superior de abstracción.
- Rational Rhapsody Developer se integra con otros productos IBM Rational, para un desarrollo completo de todo el ciclo de vida de las aplicaciones.
- Ingeniería inversa: puede analizar el código existente y construir un modelo Rational Rhapsody basado en el código.
- API Rational Rhapsody: proporciona una API que se puede utilizar para realizar acciones Rational Rhapsody desde dentro de un script. Se proporcionan dos versiones de la API: una API basada en COM que se puede utilizar con C++ o VB/VBA/VBScript, y una API Java que se puede utilizar para realizar acciones Rational Rhapsody desde un programa Java.

- Interfaz de línea de comandos de Rational Rhapsody: Una versión de línea de comandos se proporciona para realizar fácilmente acciones que no requieren la interfaz gráfica de usuario, por ejemplo, la generación de código.

Rational Rhapsody incluye características específicas de Java:

- Anotaciones Java
- Enumeraciones Java
- Importación estática
- Bloques estáticos
- Javadoc
- Modelo de referencia de Java

Algunas de las características no disponibles en la versión Linux de Rational Rhapsody versión 8:

- Desarrollo de aplicaciones Android.
- Generación de código personalizable.
- Integración de la plataforma Eclipse.
- Aplicaciones Rational Rhapsody no se pueden desarrollar en Linux con las bibliotecas SWT. (Tenga en cuenta que puede desarrollar aplicaciones Rational Rhapsody con Eclipse en Windows y utilizarlos tanto para Linux como para Windows.).
- Aplicación de 64 bits de Rational Rhapsody.

5.3 Instalar el SDK de Android y vincularlo con Eclipse

Nota antes de empezar: Existe un paquete que incluye Eclipse, con el ADT y el SDK ya todo perfectamente montado y funcionando. Puedes descargar el paquete “Eclipse + plugin ADT + SDK”. Es tan fácil como: descomprimir el contenido en una carpeta, e ir a la carpeta “IDE”, luego a la carpeta “eclipse” y ejecutar el archivo “eclipse.exe”. Con esto te ahorras esta sección.

Si se quiere aprender, completa este tutorial, donde se reflejará un paso a paso de la instalación y configuración para el desarrollo de Android.

Una nota breve más: Existe otro entorno de trabajo más nuevo llamado Android Studio. Se puede ver como se usa e instala en este vídeo <http://jarroba.com/introduccion-a-android-studio-video/>.

En apariencia es diferente a Eclipse, pero prácticamente es lo mismo y más moderno. Para los que se inicien en Android se recomienda Android Studio antes que Eclipse, ya que Eclipse ya no tiene soporte por parte de Google. Pero Android Studio no se integra con Rational Rhapsody.

5.3.1 Instalando el JDK

El Java Development Kit o (JDK), es un software que provee herramientas de desarrollo para la creación de programas en Java.

Los programas más importantes que se incluyen son:

- **appletviewer.exe:** es un visor de applets para generar sus vistas previas, ya que un applet carece de método main y no se puede ejecutar con el programa java.
- **javac.exe:** es el compilador de Java.
- **java.exe:** es el intérprete de Java.
- **javadoc.exe:** genera la documentación de las clases Java de un programa.

Se puede descargar desde:

<http://www.oracle.com/technetwork/java/javaseproducts/downloads/index.html>

En esta página podrán ver todas las versiones disponibles del JDK. Si su sistema operativo es Windows, deberá elegir, descargar e instalar una de las versiones disponibles (32 o 64 bits).

Las versiones de Windows son las siguientes:

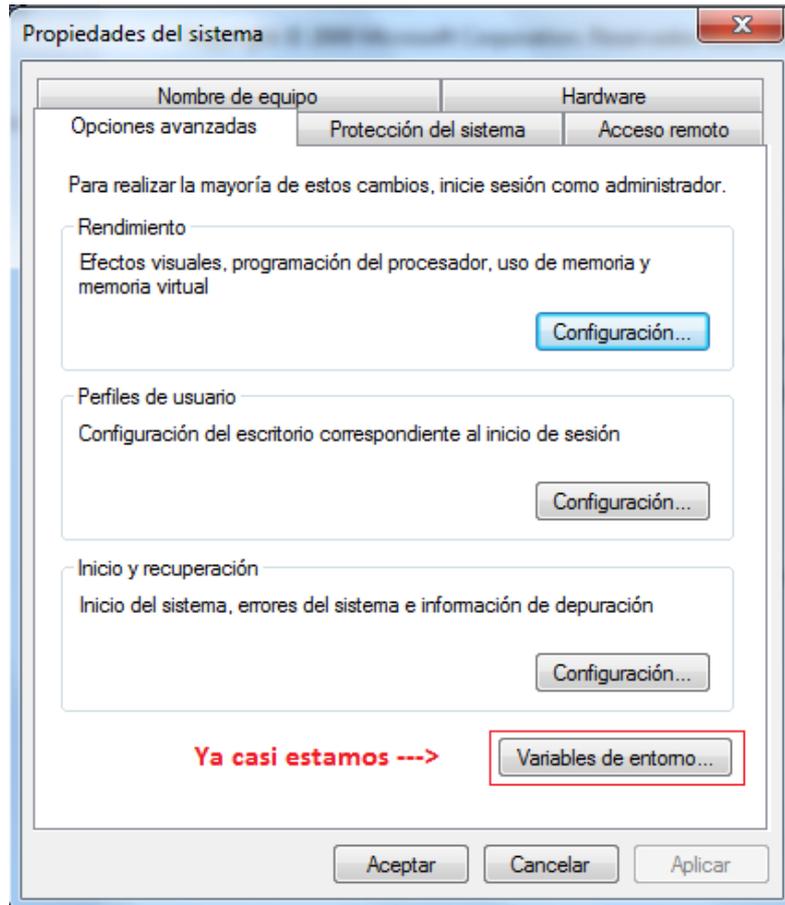
Java SE Development Kit 6 Update 25		
Product / File Description	File Size	Download
Linux x86 - RPM Installer	76.85 MB	jdk-6u25-linux-i586-rpm.bin
Linux x86 - Self Extracting Installer	81.11 MB	jdk-6u25-linux-i586.bin
Linux x64 - RPM Installer	77.06 MB	jdk-6u25-linux-x64-rpm.bin
Linux x64 - Self Extracting Installer	81.36 MB	jdk-6u25-linux-x64.bin
Solaris x86 - Self Extracting Binary	81.00 MB	jdk-6u25-solaris-i586.sh
Solaris x86 - Packages - tar.Z	136.67 MB	jdk-6u25-solaris-i586.tar.Z
Solaris SPARC - Self Extracting Binary	85.96 MB	jdk-6u25-solaris-sparc.sh
Solaris SPARC - Packages - tar.Z	141.11 MB	jdk-6u25-solaris-sparc.tar.Z
Solaris SPARC 64-bit - Self Extracting Binary	12.24 MB	jdk-6u25-solaris-sparcv9.sh
Solaris SPARC 64-bit - Packages - tar.Z	15.58 MB	jdk-6u25-solaris-sparcv9.tar.Z
Solaris x64 - Self Extracting Binary	8.49 MB	jdk-6u25-solaris-x64.sh
Solaris x64 - Packages - tar.Z	12.25 MB	jdk-6u25-solaris-x64.tar.Z
Windows x86 windows 32 bits	76.66 MB	jdk-6u25-windows-i586.exe
Windows x64 windows 64 bits	67.27 MB	jdk-6u25-windows-x64.exe

Este paso es opcional. La variable de entorno sirve únicamente para que desde la consola de Windows puedan compilar y ejecutar archivos java.

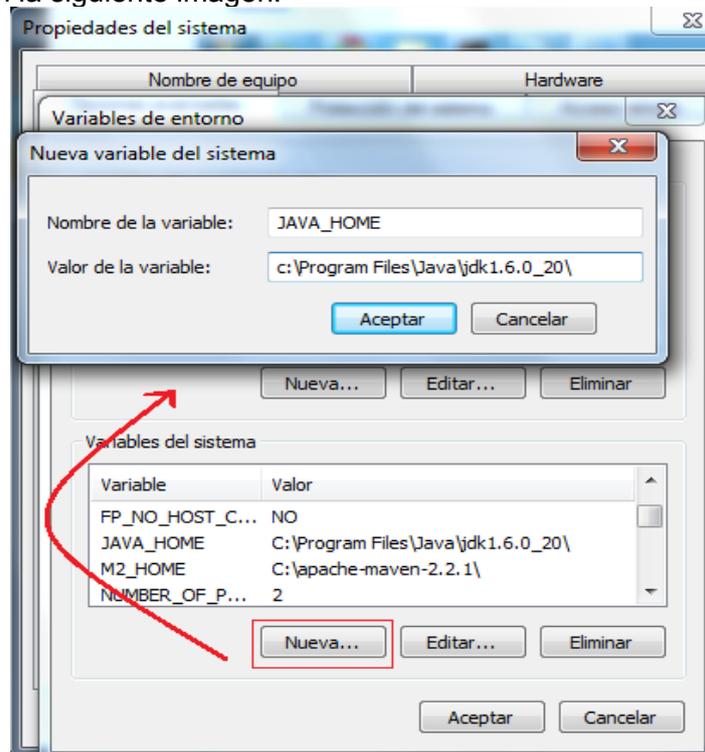
Para agregar una variable de entorno en Windows 7 deben ir a:

Panel de control > Sistema > Configuración avanzada del sistema > Variables de entorno.

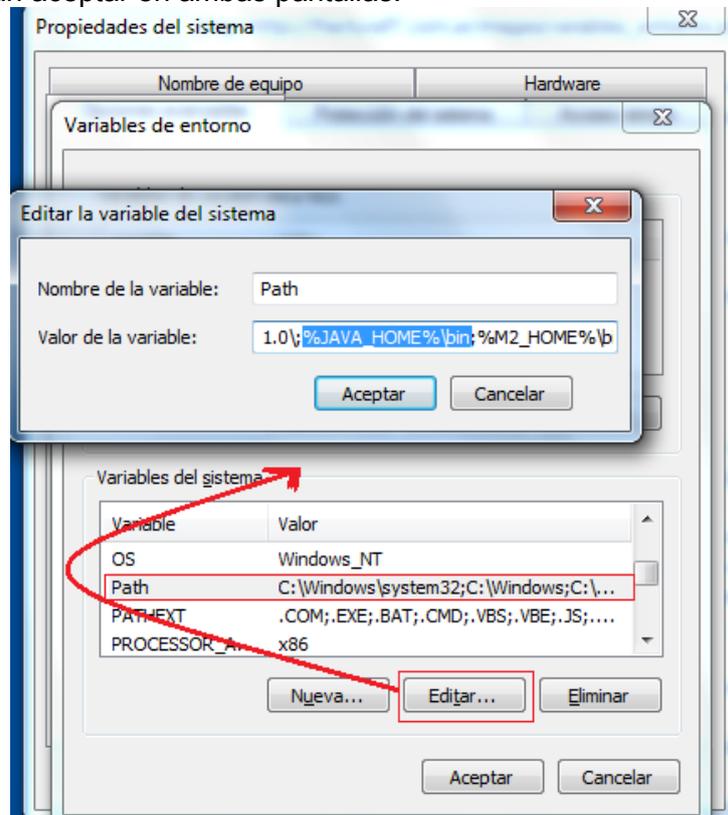
Al acceder a la Configuración avanzada del sistema vemos la siguiente pantalla:



En la variable "JAVA_HOME" el valor deberá ser sí o sí la ruta de acceso a la carpeta donde instalaron el JDK en su sistema. Les debería quedar algo similar a la imagen. Al presionar aceptar verán la nueva variable dentro de la lista de "Variables del Sistema" tal cuál figura en la siguiente imagen.



Ahora tenemos que modificar la variable "Path". Al presionar el botón "Editar..." vamos al final del todo de la variable y agregamos lo siguiente ";%JAVA_HOME%\bin;", siendo "JAVA_HOME" el nombre de la variable que agregaron anteriormente. Por último presionan aceptar en ambas pantallas.



Con esto ya tienes Java bien instalado en tu máquina. Ahora vamos con la herramienta de desarrollo.

5.3.2 Instalando Eclipse

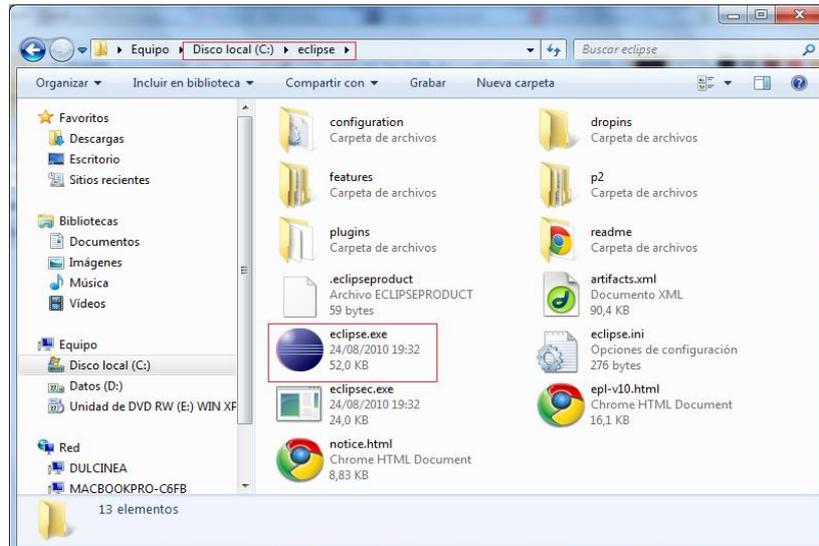
En el siguiente link podrán ver las versiones disponibles para descargar:
<https://www.eclipse.org/downloads/>

Es muy importante que sean consistentes con la versión de Java que descargaron.

Demás está decir que si seleccionaron el JDK de 32 bits deberán seleccionar el Eclipse de 32 bits. Lo mismo para las correspondientes versiones de 64 bits.

Cuando hayan terminado de descargar el archivo verán que es un .zip y no tiene un instalador. Esto se debe a que el programa no hace falta instalarlo, lo colocan donde deseen.

El contenido del .zip es lo siguiente:



5.3.3 Instalar el SDK de Android

Para este tutorial se mostrará como instalar el SDK de Android en la versión r24.4 (versiones posteriores también debería de funcionar, pero en las antiguas hay algunas cosas que cambian ligeramente) en Eclipse.

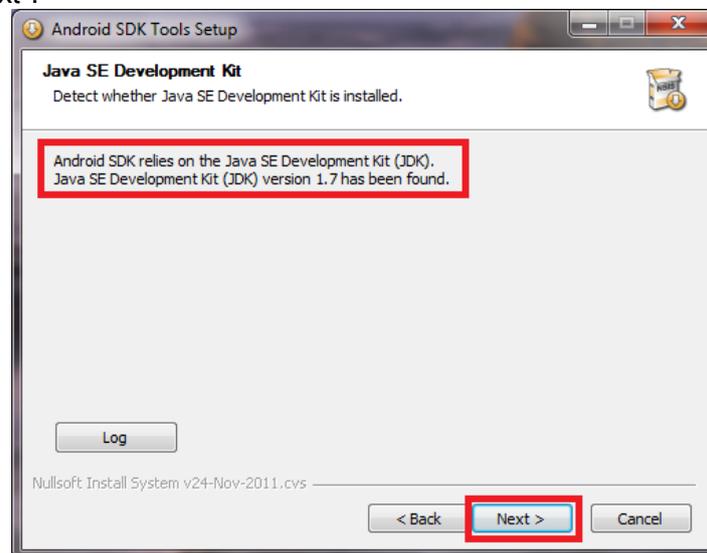
El SDK de Android es un plugin de Eclipse y se descargaba desde:

<http://developer.android.com/sdk/installing/index.html>

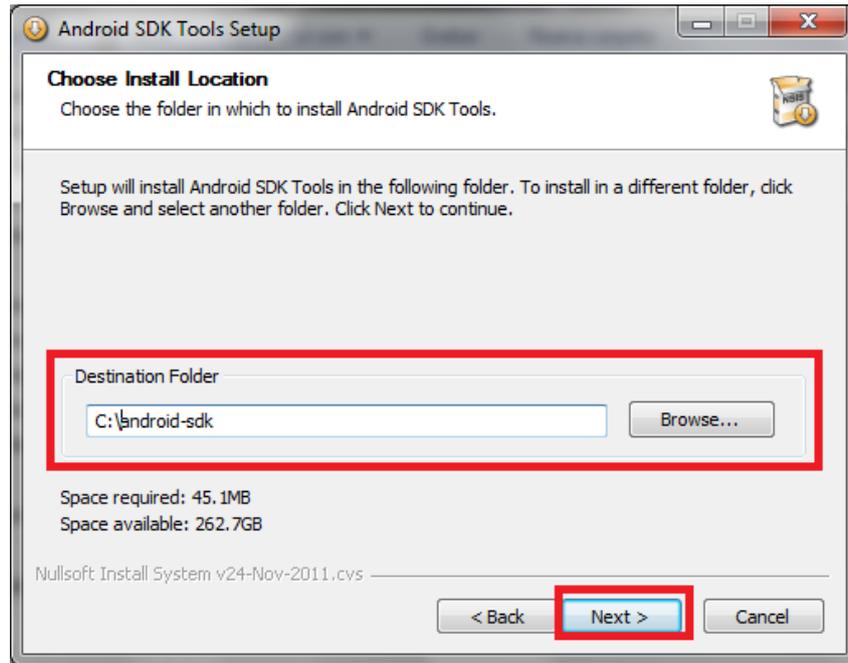
Ahora está Android Studio, el link para descargar el SDK con Eclipse ya no está disponible, este quedó obsoleto, pero dispongo del instalador correspondiente.

Ejecutamos el instalador y pulsamos una vez en “Next”.

Si detecta el instalador que el JDK de Java está instalado. Sería lo ideal y pulsamos otra vez en “Next”.



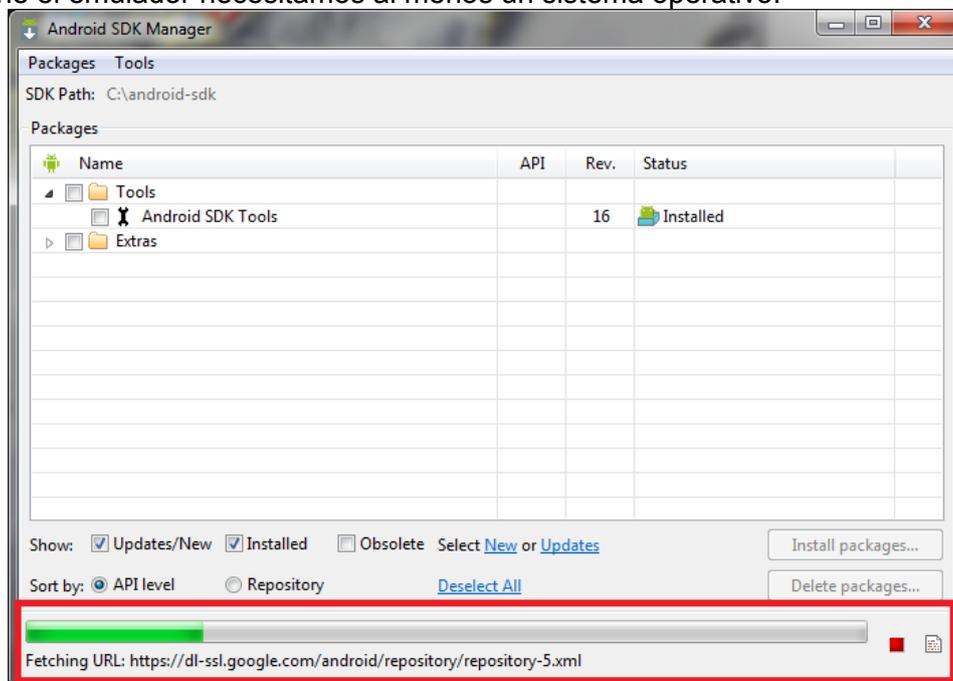
Una vez llegado al punto de elegir la ruta. Bastará con poner “C:\android-sdk”.



Ya pulsamos las veces que faltan “Next” hasta que se instale. Antes de terminar, cuando el instalador llegue al final, podemos ejecutar el SDK tildando el checkbox “Start SDK Manager”.

En este momento habremos instalado el SDK de Android. Se nos abrirá una parte del SDK que se llama “Android SDK Manager”, que estará vacío, es decir, que no tiene ningún sistema operativo Android como tal.

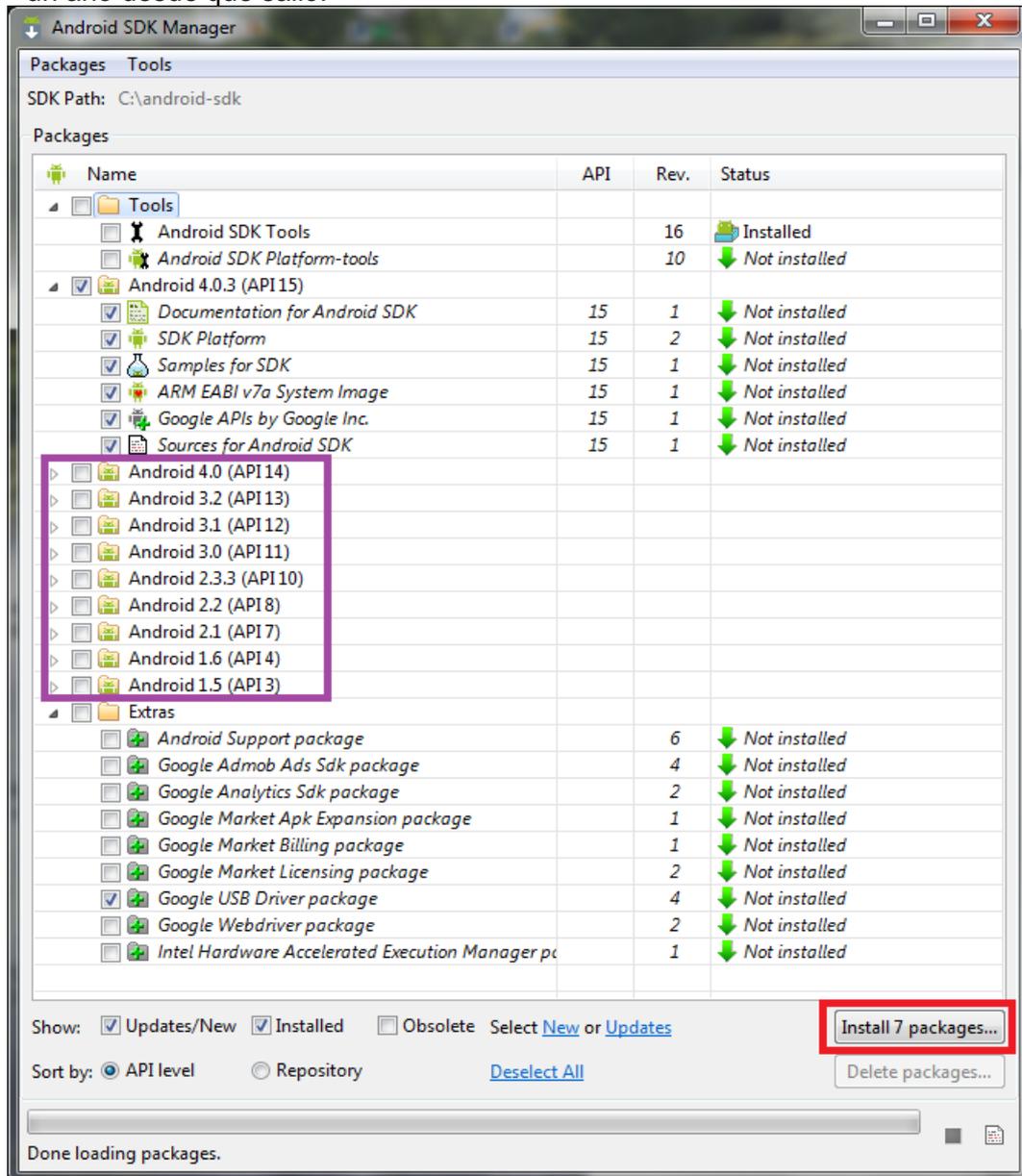
Esperamos a que la barra busque en Internet lo que podemos descargar. Para que funcione el emulador necesitamos al menos un sistema operativo.



Cuando acabe de buscar, se nos darán algunas opciones ya marcadas para instalar. Por defecto querrán instalarse los drivers de USB para poder probar las aplicaciones en algún dispositivo físico con Android que tengamos, y la última versión de Android hasta la fecha.

Aunque suena muy jugosa la última versión de Android, usarla implica aceptar una serie de problemas:

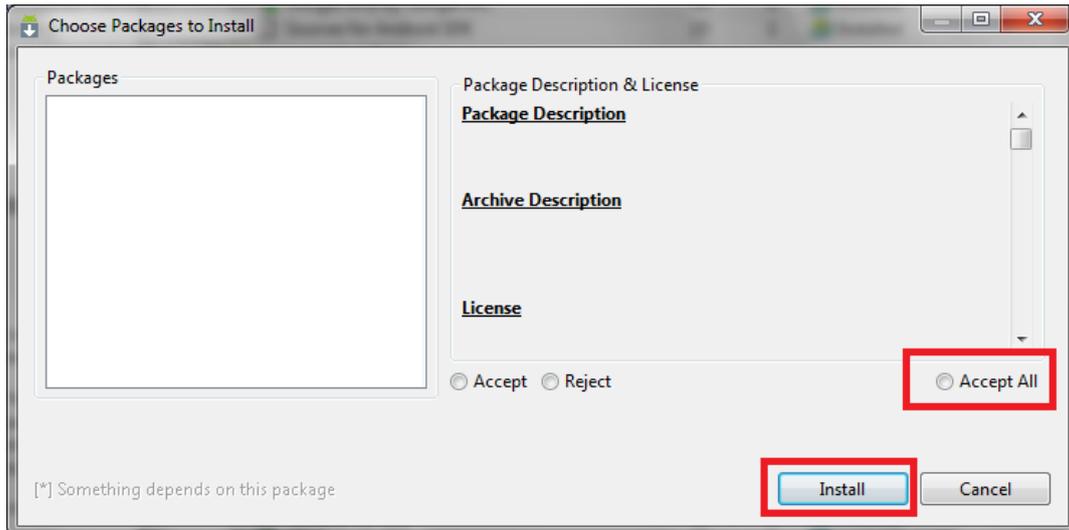
- Si tu ordenador no es muy potente, puede que desesperes de lo lento que va a ir el emulador de Android con el nuevo sistema operativo.
- Si quieres subir a la Market (Play se llama ahora) la aplicación que desarrolles para esta versión, te vas a encontrar con que muy poca gente se la va a descargar, pues casi nadie tiene esta última versión hasta que no pasa cerca de un año desde que salió.



Para elegir una versión de Android miramos la gráfica de uso del mercado: <http://developer.android.com/resources/dashboard/platform-versions.html>.

Por lo que recomiendo apostar fuerte e instalar la versión más usada, con lo que aseguramos rapidez en el emulador y que pueda ser usado por mucha gente. Además las versiones más nuevas de Android ejecutan aplicaciones escritas para versiones del sistema operativo más antiguas, pero no al revés.

Pulsamos el botón de "install X packages...". Se abrirá una nueva ventana en la que aceptaremos todo, se puede pulsar sobre "Accept All" y pulsamos "Install".



Tenemos dos programas que son individuales, nada vinculados:

- **Por un lado tenemos el SDK de Android**, la ventana que tenemos descargando es el asistente de descargas, forma parte del SDK de Android.
- **Por otro lado tenemos Eclipse**, si lo abrimos veremos que no hay nada nuevo. Quiere decir que no hay nada vinculado con Eclipse, por lo tanto, tenemos que vincularlo para que se reconozcan entre ellos.

Cuando se haya descargado todo lo que queremos cerramos la ventana de “Android SDK Manager”.

5.3.4 Instalar el plugin ADT

Ahora trabajaremos con Eclipse, donde hay que vincular el SDK a nuestro entorno de desarrollo. Esto se realiza mediante la instalación de un **plugin para Eclipse denominado ADT**.

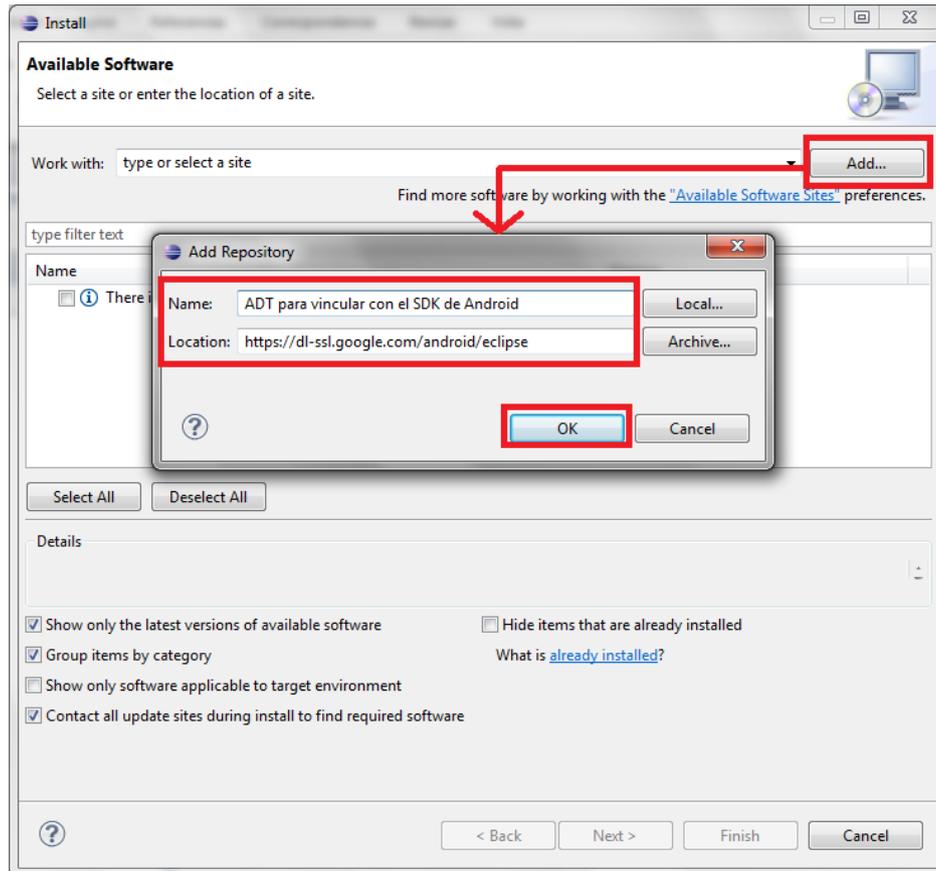
En el menú vamos a “Help”, ahí elegimos “Install New Software...”

Se nos abrirá una nueva ventana. Pulsamos en “Add...”, y en la mini-ventana emergente que nos sale le ponemos el nombre que queramos que tenga el plugin, como por ejemplo: “ADT para vincular con el SDK de Android”. En el campo URL la siguiente dirección para descargar el plugin del ADT:

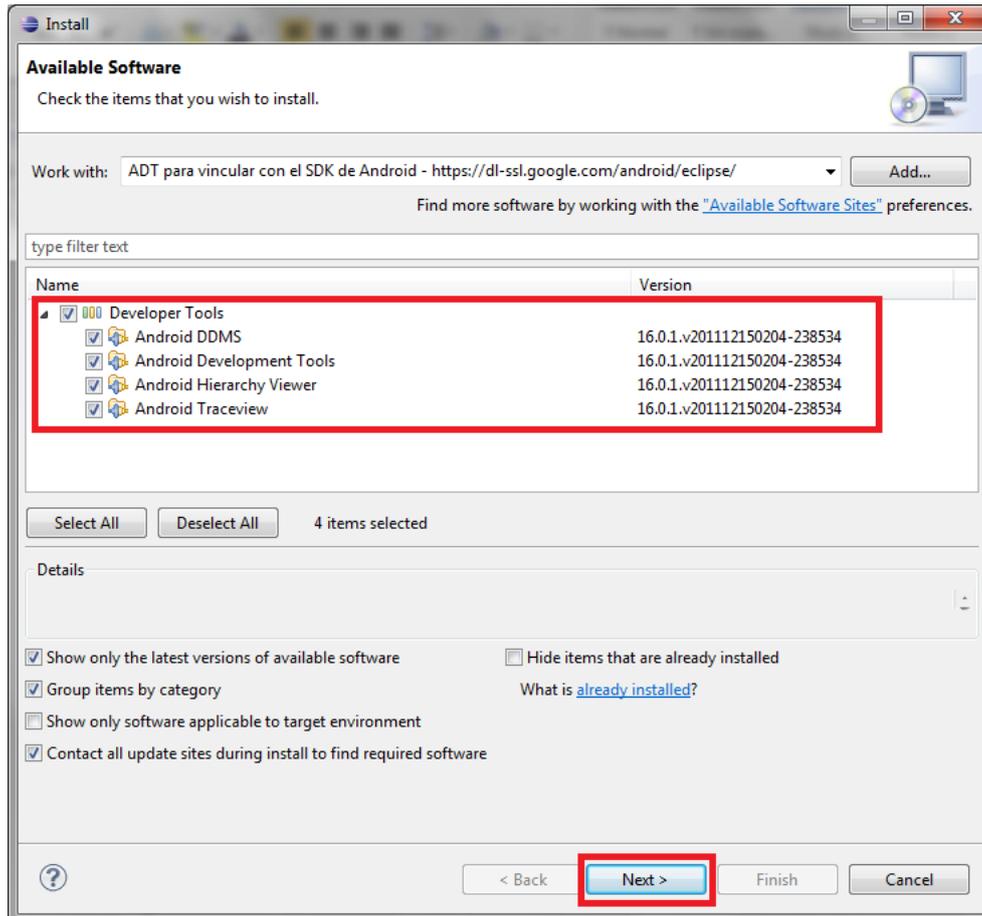
<https://dl-ssl.google.com/android/eclipse/>

Nota: No deberíamos fiarnos de esta dirección por si se descarga software no deseado. Si quieres copiarla de la fuente original de Android Developers está en: <http://developer.android.com/sdk/installing/installing-adt.html>

Cuando acabemos aceptamos la mini-ventana emergente.



Ahora marcamos la casilla de "Developer Tools" para que se marquen todas y pulsamos en siguiente.

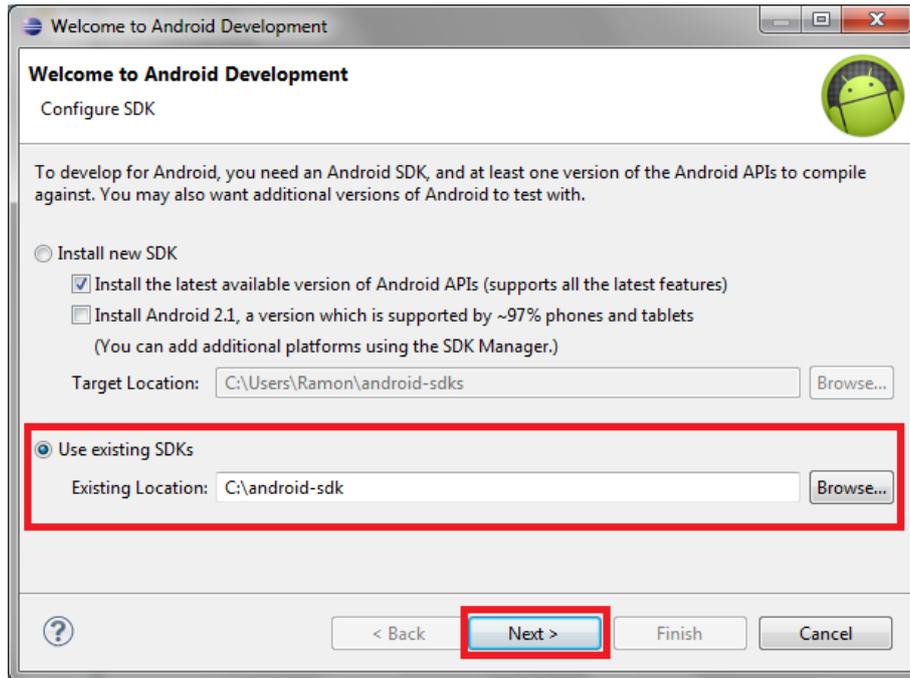


En el siguiente paso desmarcamos el check “Contact all update sites during install to find required software” y pulsamos directamente a siguiente. Luego **aceptamos** la licencia y pulsamos terminar para que comience la instalación.

Al terminar nos pedirá reiniciar Eclipse. Le decimos que reinicie ahora.

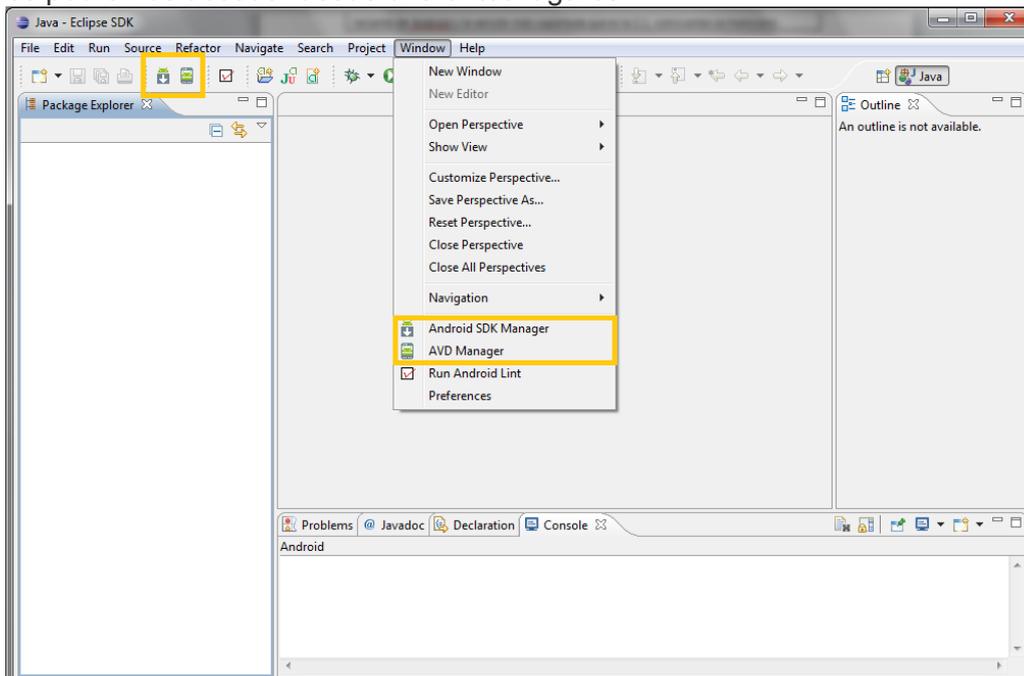
Al reiniciarse Eclipse nos saldrá una ventana que nos dirá si queremos instalar el SDK. El SDK ya está instalado, con lo que pulsamos en “Use existing SDKs” y elegimos la carpeta donde instalamos anteriormente el SDK.

Nota: Podíamos haber empezado instalando el ADT y que luego se encargara Eclipse de instalar el SDK, pero así no se ve clara la diferencia de programas; que el SDK es una cosa y el Eclipse es otra, que están vinculadas a través del ADT. Una observación referente a esta ventana: da la opción de instalar el SDK con la versión más reciente de Android y la versión más soportada.



Ya tenemos vinculados Eclipse con el SDK de Android.

Con un vistazo rápido a Eclipse veremos las nuevas herramientas instaladas, a las cuales podremos acceder desde diferentes lugares.



5.4 Descarga e instalación de IBM Rational Rhapsody Developer para Java

5.4.1 Preparación de la instalación de Rational Rhapsody Developer para Java

Rational Rhapsody Developer para Java requiere el JDK para compilar y ejecutar el código Java generado utilizando la configuración predeterminada.

Vea las notas de la versión de Rational Rhapsody (Readme.htm) para las versiones compatibles de Java SDK.

5.4.2 Instalar Rational Rhapsody para Windows

Para instalar Rational Rhapsody en un sistema Windows:

1 - En el directorio de instalación de Rational Rhapsody, haga clic en el archivo .msi para iniciar el asistente de instalación.

2 - El programa de instalación muestra una ventana de bienvenida al asistente de instalación. El asistente le guiará a través del proceso de instalación.

Nota:

Si el asistente de instalación detecta una versión de Rational Rhapsody previa en el sistema, el asistente muestra una ventana con las opciones **Modify, Repair o Remove**.

3 - Se muestra el contrato de licencia de software de IBM. Lea el acuerdo y haga clic en el botón "I accept the terms in the license agreement" y en Siguiente para continuar.

4 - Seleccione la edición de **Rational Rhapsody Developer** para instalar (necesario para los usuarios de Eclipse).

5 - Acepte el directorio de instalación predeterminado o buscar y seleccionar un path diferente.

6 - Si va a agregar una edición que requiere un lenguaje, como desarrollador, es necesario seleccionar uno o más de los cuatro lenguajes soportados (C++, C, Java y Ada) para la salida de código generado. Si desea seleccionar los ajustes individuales del sistema operativo, seleccione "Check for Real Time OS Settings". De lo contrario, dejar el check seleccionado para una configuración del entorno predeterminado y simplificar el proceso de instalación.

7 - Si ha seleccionado el lenguaje Java, seleccione el entorno de desarrollo como IBM Java SDK 6.0 SR1, JDK 1.6, 1.5, o 1.4 y haga clic en Siguiente.

8 - Si ha seleccionado el lenguaje Java, seleccione la ruta (ubicación) del Java Developer Environment y haga clic en Siguiente.

9 - Seleccione alguno de los productos add-ons. Usted tiene que comprar una licencia específica para muchos de los productos seleccionados.

10 - Seleccione el tipo de instalación:

Typical: Incluye las opciones más comunes. Para iniciar la instalación, haga clic en instalar. Para realizar cambios, haga clic en Atrás y realice los ajustes.

Custom: Seleccione este tipo de instalación si desea seleccionar sus propios elementos específicos de instalación.

11 - Seleccione el tipo y la ubicación de su licencia de Rational Rhapsody.

12 - Revise la descripción de la instalación y haga clic en Instalar si es correcto.

13 - Se muestra el estado de la instalación. Haga clic en Finalizar para completar la instalación.

5.4.3 Instalación de la clave de licencia (Original)

Pasos para obtener una clave de licencia (original) de Rational Rhapsody:

1. Para obtener las instrucciones de licencias de software de IBM, visite el sitio para soporte del Licenciamiento de Rational en

<http://www-01.ibm.com/software/rational/support/licensing/>

2. Para obtener su clave de licencia, haga clic en el enlace del Centro de claves de licencia.

3. Haga clic en Continuar.

4. Para iniciar sesión en el Centro de claves de licencia, escriba su dirección de correo electrónico y su contraseña. Esta contraseña fue enviada a su dirección de correo electrónico cuando se añadió su nombre al centro de claves de licencia.

5. Siga las instrucciones de su software de IBM Rational.

Para obtener más información, consulte la Guía de inicio rápido en el Centro.

Las siguientes configuraciones de Rational Rhapsody requieren tipos específicos de licencia:

- Si usted tiene las licencias de Rational Rhapsody C y la de C++, no se puede abrir un modelo de C en C++ o viceversa sin la licencia de Rational Rhapsody multi-lenguaje.
- Java está disponible sólo en una licencia multi-lenguaje. Puede abrir un modelo Java en Rational Rhapsody C o C++ cuando se utiliza la licencia multi-lenguaje.
- El plugin de Eclipse para Rational Rhapsody sólo está disponible con la licencia de Rational Rhapsody multi-lenguaje.

5.4.4 Instalación de la clave de licencia (evaluación por 30 días)

Pasos para la instalación de Rational Rhapsody (software con ciertas limitaciones) y obtención de una clave de licencia (evaluación por 30 días):

1. Visite el sitio para descarga de pruebas y demos en <http://www14.software.ibm.com/webapp/download/byproduct-fulllist.jsp>
2. En la lista busque a Rational Rhapsody.
3. Elegimos y hacemos click sobre una de las versiones de evaluación.
4. Esta descarga es sin cargo, pero requiere registro. Si usted no tiene un ID de usuario, se deberá registrar ahora.
5. Luego del registro ingrese el lenguaje bajo el que se descargará el producto y presione Continuar.
6. Leer la licencia, tilde el checkbox para aceptar la licencia y presione Confirmar.
7. Descargue la evaluación de IBM Rational Rhapsody y la clave de evaluación por 30 días.

The screenshot shows the IBM Rational Rhapsody evaluation page. The main heading is "Evaluación Rational Rhapsody". Under the "Descargas" tab, there is a section titled "Evaluación Rational Rhapsody para Red Hat Entrprs Lnx RHEL Base Srvr, Red Hat Entrprs Lnx RHEL Adv Pltfrm, SUSE Linux Enterprise Server (SLES), Windows 7 Enterprise, Windows Server 2008 Standard Ed, Windows Vista Business, Windows Vista Enterprise, Windows Vista Ultimate, Windows XP Professional" with various language options. Below this, there are instructions for downloading using http or Download Director. To the right, under "¿Necesitas ayuda?", there are links for "Regístrate apoyo (sólo inglés)", "Regístrate y descarga de software FAQ", and "Soporte de descarga de software (sólo en inglés)". At the bottom, the "Instrucciones de instalación" section includes a "Prueba instrucciones clave" sub-section with details about the trial key and activation process.

En nuestro caso elegimos Windows.



Como dijimos al ser propietario se necesita descargar la clave, o licencia para probar el producto:



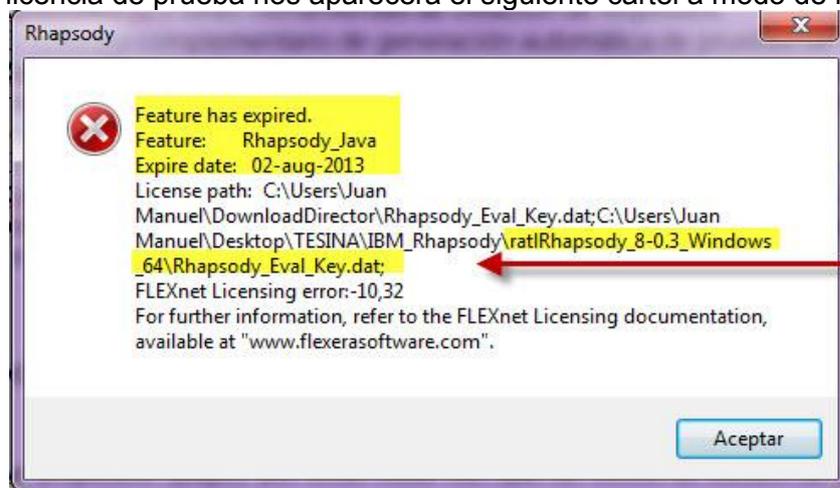
La cual necesitaremos cuando queramos iniciar por primera vez la herramienta.

8. Descomprimir el archivo e instalar Rational Rhapsody.

9. En la información de licencia, elegir la opción de archivo de licencia del cliente y especificar el path donde se encuentra el archivo con la clave de evaluación anteriormente descargada.

Este software es propietario, por ende cada vez que se vence la versión de prueba (TRIAL) hay que descargarse el archivo de licencia Rhapsody_Eval_key.dat.

Al vencerse la licencia de prueba nos aparecerá el siguiente cartel a modo de informe.



5.5 Configuración de Eclipse y de Rational Rhapsody

5.5.1 Requisitos de integración de la plataforma

El software que debe ser utilizado para crear una plataforma de integración de Eclipse en pleno funcionamiento con Rational Rhapsody es el siguiente:

- Eclipse.
- JDT plugin para el desarrollo Java TM.
- Los compiladores necesarios para el desarrollo del lenguaje o los lenguajes que está utilizando.

- El plugin de Eclipse para Rational Rhapsody requiere algunos pasos de configuración adicionales. La siguiente información debe ser considerada cuando se instale Eclipse y Rational Rhapsody:
- Rational Rhapsody o Eclipse se puede instalar en cualquier orden, pero es preferible tener el entorno de desarrollo instalado antes de instalar Rational Rhapsody.
- La plataforma de integración con Eclipse puede usarse para el desarrollo de aplicaciones en modelos de Rational Rhapsody C, C++ o Java.
- La plataforma de integración de Rational Rhapsody para Eclipse requiere una licencia de Rational Rhapsody multi-lenguaje.
- La versión independiente de Rational Rhapsody y la plataforma Rational Rhapsody dentro de Eclipse utilizan el mismo repositorio de modo que si usted quiere puede cambiar entre las dos interfaces.

5.5.2 Vinculación de Rational Rhapsody para Eclipse

Puede sincronizar el trabajo de modelado en IBM Rational Rhapsody con su código de trabajo de desarrollo en Eclipse. Esto puede hacerse de dos maneras:

- ejecutando Rational Rhapsody como un plug-in de Eclipse (opción utilizada).
- ejecutando Rational Rhapsody como una aplicación independiente que se comunica con Eclipse.

Importante: Para asegurarse que vea su modelo de Rational Rhapsody en el Model Browser en su integración de Eclipse, utilice el mismo nombre para el proyecto Eclipse y su modelo de Rational Rhapsody.

5.5.3 Instalando los plugins para integración con Eclipse

Después de que todo el software se ha instalado, utilice las facilidades de Eclipse para vincular la funcionalidad de Rational Rhapsody en el IDE Eclipse:

- 1 - Abra Eclipse y seleccione **Help > Install New Software...**
- 2 - Haga clic en la pestaña **Available Software** y haga clic en **Add Site**.
- 3 - En la ventana Add Site, haga clic en **Local** para mostrar los directorios de su ordenador.
- 4 - Vaya a la ruta de instalación de Rational Rhapsody y resalte la carpeta Eclipse. Haga clic en Aceptar.
- 5 - Expandir la lista de selecciones de Rational Rhapsody y Eclipse en la pestaña Available Software.
- 6 - Seleccione sólo uno de los plugins de Rational Rhapsody para Eclipse:
 - **Rhapsody Platform Integration** permite a los desarrolladores trabajar en un proyecto Rational Rhapsody completamente dentro de Eclipse. Rational Rhapsody no necesita estar abierto para esta implementación. Esta integración se puede utilizar para el desarrollo de aplicaciones en C, C++ o Java en un entorno de Windows solamente (opción utilizada).
 - **Rhapsody Workflow Integration** permite al desarrollador de software trabajar en Rational Rhapsody y utilizar algunas de las características de Eclipse a través de opciones de menú Rhapsody Rational. Esta integración se puede utilizar para desarrollar en C y C++, tanto en entornos Windows como en Linux. Eclipse y Rational Rhapsody deben estar abiertos cuando el desarrollador está utilizando esta integración.
- 7 - Seleccione el tipo de integración que desea utilizar y haga clic en Instalar. Si selecciona una integración que incluye la opción "Workbench", tiene la intención de utilizar la versión de Wind River de Eclipse.

Nota: Sólo una versión de lanzamiento de Eclipse debe estar instalado en su ordenador. Para cargar otra versión posterior, primero debe extraer la versión actual de Eclipse y a continuación, ejecutar la instalación para la nueva versión de Eclipse.

8 - El programa muestra una ventana que indica que trata de "resolver las dependencias" y luego muestra la licencia para el plugin seleccionado. Examine la licencia, haga clic en la opción "Acepto los términos del contrato de licencia" y haga clic en Finalizar.

9 - Cuando haya finalizado la instalación, abrir como administrador la línea de comandos y parados en la carpeta donde se instaló Rational Rhapsody ejecutar "regsvr32 RhapsodyServer.dll" y reinicie el Eclipse.

5.5.4 Comprobación de la configuración de Eclipse

Para comprobar si Rational Rhapsody está conectado a su instalación de Eclipse:

1 - Abra Eclipse.

2 - Seleccione **Help > About Eclipse SDK**. El icono de Rational Rhapsody deberá aparecer en la ventana.

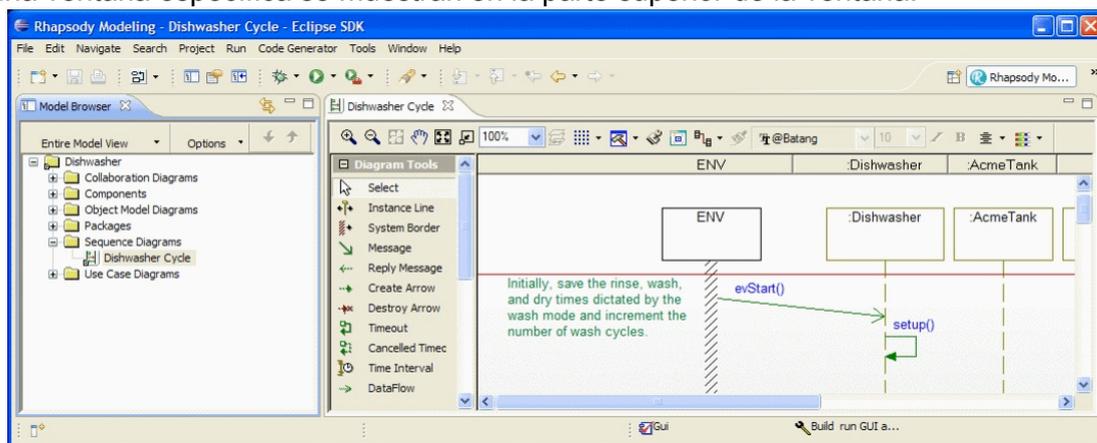
Si no se muestra el icono, compruebe las instalaciones de los paquetes de software necesarios para estar seguros de que están funcionando.

5.6 Utilizando la plataforma de integración con Eclipse

Cuando se utiliza la plataforma de integración con Eclipse, los elementos de Rational Rhapsody estándar, tales como el navegador, el área de visualización de diagramas, y la ventana de salida se muestran en la interfaz de usuario de Eclipse. Estos elementos de la interfaz se pueden manipular como cualquier otro plugin dentro de Eclipse. Casi todas las funciones de Rhapsody están disponibles cuando se utiliza la integración de la plataforma, pero hay algunas excepciones. Ciertas funciones Rational Rhapsody se acceden de manera diferente dentro de Eclipse que en la versión independiente de Rhapsody, pero estas diferencias son bastante intuitivas.

5.6.1 Perspectivas de Rational Rhapsody en Eclipse

Los elementos de la interfaz que se muestran en Eclipse tienen las mismas características que en la versión independiente excepto que los iconos asociados a una ventana específica se muestran en la parte superior de la ventana.



La plataforma de integración de Rational Rhapsody añade dos perspectivas dentro de Eclipse:

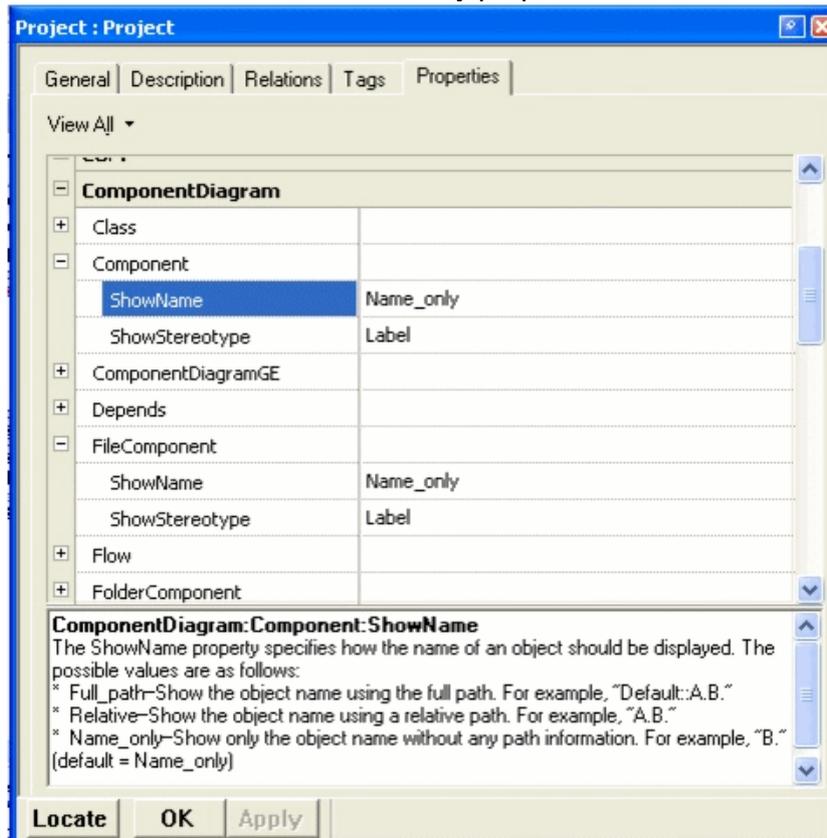
- Perspectiva de modelado Rational Rhapsody.
- Perspectiva de depuración Rational Rhapsody.

Cuando se crea un proyecto de Rational Rhapsody en Eclipse, la perspectiva de modelado Rhapsody se muestra automáticamente.

5.7 Configuración de Rational Rhapsody con propiedades

Las propiedades del proyecto son pares nombre-valor que se utilizan para personalizar aspectos de la interacción del ambiente y la generación de código. Puede establecer las propiedades existentes o crear conjuntos completos de otras nuevas para adaptar Rational Rhapsody a sus necesidades particulares.

Para ver las propiedades disponibles en el producto y cambiarlas si es necesario, consulte la solapa **Properties** de la ventana de **Features**, la cual utiliza una estructura de árbol para mostrar los temas, meta-clases y propiedades.



En la columna de la izquierda, se muestra la meta-clase y la propiedad. Al expandir una meta-clase, las propiedades correspondientes se enumeran en la columna de la izquierda, con sus valores actuales (si los hay) que figuran en la columna de la derecha.

Cada vez que una nueva propiedad está seleccionada, la definición de la propiedad se actualiza en el panel inferior. Esta definición muestra el nombre del tema, meta-clase, la propiedad y la definición de la propiedad.

5.8 Proyecto

Un proyecto incluye los diagramas, paquetes y configuraciones de generación de código que especifican el modelo y el código generado. El término proyecto es equivalente al modelo de Rational Rhapsody.

5.8.1 Perfiles del proyecto

Los perfiles de proyectos disponibles dependen del lenguaje de desarrollo y los productos complementarios de licencia para IBM Rational Rhapsody. En general, los

perfiles ofrecen, las etiquetas específicas de dominio y estereotipos predefinidos para simplificar su esfuerzo de desarrollo.

5.8.2 Perfil para aplicaciones Android

IBM Rational Rhapsody incluye un perfil para el desarrollo de aplicaciones para dispositivos basados en Android. Rational Rhapsody visualiza su aplicación Roundtripping los elementos agregados utilizando el Android plugin para Eclipse. Usted puede tomar ventaja de las capacidades de animación Rational Rhapsody mientras se ejecuta la aplicación con el simulador de Android.

El perfil ofrece:

- Visualización del SDK de Android, incluyendo diagramas de modelo de objetos.
- Los estereotipos se utilizan cuando los elementos de Android son Roundtripping en el modelo y cuando se realiza la generación de código específico de Android.

Nota: El nombre del perfil refleja la versión del SDK de Android que soporta. Por ejemplo, el perfil de Android23 proporciona soporte para la versión 2.3 del SDK de Android.

5.8.3 El uso de unidades de proyecto

Una unidad es cualquier elemento de un proyecto que se guarda en un archivo independiente.

Puede dividir el modelo en unidades hasta el nivel de la clase. La creación de unidades simplifica la colaboración en entornos de equipo. Con esta característica, usted tiene el control explícito sobre los nombres de archivo y los derechos de modificación en un sistema de gestión de la configuración.

En la siguiente tabla se muestran los elementos del proyecto que pueden ser unidades.

Elemento	Extensión del archivo	Unidad por Defecto ?
Actores	.cls	No
Componentes	.cmp	Yes
Paquetes	.sbs	Yes
Proyectos	.rpy	Yes
Clases	.cls	No
Objetos implícitos	.cls	No
Archivos	.cls	No
Diagramas (excepto diagramas de actividad y de estado)	Diagrama de Definición de Bloques (*.omd)	No
	Diagrama de Componentes (*.ctd)	No
	Diagrama de Colaboración (*.clb)	No
	Diagrama de Deploy (*.dpd)	No
	Diagrama de Bloques Internos	No
	Diagrama de Modelo de Objetos (*.omd)	No
	Diagrama de Secuencia (*.msc)	No
	Diagrama de Estructura (*.std)	No
	Diagrama de Casos de Uso (*.ucd)	No

5.9 Diseño de proyectos

Un proyecto consta de los elementos que definen el modelo, tales como paquetes, clases y diagramas. El navegador muestra estos elementos en una estructura

jerárquica que coincide con la organización de su modelo. IBM Rational Rhapsody utiliza estos elementos para generar el código para su aplicación final.

Un proyecto de Rational Rhapsody tiene los siguientes elementos de nivel superior:

- **Paquetes y paquetes de perfiles** contienen otros paquetes, componentes, actores, casos de uso, clases, los tipos de objetos, eventos, tipos, objetos, dependencias, restricciones, variables, diagramas de secuencia, diagramas de modelo de objetos, diagramas de colaboración, diagramas de casos de uso y diagramas de despliegue.
- Los **componentes** contienen las configuraciones, los archivos, y las variantes para diferentes líneas de productos de software.
- **Diagramas UML** por ejemplo, casos de uso, modelo de objetos, secuencia, colaboración, componentes y diagramas de despliegue.

5.9.1 Agregando elementos al modelo Rational Rhapsody

Usando la perspectiva de Modelado de Rhapsody en Eclipse, se puede utilizar los modelos del menú expandible para añadir elementos a tu modelo.

Procedimiento:

1. Con Eclipse Modelo Browser abierto, haga clic derecho en un elemento para el que desea agregar un elemento.
2. Seleccione **Add New** (tipo de elemento) en el menú en función del tipo de elemento seleccionado. Por ejemplo, **Add New > Package**. Para modificar un elemento, puede editar el nombre, descripción, tipo y código de implementación mediante la ventana **Features**.

5.9.2 Agregar elementos a las aplicaciones Android

Una vez que haya asociado un proyecto de Android con un modelo de Rational Rhapsody en Eclipse, se pueden agregar nuevos elementos de Android para su modelo.

Procedimiento:

1. Utilice el menú de Rational Rhapsody para añadir el elemento al modelo. Si está utilizando la función de DMCA (sincronización entre los modelos y el código), se genera automáticamente el código del proyecto Android. Los archivos de código generados se muestran en su proyecto Android.
2. Alternativamente, puede agregar el elemento a su código con el menú de Eclipse. Si está utilizando la función de DMCA, el elemento se añade automáticamente a su modelo de Rational Rhapsody. El elemento se mostrará como parte de su modelo en el navegador Rational Rhapsody.

5.9.3 Creación de paquetes

Puede crear paquetes para dividir el modelo en dominios funcionales.

Un proyecto de Rational Rhapsody debe contener al menos un paquete. Un modelo, que siempre incluye un paquete por defecto, llamado "Default", donde los elementos del modelo se guardan a menos que especifique un paquete diferente.

Al crear un paquete se le dará un nombre, un diagrama principal, una descripción (en el código generado, se convierte en un comentario) y opcionalmente se le puede seleccionar un estereotipo.

5.9.4 Creación de componentes bajo un paquete

Cuando se crea un proyecto Rhapsody, un componente llamado DefaultComponent es creado directamente por debajo del nivel de los proyectos. También puede crear componentes adicionales a este nivel jerárquico.

Además es posible crear un componente como parte de un paquete en el modelo. Una de las ventajas de este enfoque es que si sólo desea generar código para un paquete específico, sólo tienes que mirar a ese paquete.

Cuando se crea un componente en un paquete, se convierte automáticamente en el componente activo para el proyecto.

5.9.5 Crear diagramas

Se pueden crear diferentes tipos de diagramas, que se pueden agrupar bajo el proyecto o un paquete en la jerarquía del proyecto. Usted puede utilizar el menú **Tools** o el botón correspondiente en la barra de herramientas de diagramas:

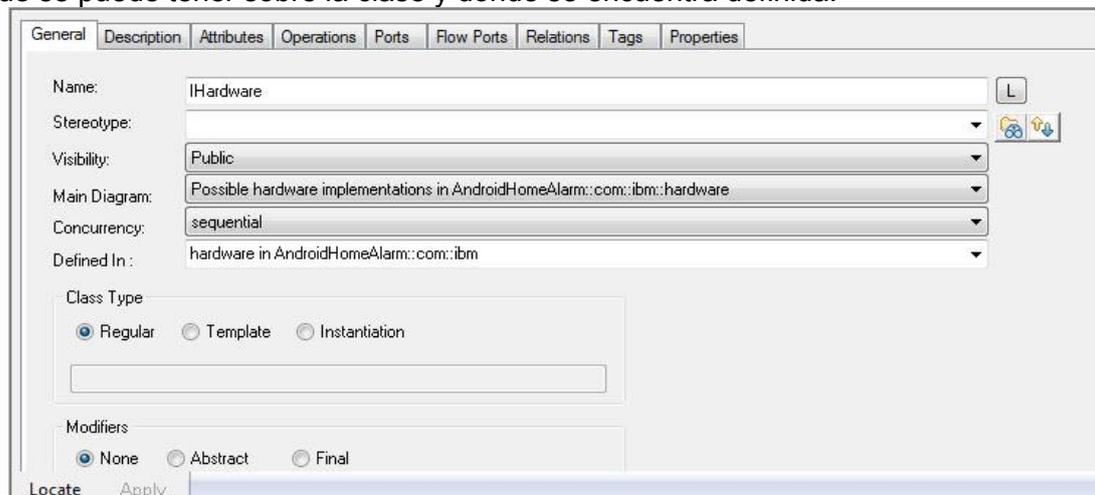
	Diagrama Modelo de Objetos		Diagrama de Componentes
	Diagrama de Estructura		Diagrama de Despliegue
	Diagrama Caso de Uso		Diagrama de Colaboración
	Diagrama de Secuencia		Diagrama de Panel

Ciertos perfiles tienen diagramas específicos para ellos. Por ejemplo, el diagrama de definición de bloque es específico para el perfil SysML.

5.9.6 Definición de los atributos de una clase

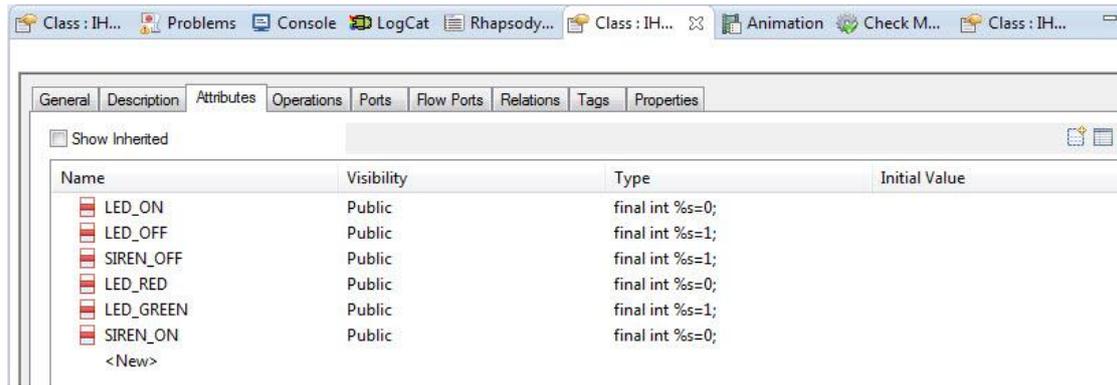
Las clases pueden contener atributos, operaciones, eventos, relaciones, componentes, superclases, tipos, actores, casos de uso, diagramas y otras clases.

Visualizar atributos “generales” como su nombre lo indica, tales como: nombre, estereotipo, la visibilidad, el diagrama donde se localiza, el tipo de acceso concurrente que se puede tener sobre la clase y donde se encuentra definida.



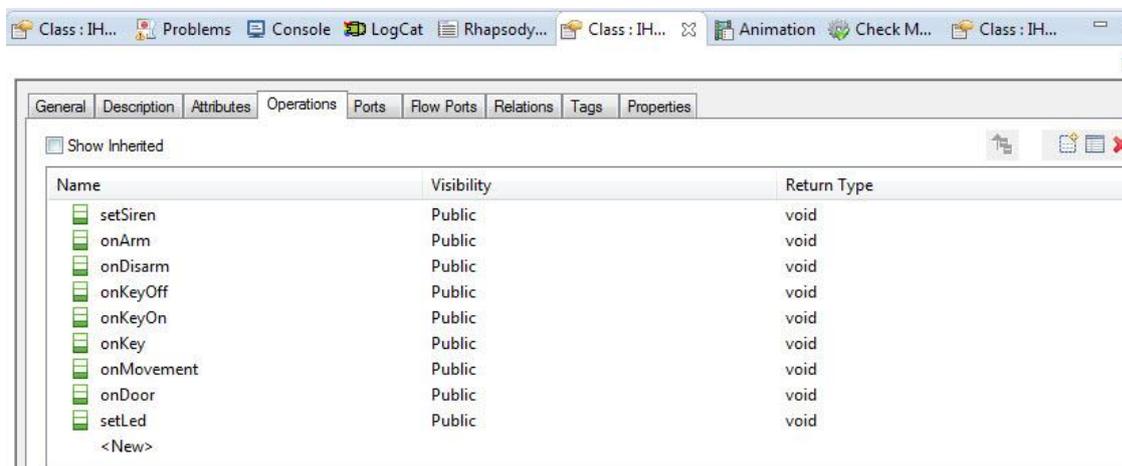
Puede utilizar Rational Rhapsody para generar automáticamente los métodos de los atributos (get y set), por lo que no es necesario definirlos por sí mismo.

La solapa **Atributos** de la ventana Features contiene una lista de los atributos que pertenecen a la clase y controles para su administración.

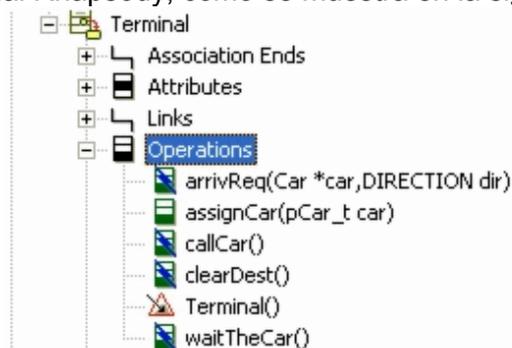


5.9.7 Administración de operaciones y creación de constructores

Utilice la solapa **Operations** de una clase o evento para agregar, editar o eliminar las operaciones a partir del modelo o de la vista actual del diagrama de modelo de objetos.



De la misma manera se pueden crear constructores con sus correspondientes argumentos. El nuevo constructor aparece en la categoría de operaciones de la clase en el navegador Rational Rhapsody, como se muestra en la siguiente figura:

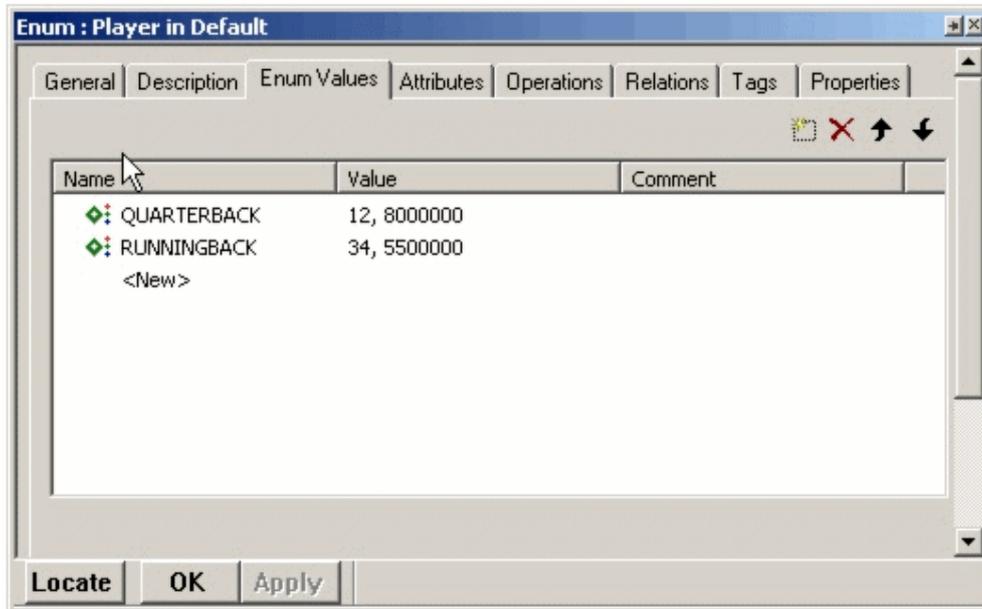


Utilice la ventana Features para cambiar las características de un constructor, incluyendo su nombre, visibilidad, sus argumentos y el código de inicialización.

5.9.8 Adición de enumeraciones de Java a un modelo

Para agregar enumeraciones de Java en el modelo, haga clic en el paquete para el que desea añadir las enumeraciones y seleccione **Add New > Enum**.

En su ventana de Features, seleccione la pestaña **Enum Values**, modificar las propiedades de los enumerativos.



Para agregar un enumerativo a un diagrama de modelo de objetos, arrastre la enumeración desde el navegador al diagrama.

5.9.9 Bloques estáticos de clases en un modelo

Java le permite definir bloques de código como algo estático. El código de estos bloques se ejecuta sólo una vez, cuando la clase se carga por primera vez. Rational Rhapsody le permite añadir bloques estáticos de clases en el modelo, y generar código apropiado para tales bloques.

El código para el cuerpo del bloque se introduce en la solapa **Implementation** de la ventana Features del bloque estático.

5.9.10 Comentarios Javadoc

Rational Rhapsody proporciona un mecanismo de generación de Javadoc que se basa en los siguientes elementos:

- Propiedades de Rational Rhapsody llamados DescriptionTemplate para elementos tales como clases y operaciones. El contenido de estas propiedades determina el aspecto de los comentarios Javadoc generados.
- Recuperación automática de elementos que se corresponden con las etiquetas Javadoc, tales como descripciones y argumentos de la operación.
- Las etiquetas especiales y palabras clave que se pueden utilizar para las etiquetas Javadoc estándar que no tienen los elementos de Rational Rhapsody, por ejemplo, @version.

Los comentarios Javadoc se agregan proporcionando valores para las distintas etiquetas que aparecen en la solapa **Tags**.

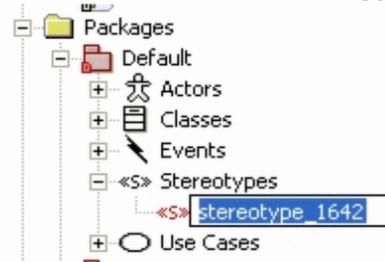
5.9.11 Definición de los estereotipos

Una metaclass es una clase cuyas instancias son clases. En otras palabras, como los objetos son instancias de una clase, las clases son instancias de una metaclass. Los estereotipos son clases que extienden metaclasses. Un estereotipo no puede utilizarse solo: existe únicamente si está asociado al menos a una metaclass y cada instancia de la metaclass está vinculada o no a una instancia del estereotipo.

Los estereotipos se muestran en el navegador y pueden ser propiedad de paquetes y perfiles.

Procedimiento:

1. En el navegador, haga clic derecho en el perfil o paquete al cual pertenece el estereotipo y luego seleccione **Add New > Stereotype**.



2. Introduzca un nombre para el nuevo estereotipo.
3. Haga doble clic en el nuevo estereotipo para abrir su ventana de Características.
4. Seleccionar una o más metaclasses a la que se aplicará el estereotipo.
5. Para los perfiles, si se está trabajando en un dominio con la terminología y notación especializada, seleccione el check **New Term** para crear una meta-clase.
6. Haga clic en **Ok**.

5.9.12 El establecimiento de la herencia de estereotipos

Con la herencia de estereotipos puede extender los estereotipos existentes. Los estereotipos pueden heredar de los estereotipos predefinidos o de los que se han creado.

El estereotipo derivado hereda las siguientes características de su estereotipo base:

- Aplicabilidad (elementos a los que se puede aplicar)
- Propiedades (incluye propiedades sobrescritas localmente)
- Etiquetas

Mientras que los valores iniciales de las propiedades para el estereotipo derivado son aquellos que fueron heredados del estereotipo base, los valores pueden ser anulados por el estereotipo derivado. Además se pueden agregar etiquetas al estereotipo derivado, y añadir elementos a la lista de elementos a los que se puede aplicar.

5.9.13 La asociación de los estereotipos con un elemento

Puede asociar los estereotipos con un elemento del modelo utilizando el campo **Stereotype** de la ventana de características de ese elemento.

El campo **Stereotype** incluye una lista de todos los estereotipos definidos en los perfiles y paquetes que pueden ampliar la meta-clase de ese elemento.

5.9.14 Definición de librerías de modelos

Al modelar aplicaciones en Rational Rhapsody, puede que le resulte útil tener en el modelo las clases que se utilizan a partir de librerías predefinidas.

Uno de los inconvenientes de la inclusión de este tipo de librerías en su modelo es que a menudo son bastante grandes y por lo tanto puede utilizar una gran cantidad de recursos si se carga en su totalidad en el modelo que se está trabajando. La función de librerías de modelos permite cargar sólo las clases que están referenciadas en el modelo.

Rational Rhapsody usa esta función para cargar selectivamente clases de librerías para los desarrolladores que utilizan el SDK de Android.

Las clases no utilizadas seguirán estando descargadas. Esto se indica por el símbolo U (unloaded) que aparece al lado de ellos.

5.9.15 Rational Rhapsody API

Se puede encontrar información detallada sobre las interfaces y los métodos contenidos en el Rational Rhapsody API, la cual se puede utilizar para realizar acciones Rational Rhapsody desde dentro de un script. Se proporcionan dos versiones

de la API: una API basada en COM que se puede utilizar con C++ o VB/VBA/VBScript, y una API Java que se puede utilizar para realizar acciones Rational Rhapsody desde un programa Java.

5.9.16 Características de la API de Java

La API de Java se utiliza para trabajar con modelos de Rational Rhapsody tanto en Windows y Linux, permitiendo escribir aplicaciones multiplataforma.

Para los detalles de la versión API de Java ver el Javadoc, que se puede encontrar en [directorio de instalación de Rational Rhapsody]\doc\Java_API\index.html.

Rational Rhapsody proporciona dos archivos que se pueden encontrar en el [directorio de instalación directorio]/Share/JavaAPI:

- Rhapsody.jar contiene las clases e interfaces Java.
- Aplicación Rhapsody.dll (o Rhapsody.so para Linux) es una nativa aplicación de las interfaces Java.

El archivo .jar debe incluirse en el CLASSPATH del proyecto de Java, y el .dll (o archivo.so) debe ser incluido en el path lib.

5.10 Proyectos Android y Rational Rhapsody sobre Eclipse

Antes de comenzar debe tener el paquete de inicio de Android SDK y el plugin ADT para Eclipse instalado en su ordenador.

5.10.1 Crear proyecto Rhapsody y asociarlo a un nuevo proyecto Android

- En el Package Explorer, click-derecho, **New → Rhapsody Project**.
- En el wizard, elija el nombre, lenguaje Java, tipo de proyecto Android, el path donde ubicarlo y el botón finish.
Además del Package Explorer el nuevo proyecto aparecerá en el Model Browser.
- En el Model Browser, abra el árbol en “nombre del nuevo proyecto” → Components → DefaultComponent → Eclipse Configurations → DefaultConfig.
- En DefaultConfig click-derecho, **Create IDE Project**.
- Cuando aparezca la ventana Asistente para configuración de Rhapsody, elija **New Project** y finish.
- En el cuadro de diálogo New Project, seleccione la opción **Android Application Project** y complete todos los campos requeridos del wizard.

El Package Explorer mostrara tanto el proyecto de Rational Rhapsody y el proyecto Android Eclipse, mientras que en el Model Browser se observa que el proyecto Rational Rhapsody ahora contiene lo siguiente:

- Un componente con el estereotipo AndroidComponent.
- Visualización de la biblioteca Android que incluye un diagrama de modelo de objetos.
- El perfil AndroidProfile.

Ahora usted puede comenzar a agregar elementos al proyecto Android.

5.10.2 Crear proyecto Rhapsody y asociarlo a un proyecto Android importado

Una vez creado el proyecto Rhapsody, se realizan los siguientes pasos:

- En el Package Explorer, click-derecho, Import....
- Cuando aparezca la ventana Asistente elija **Android → Existing Android Code Into Workspace**.
- Buscar el proyecto Android a importar, seleccionarlo y finish.
- En el Model Browser, abra el árbol en “nombre del nuevo proyecto” → Components → DefaultComponent → Eclipse Configurations → DefaultConfig.
- En DefaultConfig click-derecho, **Create IDE Project**.

- f) Cuando aparezca la ventana Asistente para configuración de Rhapsody, elija el proyecto importado y finish.

5.10.3 Importar un proyecto Rhapsody y asociarlo a un nuevo proyecto Android

- a) En el Package Explorer, click-derecho, **Import...**
- b) Cuando aparezca la ventana Asistente elija Rhapsody → Rhapsody Project.
- c) Buscar el proyecto Rhapsody a importar, seleccionarlo y finish.
Además del Package Explorer, el nuevo proyecto aparecerá en el Model Browser.
- d) En el Model Browser, abra el árbol en “nombre del nuevo proyecto” → Components → DefaultComponent → Eclipse Configurations → DefaultConfig.
- e) En DefaultConfig click-derecho, **Create IDE Project.**
- f) Cuando aparezca la ventana Asistente para configuración de Rhapsody, elija New Project y finish.
- g) En el cuadro de diálogo New Project, seleccione la opción Android Application Project y complete todos los campos requeridos del wizard.

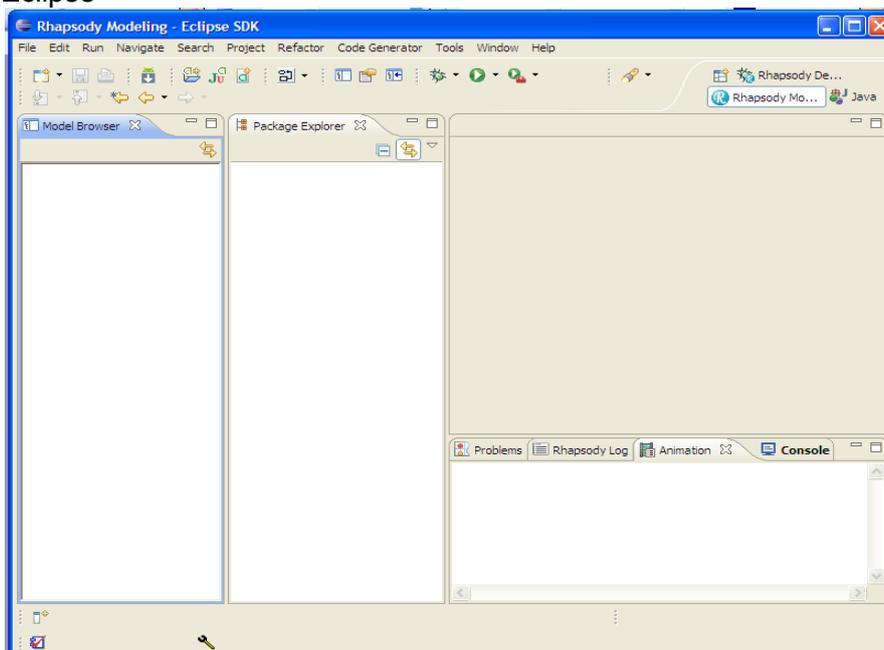
5.10.4 Importar y asociar proyectos Rhapsody y Android

Una vez importado el proyecto Rhapsody, se realizan los siguientes pasos:

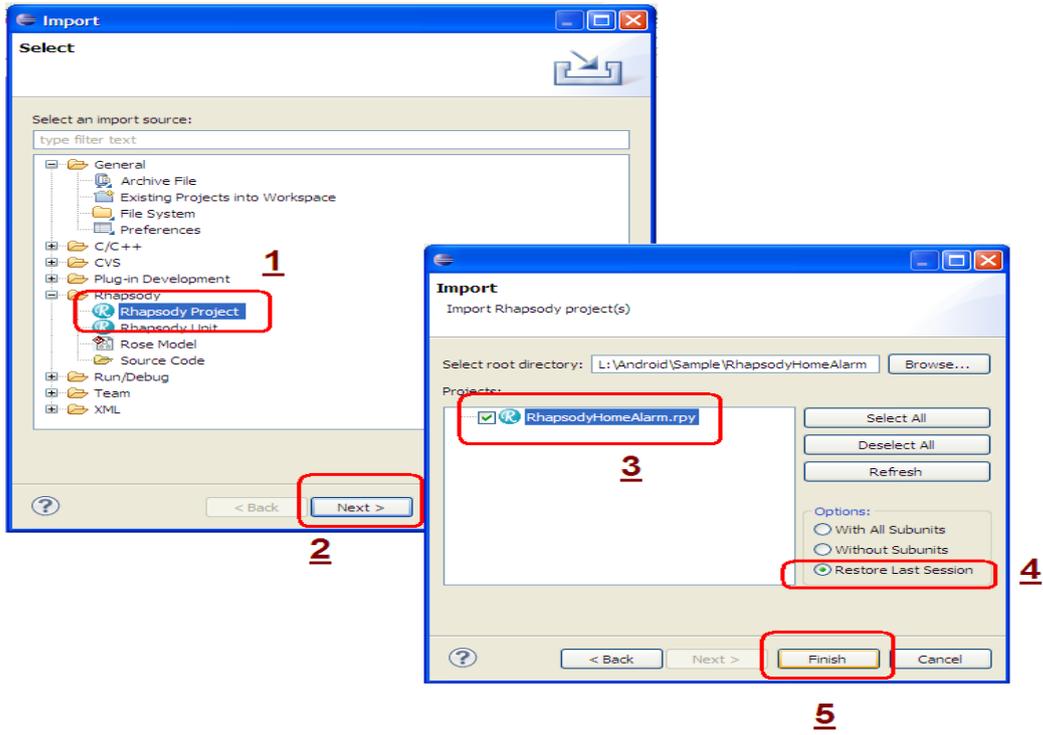
- a) En el Package Explorer, click-derecho, Import...
- b) Cuando aparezca la ventana Asistente elija **Android → Existing Android Code Into Workspace.**
- c) Buscar el proyecto Android a importar, seleccionarlo y finish.
- d) En el Model Browser, abra el árbol en “nombre del nuevo proyecto” → Components → DefaultComponent → Eclipse Configurations → DefaultConfig.
- e) En DefaultConfig click-derecho, **Create IDE Project.**
- g) Cuando aparezca la ventana Asistente para configuración de Rhapsody, elija el proyecto android importado y finish.

5.10.5 Ejemplo de Android Home Alarm

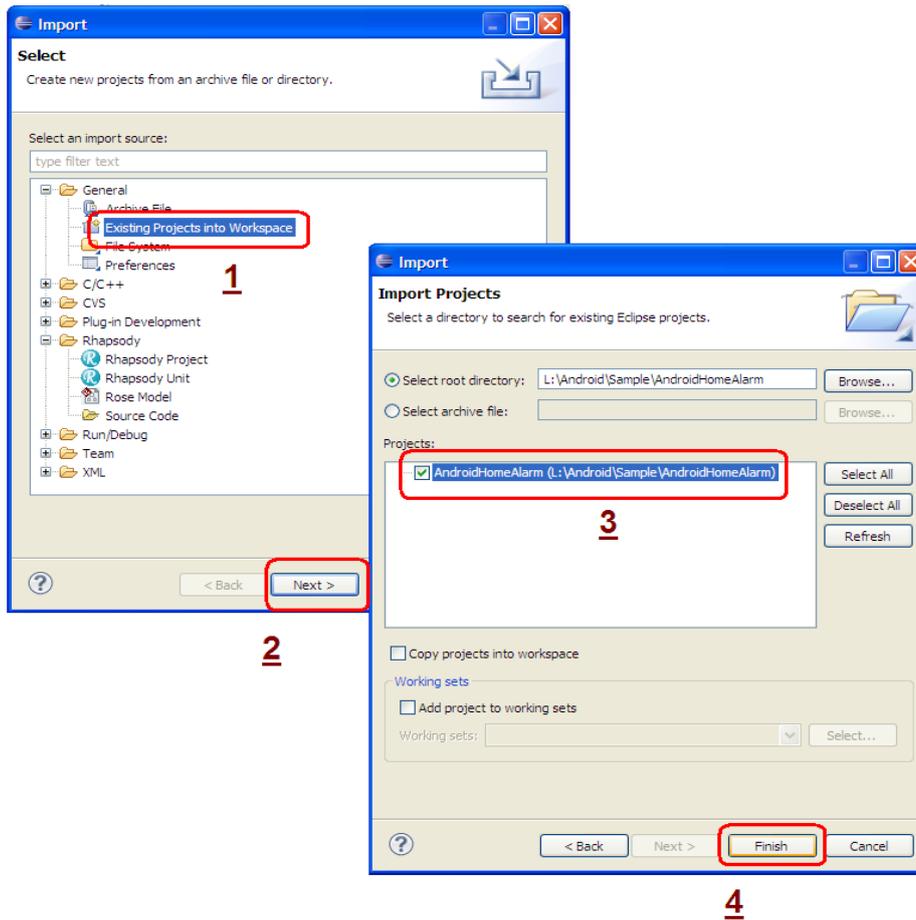
- 1) Open Eclipse



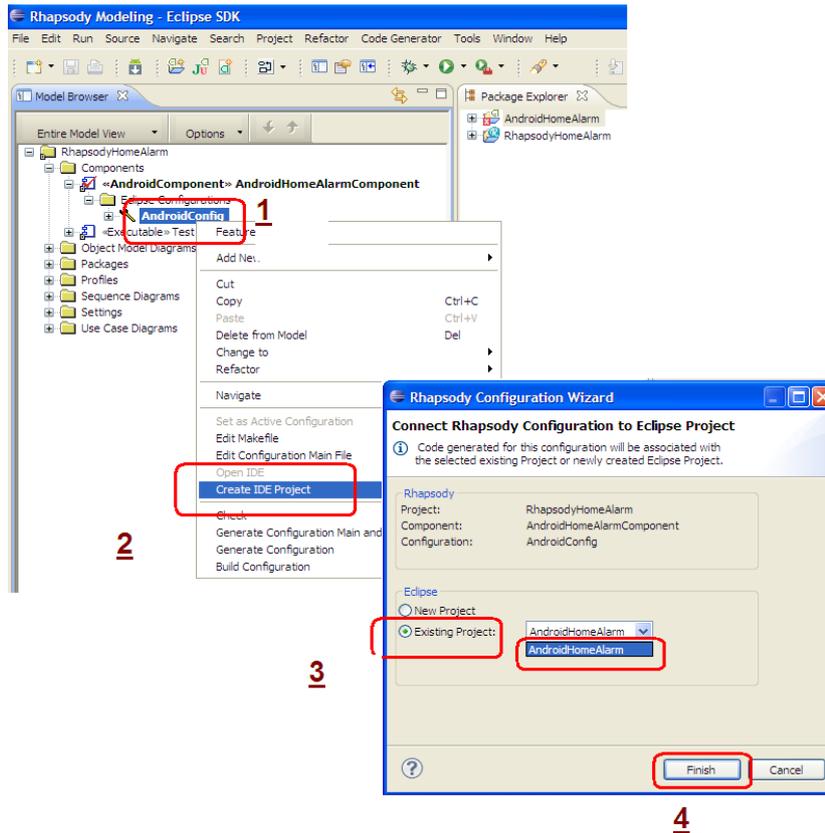
- 2) Import RhapsodyHomeAlarm



3) Import AndroidHomeAlarm



4) Create IDE project -> Existing project



5.11 Generación de código a partir de un modelo de Rational Rhapsody

Rational Rhapsody puede generar código de implementación de su modelo UML para:

- Toda la configuración
- Varios componentes
- Todo el proyecto
- Ciertas clases

Las entradas al generador de código son el modelo y las propiedades de generación de código. Las salidas del generador de código son archivos de código fuente en el lenguaje de destino: archivos de especificación, archivos de ejecución y los archivos make (ficheros de texto que utiliza make para llevar la gestión de la compilación de programas).

Antes de generar código, debe establecer la configuración activa. El generador de código ejecuta automáticamente el chequeador para comprobar si hay inconsistencias que podrían causar problemas en la generación o compilación del código. Seleccione **Code > Generate** (configuración activa), o presione **Ctrl + F7**.

5.11.1 Sincronización entre modelos y código

Rational Rhapsody le permite trabajar en el modelo o código y mantener la sincronización entre cada uno para que los cambios en una se reflejan en la otra de forma automática. Esto se conoce como Asociatividad dinámica entre Modelo-Código (DMCA).

Para seleccionar la opción de DMCA que desee utilizar:

1. Elija **Code Generator > Dynamic Model Code Associativity**.
2. En el submenú, seleccione uno de los siguientes comandos:

Además si se está generando código para un componente grande, le proporciona la ventaja de poder continuar trabajando en el Rational Rhapsody GUI mientras se genera código.

Propiedades utilizadas para la generación de código paralelo
CG::General::ParallelCodeGeneration.

5.11.5 Generación de código en proyectos multilingües

Cuando se ejecuta IBM Rational Rhapsody, se selecciona una versión específica del lenguaje. Esta selección determina el lenguaje que se asocia con sus proyectos. Sin embargo, los proyectos abiertos de Rational Rhapsody fueron creados con otros lenguajes del producto.

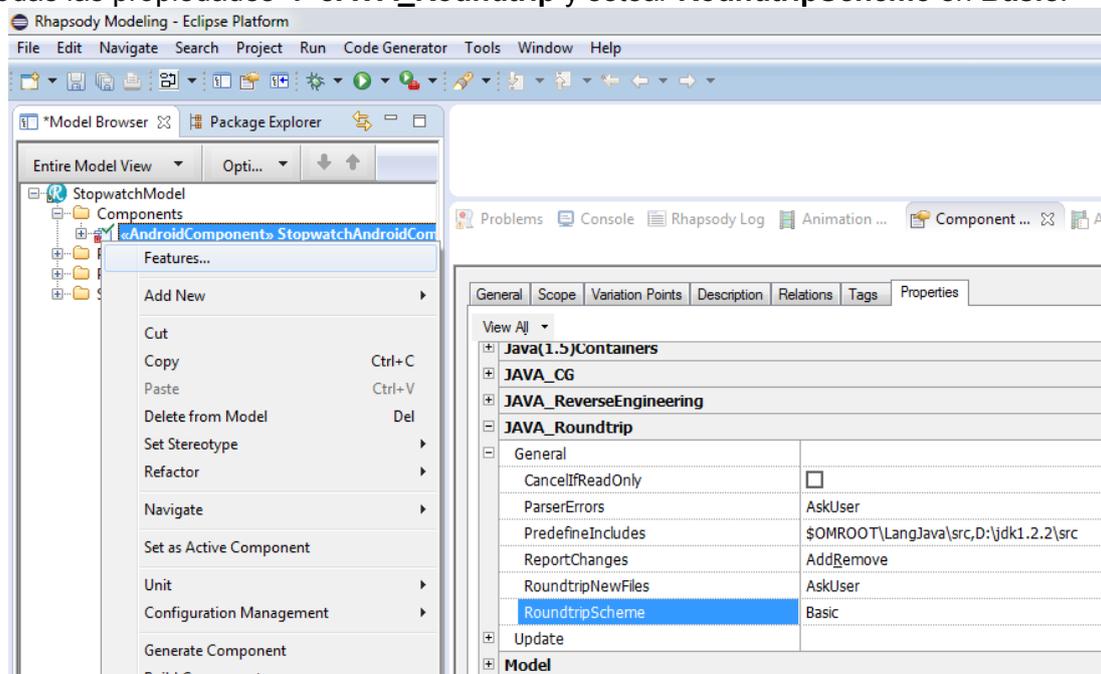
Además, el producto permite un único modelo para contener unidades que se asocien a diferentes lenguajes. El modelo puede incluir unidades asociadas a C, C++ o Java. El código puede entonces ser generado en el lenguaje apropiado para cada unidad. Para que la generación de código funcione correctamente, se tiene que adherir a las siguientes normas:

- Para generar el código para ciertas unidades en un lenguaje, el lenguaje apropiado se debe especificar en el nivel de componente.
- Los elementos incluidos en el alcance del componente deben ser del mismo lenguaje para el componente. Si otros elementos del lenguaje están incluidas en el ámbito de aplicación, una advertencia será emitida durante la generación de código.

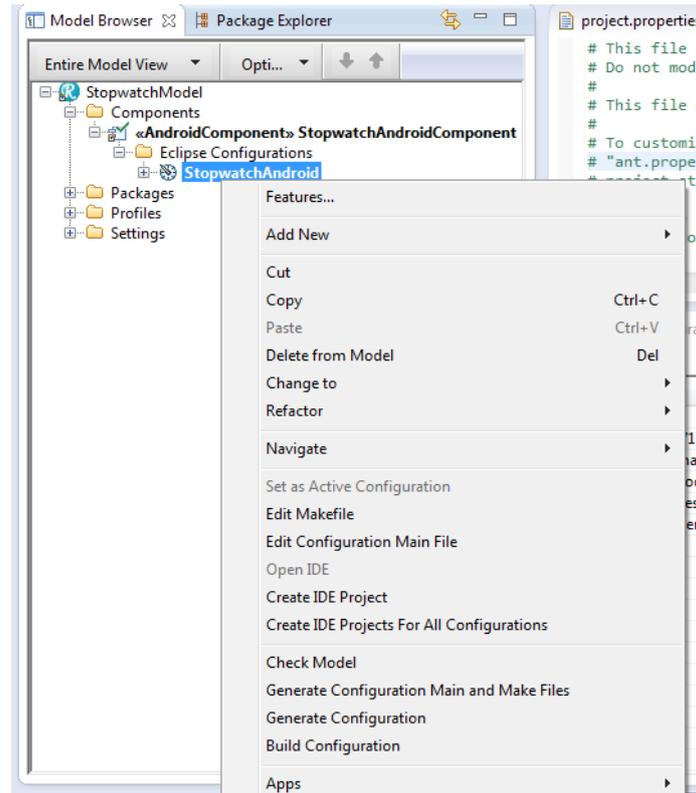
5.12 Generar y compilar el código de una Aplicación Rational Rhapsody integrada a Eclipse

1. **Nota:** Ante todo si tienes Windows de 64 bit, se recomienda setear la propiedad RoundtripScheme en Basic, para trabajar adecuadamente.

En el Model Browser seleccionamos el componente y dándole click derecho vamos a la opción **Features...** → solapa **Properties** → seleccionamos **View All** para visualizar todas las propiedades → **JAVA_Roundtrip** y setear **RoundtripScheme** en **Basic**.

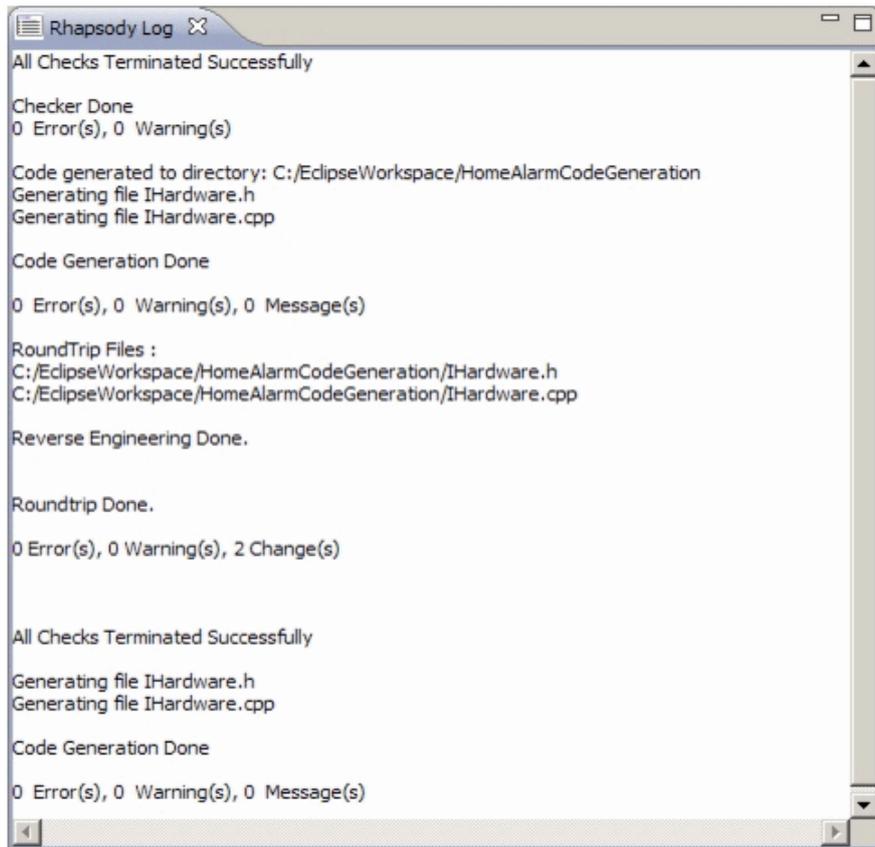


2. Para generar el código, compilar y luego ejecutarlo se debe indicar un componente activo sobre el que se trabajará. Para realizarlo se debe seleccionar un componente, darle clic derecho y seleccionar la opción **Set as Active Component**.



3. Acto seguido de seleccionar la configuración activa, se podría chequear el modelo “**Check Model**” para obtener errores y advertencias sobre el mismo.
4. Luego se debe generar los archivos de configuración a ser compilados y enlazados con sus dependencias y sus respectivos comandos a través de la opción “**Generate Configuration Main and Make Files**”.
5. Para generar el código, se realiza click derecho sobre el componente y se ejecuta la opción **Generate Configuration** o bien desde el menú principal se selecciona **Code Generator** → **Generate** → **<componente activo>**.

El sistema genera el código solicitado y muestra los mensajes en un archivo de log.



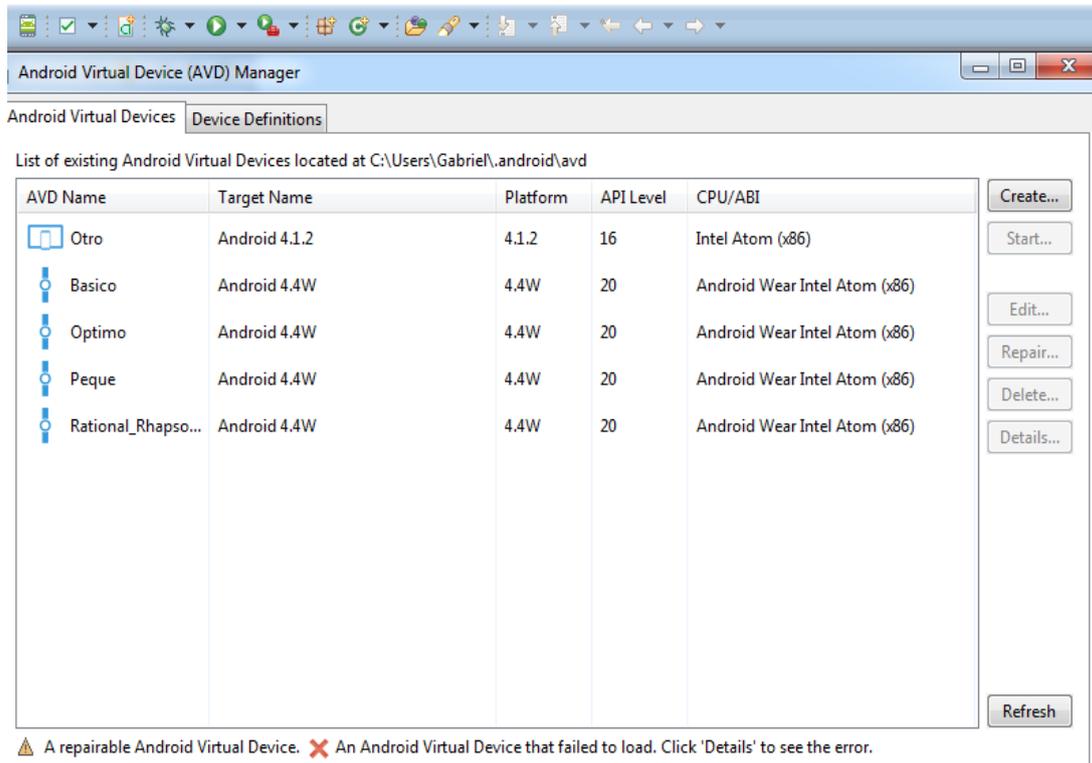
6. Para compilar el código generado se realiza click derecho sobre el componente y se ejecuta la opción **Build Configuration**.

5.12.1 Ejecutar y testear una aplicación de Android en Eclipse

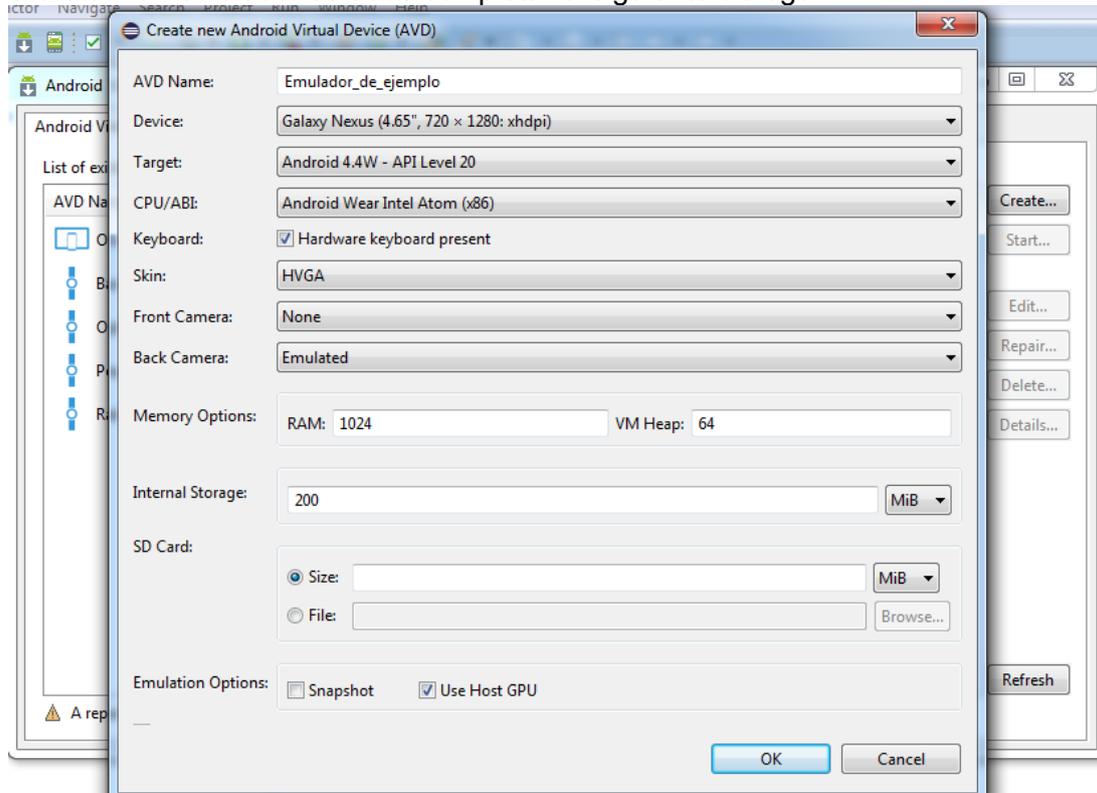
Cuando se desarrolla en Android, el SDK provee una lista de herramientas entre las que encontramos el emulador dónde podemos correr nuestros demos en un entorno que simula el de un teléfono real. Obviamente nos enfrentamos a sus limitaciones, ya que para desarrollar en ambientes móviles es vital contar con un equipo con el OS en el que estés trabajando ya que eso nos permite tener una visión clara de cómo funciona en verdad nuestra aplicación.

5.12.2 Configuración de un emulador

Vimos que al instalar el plugin ADT, se vinculan Eclipse con el SDK de Android para poder instalar herramientas (como Google USB Driver) y las diferentes versiones de Android en Eclipse. Ésto nos permite interactuar con dispositivos móviles y crear una gran variedad de emuladores con diferentes características a través del AVD Manager como se muestra a continuación.



Con esta herramienta se permite administrar las características de los emuladores. Para crear un emulador se debe completar el siguiente diálogo.



5.12.3 Configuración de un dispositivo móvil

Se configurará un teléfono Android a modo que cuando se ejecute un ejemplo en Eclipse se vea corriendo directamente en el teléfono que deberá estar conectado vía USB.

Si estás trabajando con **Eclipse**, estos son los pasos que se deben seguir:

1. Trabajando sobre Windows, será necesario instalar los drivers específicos del fabricante. Si contamos con un *Android Developer Phone*, ya sea Nexus One, o Nexus S, deberás usar el Google USB Driver.

5.12.4 Google USB Driver

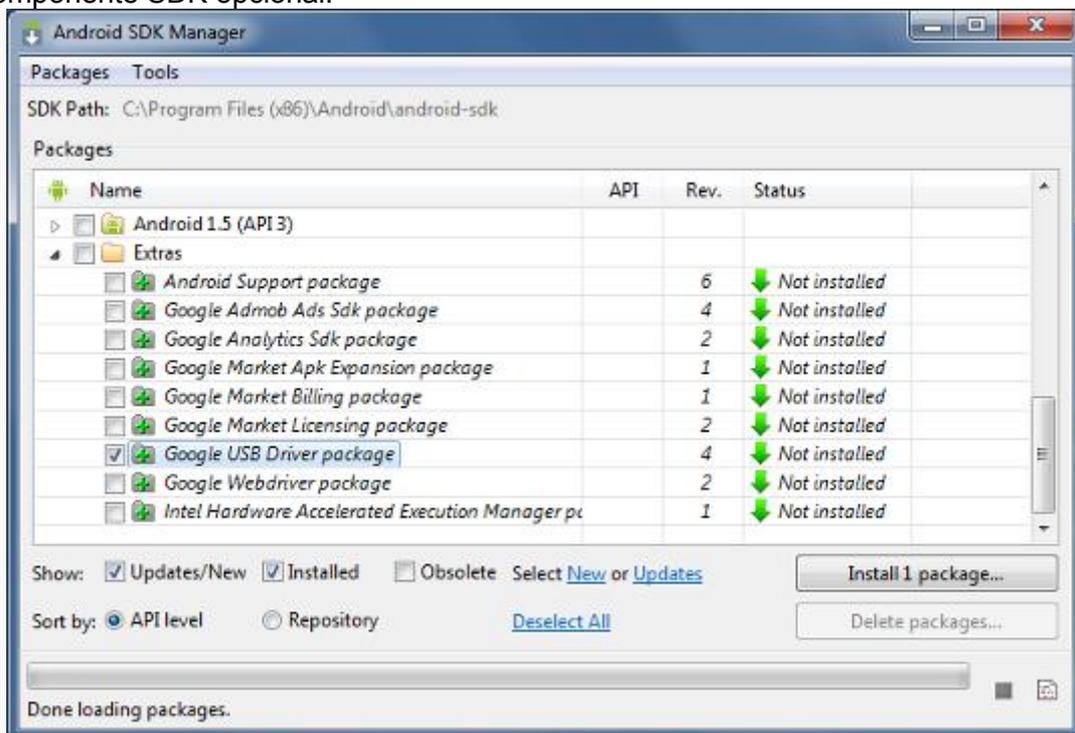
El Google USB Driver se **requiere sólo para Windows** con el fin de realizar depuración con cualquiera de los **dispositivos Nexus de Google**. La única excepción es el Galaxy Nexus: el controlador para Galaxy Nexus es distribuido por Samsung (que aparece como modelo SCH-I515).

Los controladores de Windows para todos los demás dispositivos son proporcionados por el fabricante de hardware respectivo.

Nota: Si usted está desarrollando en Mac OS X o Linux, entonces usted no necesita instalar un controlador USB.

5.12.5 Descarga del Google USB Driver

El controlador USB Google para Windows está disponible para su descarga como un componente SDK opcional.

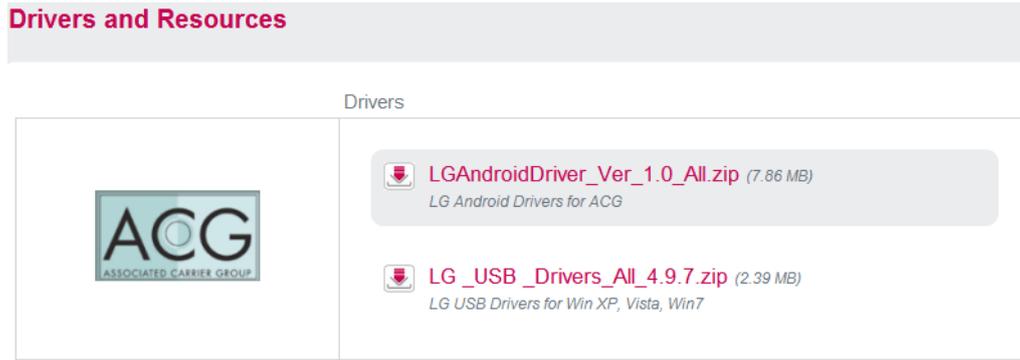


Inicie el Administrador de Android SDK haciendo doble clic en SDK Manager.exe, en la raíz de tu directorio SDK.

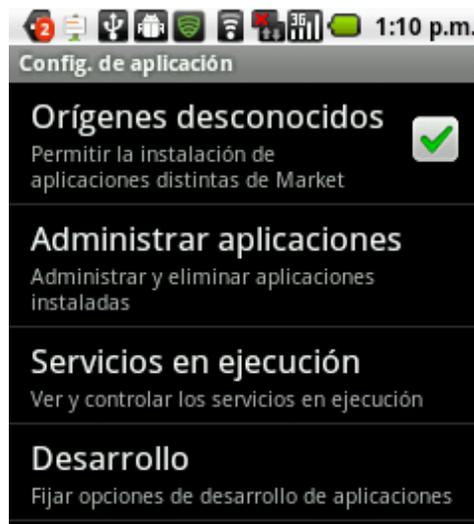
- 1) Expandir *Extras*.
- 2) Revise **el paquete del controlador USB Google** y haga clic en **Instalar**.
- 3) Proceda a instalar el paquete. Al finalizar, los archivos del controlador se descargan en el directorio <sdk>\extras\google\USB_Driver\.

En caso de tener un teléfono Android de otra marca deberemos instalar el driver OEM específico del fabricante.

En este caso específico, cuento con un teléfono LG, por lo que tuve que descargarme el driver seleccionando el archivo que te muestro a continuación:



2. Descomprimos el archivo y obtendremos un .exe que ejecutaremos. Si todo marcha bien, cuando el proceso finalice un cuadro de diálogo nos avisará que el driver del teléfono se ha instalado correctamente.
3. En nuestro teléfono, seleccionamos la opción *Ajustes > Aplicaciones* y habilitamos la opción de *Orígenes desconocidos*. Ésto nos permitirá instalar aplicaciones que no provengan del Android Market, que es el caso de las aplicaciones que estamos creando.



4. Conectamos nuestro dispositivo a la PC utilizando USB sin habilitar el modo de almacenamiento masivo. En lugar de ello nos vamos a la opción de *Ajustes > Aplicaciones > Ajustes* y habilitamos la opción *Depuración de USB*.



5.12.6 Uso del dispositivo de hardware

Se describirá cómo configurar el entorno de desarrollo y el dispositivo con Android para las pruebas y la depuración en el dispositivo.

Se puede utilizar cualquier dispositivo con Android como un entorno para ejecutar, depurar y probar sus aplicaciones. Las herramientas incluidas en el SDK hacen que sea fácil instalar y ejecutar la aplicación en el dispositivo cada vez que se compila.

Nota: Cuando se desarrolla en un dispositivo, tenga en cuenta que usted debe seguir utilizando el emulador de Android para probar la aplicación en configuraciones que no son equivalentes a las de su dispositivo real. Aunque el emulador no permite poner a prueba todas las funciones del dispositivo (como el acelerómetro), sí permite que usted verifique que la aplicación funciona correctamente en diferentes versiones de la plataforma Android, en diferentes tamaños de pantalla, orientaciones, y más.

5.12.7 Activación de opciones para desarrolladores en el dispositivo

Los dispositivos Android tienen una gran cantidad de opciones para desarrolladores que se puede acceder por teléfono, que le permiten:

- Habilitar la depuración a través de USB.
- Capturar rápidamente los informes de error en el dispositivo.
- Mostrar uso de la CPU en la pantalla.
- Dibujar la información de depuración en la pantalla tales como límites de diseño, cambios en las vistas de la GPU y capas de hardware, y otra información.
- Además de muchas más opciones para simular las tensiones de aplicaciones o activar las opciones de depuración.

Para acceder a estos ajustes, abra las *opciones de Desarrollador* en la configuración del sistema. En Android 4.2 y superior, la pantalla de opciones para desarrolladores está oculta de manera predeterminada. Para hacerlo visible, vaya a **Ajustes > Acerca del teléfono** y toque **Build number** siete veces. Regresar a la pantalla anterior para encontrar opciones del desarrollador en la parte inferior.

5.12.8 Configuración de un dispositivo para el desarrollo

Con un dispositivo con Android, se puede desarrollar y depurar aplicaciones Android al igual que lo haría en el emulador. Antes de empezar, sólo hay un par de cosas que hacer:

- Declare su aplicación como "depurable" en su Android Manifest.
Al utilizar Eclipse, puede saltarse este paso, ya que el funcionamiento de su aplicación directamente desde el IDE de Eclipse permite automáticamente la depuración.
En el archivo *AndroidManifest.xml*, agregue *android:debuggable="true"* elemento `<application>`.

Nota: Si habilita manualmente la depuración en el archivo de manifest, asegúrese de desactivarlo antes de construir para la liberación (la aplicación publicada normalmente debería *no* ser depurable).

- Habilitar **la depuración USB** del dispositivo.
En la mayoría de los dispositivos con Android 3.2 o mayor, usted puede encontrar la opción en **Ajustes > Aplicaciones > Desarrollo**.
En Android 4.0 y más reciente, está en **Ajustes > Opciones de Desarrollador**.
- Configure su sistema para detectar su dispositivo.
Si está desarrollando en Windows, es necesario instalar un controlador USB para Android Debug Bridge (adb).
Si está desarrollando en Mac OS X, simplemente funciona. Omita este paso.
Si está desarrollando en Ubuntu Linux, es necesario agregar un archivo de reglas *udev* que contiene una configuración USB para cada tipo de dispositivo que desea

utilizar para el desarrollo. En el archivo de reglas, cada fabricante del dispositivo se identifica mediante un ID de proveedor único, según lo especificado por la propiedad `ATTR {idVendor}`.

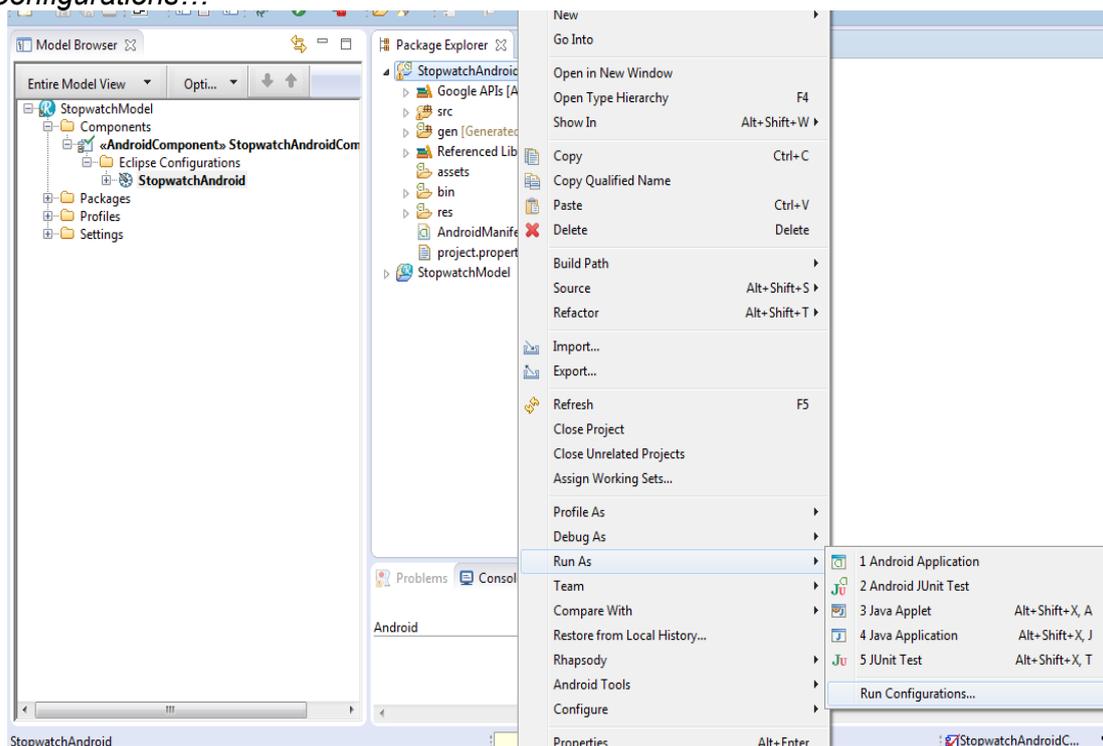
Nota: Cuando se conecta un dispositivo con Android 4.2.2 o superior para su equipo, el sistema muestra un cuadro de diálogo preguntando si acepta una clave RSA que permite la depuración a través de este equipo. Este mecanismo de seguridad protege a los dispositivos de usuario, ya que garantiza que la depuración USB y otros comandos adb no pueden ejecutarse a menos que seas capaz de desbloquear el dispositivo y reconocer el diálogo. Ésto requiere que usted tenga la versión 1.0.31 adb (disponible con SDK Platform-tools r16.0.1 y superior) con el fin de depurar en un dispositivo con Android 4.2.2 o superior.

Cuando se conecta a través de USB, se puede verificar que el dispositivo está conectado mediante la ejecución de `adb devices` disponible dentro de su directorio `platform-tools/` del SDK. Si está conectado, verá que el nombre del dispositivo aparece como un "dispositivo".

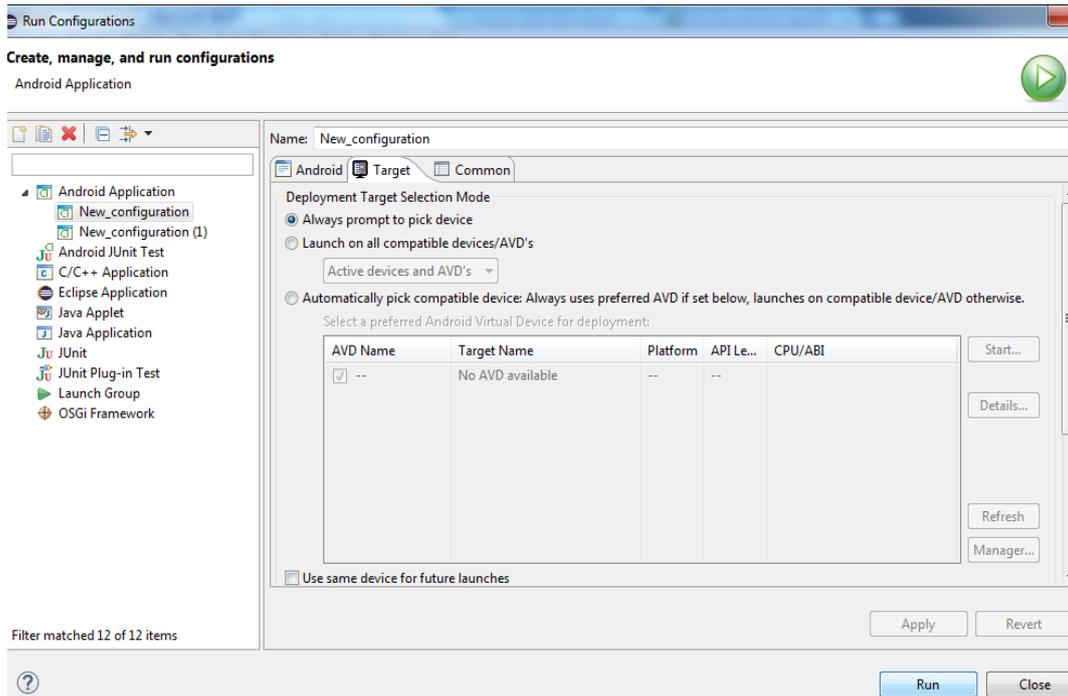
5. Ahora solo abrimos Eclipse y seleccionamos alguno de los proyectos que tengamos y lo ejecutamos como lo hacemos usualmente.

Con ésto ya podrás probar tus aplicaciones en un entorno real y detectar con mayor facilidad errores de funcionalidad y diseño.

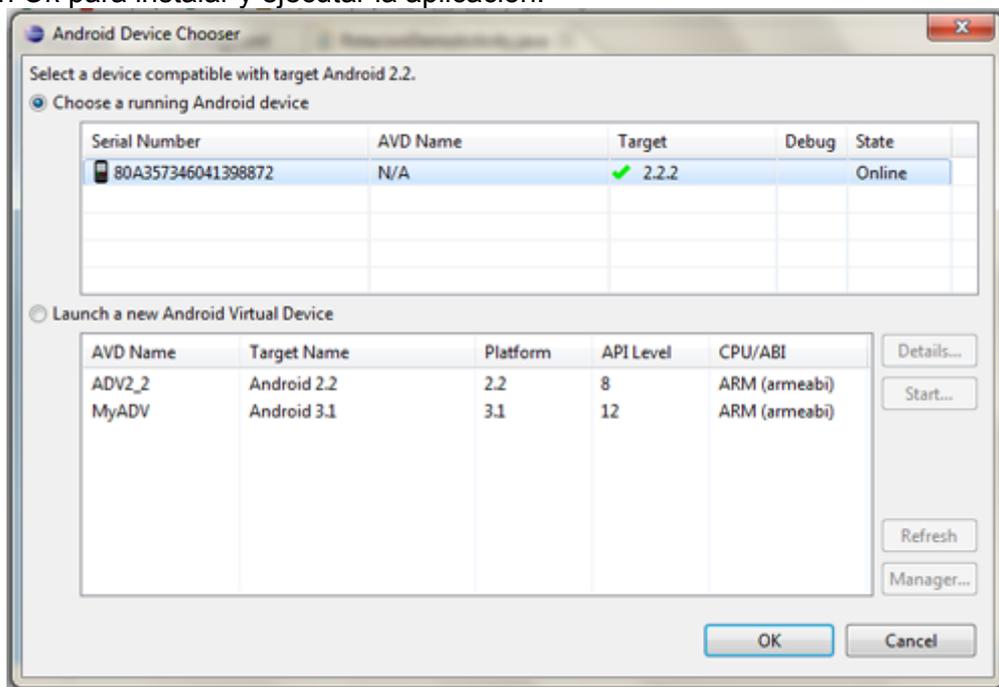
Le damos clic derecho al proyecto y seleccionamos la opción `Run as > Run Configurations...`



En la pantalla que te aparecerá, selecciona la pestaña `Target` en dónde verás el `RadioButton` que permite elegir en que dispositivo ejecutar la aplicación (emulador o dispositivo) y posteriormente damos clic sobre el botón `Run` como te muestro a continuación:



Con esto podemos acceder a la ventana *Android Device Chooser* en donde deberán aparecer los emuladores creados y nuestro teléfono con un estado *Online* que indicará que se encuentra disponible para utilizar. Lo seleccionamos y damos clic en el botón *Ok* para instalar y ejecutar la aplicación.



En mi caso, Eclipse detectó el teléfono y automáticamente instaló el demo entre las aplicaciones de él, para posteriormente poder ejecutarlo. En la siguiente figura se puede observa a la aplicación ejecutándose sobre el teléfono.



5.13 Animación de aplicaciones Rational Rhapsody en Eclipse

Las características de animación ayudan con la depuración a nivel de diseño del modelo a medida que se crea, de modo que pueda eliminar defectos fácilmente. El soporte de animación le permite controlar el diseño como lo haría con un depurador tradicional, pero a un nivel superior de abstracción.

Puede inyectar eventos, invocar operaciones, recorrer el modelo, ver el estado activo resaltado en gráficos de estado y generar automáticamente diagramas de secuencia creados a partir de la ejecución del ejecutable.

Cuando se ejecuta aplicaciones animadas Rational Rhapsody en Eclipse se cambia a la perspectiva Rhapsody Rational de depuración. Se puede depurar una aplicación de animación utilizando la funcionalidad Rhapsody Rational Animation y herramientas de depuración de Eclipse.

5.13.1 Preparación para la animación

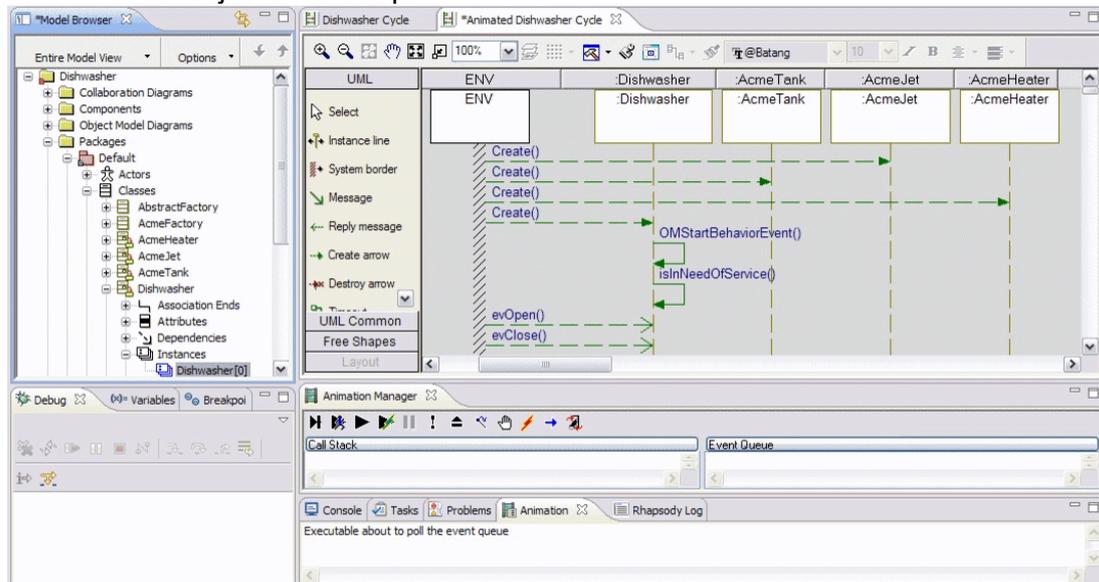
Procedimiento:

1. La animación utiliza el protocolo TCP/IP. Para habilitar el uso de este protocolo con la aplicación Android, abra el archivo `AndroidManifest.xml` y en la solapa **Permissions** agregar un **Uses Permissions** para `android.permission.INTERNET`.
2. En el model browser para el proyecto, expanda la carpeta **Components** y compruebe que en la configuración de Rational Rhapsody, el **Instrumentation Mode** (en la solapa **Settings** del cuadro de diálogo **Features**) se ha establecido en **Animation**. Si no es así, cambie el ajuste a **Animation**.
3. Haga clic derecho en la carpeta de configuración que desea animar y seleccione **Generate configuration** para que se regenere el código.

5.13.2 Ejecución de animación

Eclipse muestra la ventana **Animation Manager** con Call Stack y Event Queue y una barra de herramientas de animación (en la parte superior).

Para ejecutar la aplicación Android con el simulador de Android, haga clic en los botones de la barra de herramientas de animación. Mire la animación, y tenga en cuenta los mensajes en la solapa Animation.



5.13.3 La validación de un sistema

Rational Rhapsody permite visualizar el modelo a través de la simulación.

La simulación es la ejecución de los comportamientos y las definiciones asociadas en el modelo. Una vez que se simula el modelo, puede abrir diagramas simulados que le permiten observar el modelo, ya que está en marcha y llevar a cabo la depuración a nivel de diseño.

5.13.4 Verificación del modelo

Rational Rhapsody comprueba el modelo para los problemas antes de generar código. Para esto se puede controlar que chequeos deben ser llevados a cabo, sin embargo, algunos controles no se pueden desactivar. Además de los controles predefinidos que Rational Rhapsody ofrece, puede utilizar la API de Rhapsody para crear controles de modelo personalizado.

Los problemas encontrados durante la comprobación de modelo se encuentran en la solapa Check Model de la ventana Output. Si se encuentran errores en el modelo, no se genera código. Las advertencias no detienen la generación del código.

6 Acceleo

6.1.1 Definición

Acceleo es una herramienta de código abierto de la fundación Eclipse que permite generar código fuente a partir de un modelo determinado y de esta manera crear aplicaciones.

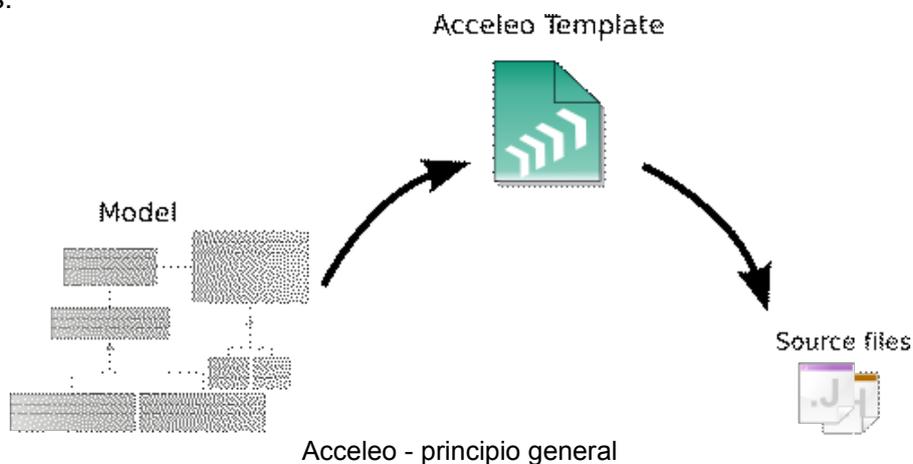
Se trata de una aplicación del estándar MOFM2T, definido por el Object Management Group (OMG), para llevar a cabo la transformación de modelo a texto (M2T).

Acceleo se puede utilizar para generar código de cualquier metamodelo, ya sea Ecore, UML o un DSL de encargo, con tal de que se pueda leer a través de EMF (Framework de Modelado en Eclipse), para generar cualquier tipo de código (Java, C, PHP...) mientras se mantiene la trazabilidad del texto generado.

Acceleo es el resultado de varios años-hombre de investigación y desarrollo que se inició en la empresa francesa Obeo en 2005. Es un primer paso en la prestación de las herramientas MDA, lo que permite una implementación pragmática y operativa de esta refactorización del ciclo de vida del software. Además es muy eficaz para proyectos a gran escala.

6.1.2 Fundamento

El principio de Acceleo es básicamente utilizar plantillas para generar el texto de un modelo abstracto. Entonces, el crear un nuevo módulo generador se reduce a crear plantillas.



6.2 Historia y desarrollo

El proyecto Acceleo nace en 2006 con el sitio Web Acceleo.org.

La primera versión pública fue Acceleo 0.8, una aplicación del estándar MOF, lenguaje de transformación de modelo a texto de la OMG. Acceleo 1.0 fue licenciado bajo la Licencia Pública GNU (GPL) y era compatible con Eclipse 3.0, 3.1 y varios modeladores basados en EMF y UML 1.2. Pocas semanas más tarde, Acceleo 1.1 trajo la compatibilidad con Eclipse 3.2 y UML 1.3. Acceleo cambió su licencia con la liberación de Acceleo 1.2 al adoptar la Licencia Pública Eclipse (EPL) utilizado por los proyectos de la fundación Eclipse. El proyecto Acceleo creció de manera constante desde su primera versión estable hasta sus versiones 2.x.

Acceleo 2 estaba disponible el 5 de junio de 2007 tras el lanzamiento de la página Web planet.acceleo.org que incluye artículos creados por los miembros de la

comunidad Acceleo y módulos de repositorio Acceleo que contiene generadores de código basados en Acceleo 2.

6.2.1 La entrada en la fundación Eclipse

En 2009, mientras se mueve a Acceleo 3, el proyecto ha sido aceptado en la fundación Eclipse. Durante esta transición, el lenguaje utilizado por Acceleo para definir un generador de código se ha cambiado para utilizar el nuevo estándar de la OMG para la transformación de modelo a texto, MOFM2T. Este lenguaje de generación de código utiliza un enfoque basado en la plantilla. Con este enfoque, una plantilla es un texto que contiene parte dedicada donde el texto se calcula a partir de los elementos proporcionados por los modelos de entrada. Dentro de Acceleo, esas expresiones se basan en el lenguaje OCL.

Acceleo se construye en la parte superior de varias tecnologías clave, como Eclipse EMF y, desde el lanzamiento de Acceleo 3, la aplicación Eclipse de OCL.

Con el lanzamiento de Eclipse 3.6 Helios, Acceleo está incluido en el tren de lanzamientos de Eclipse. Acceleo es un plugin de Eclipse y, como tal, está integrado en el IDE de Eclipse.

No hay muchas diferencias entre la versión antigua y la nueva. Este proyecto forma parte de Marte, Luna, Kepler, Juno, Indigo, Helios y Galileo.

El Equipo Acceleo seguirá manteniendo la vieja sintaxis del Acceleo fuera de Eclipse (www.acceleo.org) durante un par de años, pero las nuevas versiones y las nuevas funciones tendrán lugar en el Eclipse.org. Existe una herramienta automatizada que le ayuda a migrar sus plantillas desde una sintaxis a otra.

En Eclipse Acceleo se encuentra todo lo que han querido en la versión Acceleo.org y más.

Versión	Fecha de lanzamiento	Registro de estreno
1.0	01 de abril 2006	Primera versión de Acceleo disponible bajo la licencia GPL, la generación de códigos basados en modelos EMF.
1.1	20 de octubre 2006	Compatibilidad con Eclipse 3.2, el soporte para los modelos creados con GMF, ArgoUML, Poseidon, Umbrello y Rational Rose.
1.2	05 de enero 2007	Cambio a la Licencia Pública Eclipse (EPL), el nuevo servicio de indentación, lanzador parametrizado, compatibilidad Ant.
2.0	05 de junio 2007	Mejoras del lenguaje, compatibilidad con XML, exportación de generadores como Eclipse plug-in.
2.1	12 de julio 2007	Compatibilidad con Eclipse 3.3, generador de código depurador.
2.2	07 de enero 2008	Soporte de localización, el apoyo a diferente codificación de plantillas, la liberación de los siguientes generadores. JEE, PHP, Python y WISS.
2.3	25 de julio 2008	Compatibilidad con Eclipse 3.4, capacidad de llamar EOperations estándar definidos en el modelo.
2.4	25 de septiembre 2008	Posibilidad de exportar un generador como un módulo independiente o como una aplicación de RCP.
2.5	03 de diciembre 2008	Generación de perfiles, acciones en el esquema.

2.6	26 de junio 2009	Compatibilidad con Eclipse 3.5 Galileo.
2.7	06 de abril 2010	Mejora del perfilador, compatibilidad con Enterprise Architect.
3.0	23 de junio 2010	Un nuevo lenguaje para definir generadores de código basados en MOFM2T, el apoyo a consultas basadas OCL, la compilación en tiempo real con la detección de errores.
3.1	23 de junio 2011	Apoyo a la documentación de los generadores, la detección de posibles errores con la advertencia, el apoyo Maven, compilación binaria del generador.
3.2	27 de octubre 2011	Siguiente versión principal de Acceleo con la nueva vista "intérprete" para evaluar expresiones Acceleo de un determinado conjunto de elementos del modelo.
3.3	27 de junio 2012	El lanzamiento de Juno Acceleo introduce algunas nuevas APIs.
3.4	27 de junio 2013	La liberación de Kepler Acceleo introduce algunas nuevas APIs.
3.5	25 de junio 2014	Es parte de Eclipse Luna. Centrada en el mantenimiento se eliminan componentes obsoletos y cambia a Java 6.
3.6	24 de junio 2015	Es parte de Eclipse Mars. Centrada en el mantenimiento, la estabilidad, la mejora del rendimiento y en la documentación.
3.6.4	22 de junio 2016	Es parte de la serie de actualizaciones de Eclipse Neón. Es una versión de mantenimiento centrado en mejorar el rendimiento, la estabilidad y la estructura del proyecto.

6.2.2 El código fuente

Acceleo es un proyecto de Eclipse mayormente desarrollado en Java y está disponible bajo la Licencia Pública Eclipse (EPL). El código fuente de Acceleo 1.x y 2.x es disponible en SVN en la página Web del Consorcio OW2. Con su entrada en la fundación Eclipse en 2009, el código fuente de Acceleo 3 fue trasladado desde SVN a un CVS repositorio basado en los servidores de la fundación Eclipse. Después del lanzamiento de Eclipse 3.7 Indigo en 2011, el proyecto Acceleo ha emigrado a Git, después de la migración iniciado por varios proyecto oficial de la fundación Eclipse y desde julio de 2011, el código fuente Acceleo está disponible en Github.

6.2.3 Licencia

Acceleo se distribuye como software libre, los términos EPL significan que usted puede:

- utilizar Acceleo sin deberle las regalías.
- copiar y redistribuir Acceleo.
- modificar el código fuente Acceleo.
- crear sus propios módulos, libremente y redistribuirlos.

6.3 Compatibilidad

6.3.1 Los cambios de comportamiento entre las versiones

Tome en cuenta que a medida que las nuevas características OCL se habilitan en las nuevas versiones de Eclipse, son incompatibles (en tiempo de compilación) con antiguos Eclipses. En resumen, los módulos que se compilan en un Eclipse dado

siempre compilan en versiones posteriores, y los módulos que se han compilado (sea cual sea la versión) deben ser capaces de lanzarse en cualquier Eclipse. Se proporcionan servicios de migración. Específicamente, la migración automática de los proyectos de generación de Acceleo 2 para Acceleo 3.

6.3.2 Compatibilidad entre Eclipse y Acceleo

El equipo de desarrollo Acceleo hace todo lo posible para mantener la compatibilidad descendente hacia Ganímedes (Eclipse 3.4). A continuación se presenta la tabla de compatibilidad:

Acceleo	Eclipse Ganymede 3.4	Eclipse Galileo 3.5	Eclipse Helios 3.6	Eclipse Indigo 3.7	Eclipse Juno 3.8/4.5
3.0	OK	OK	OK	OK	OK
3.1	OK	OK	OK	OK	OK
3.2	OK	OK	OK	OK	OK
3.3	OK	OK	OK	OK	OK
3.4	OK	OK	OK	OK	OK
3.5	OK	OK	OK	OK	OK
3.6	OK	OK	OK	OK	OK

En verde, se puede encontrar la versión recomendada de Eclipse para cada versión de Acceleo.

6.3.3 Plataformas compatibles

Acceleo está escrito en Java y se despliega como un plugin en el IDE de Eclipse. Acceleo es compatible con entornos basados en Java 5+, en las siguientes plataformas:

- GNU / Linux.
- Mac OS X.
- Windows XP, Windows Vista, Windows 7.

Sistema Operativo	Acceleo 1.0	Acceleo 1.1	2.x Acceleo	3.x Acceleo
Windows	Sí	Sí	Sí	Sí
Gnu / Linux	No	Sí	Sí	Sí
Mac OS X	No	No	Sí	Sí

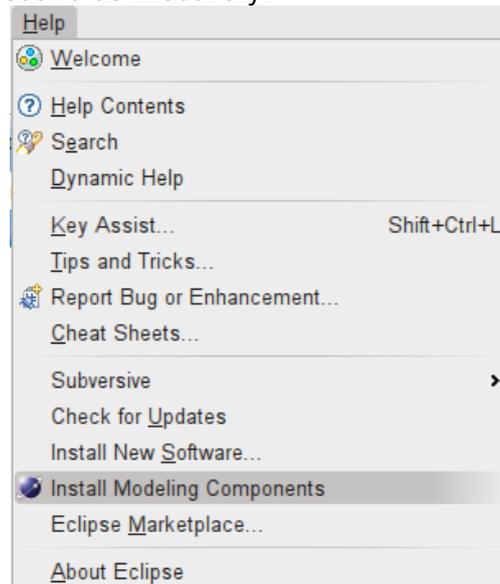
6.4 Instalación de Acceleo

Acceleo puede ser descargado e instalado de varias maneras.

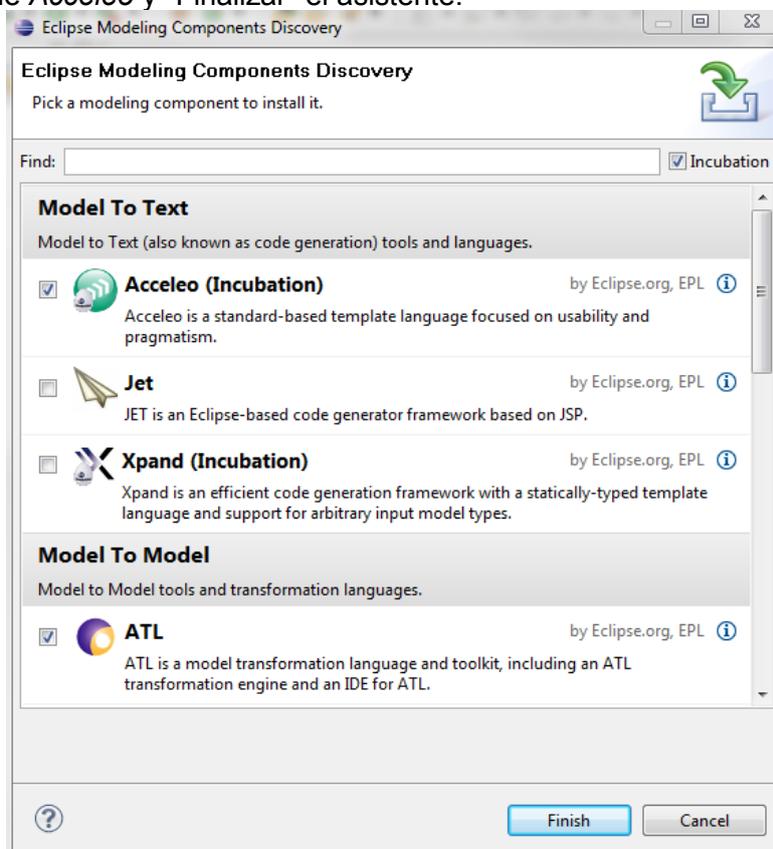
- Si se prefiere instalar un nuevo Eclipse con Acceleo, es posible que desee echar un vistazo a las instalaciones proporcionadas por el **proyecto de fusión**.
- Si se tiene una instalación de Eclipse existente y simplemente desea instalar Acceleo en ella, **la instalación a través del sitio de actualización es la forma más fácil**.

6.4.1 Proyecto de Fusión

1. Descargue el último paquete de modelado de Eclipse.
2. Abra la interfaz de usuario de Discovery.

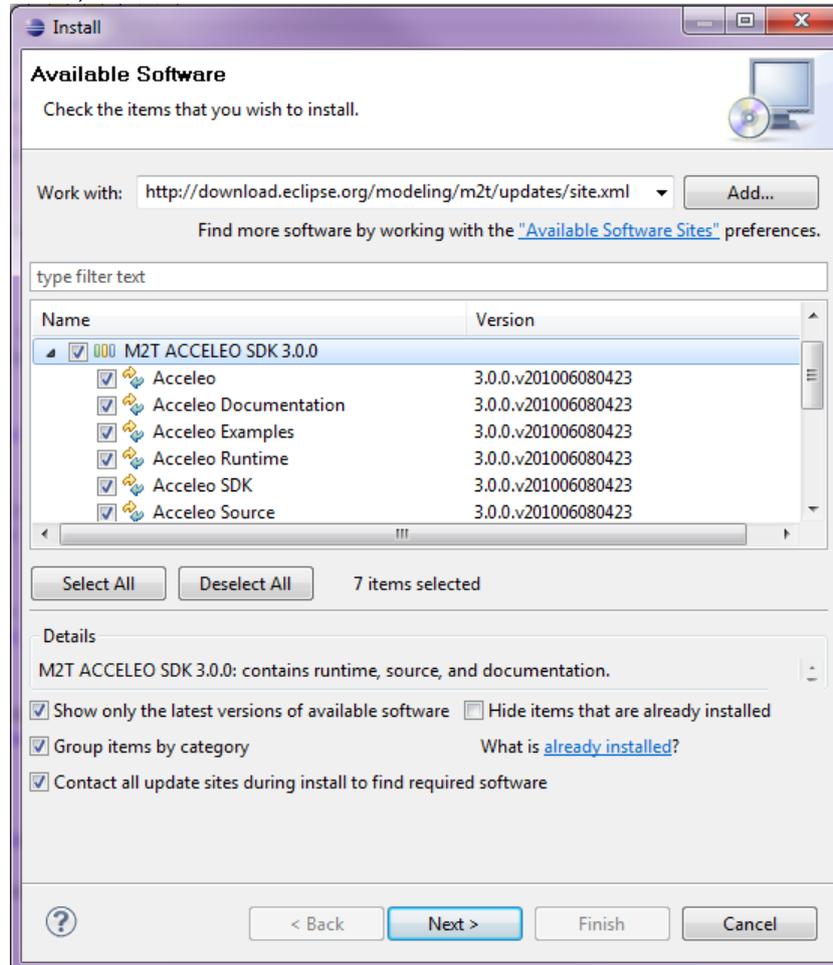


3. Seleccione *Acceleo* y "Finalizar" el asistente.



6.4.2 Sitio de actualización

1. Instale su distribución de Eclipse preferido de la página de descarga de Eclipse.
2. Ir a Help => Install New Software...
3. En la parte superior, haga clic en Add y escriba la ubicación del sitio de actualización (última versión:
<http://download.eclipse.org/modeling/m2t/acceleo/updates/releases>)
4. Seleccionar este nuevo sitio de actualización en el combo box y tilde *Acceleo* en el panel inferior, tal como se muestra a continuación.



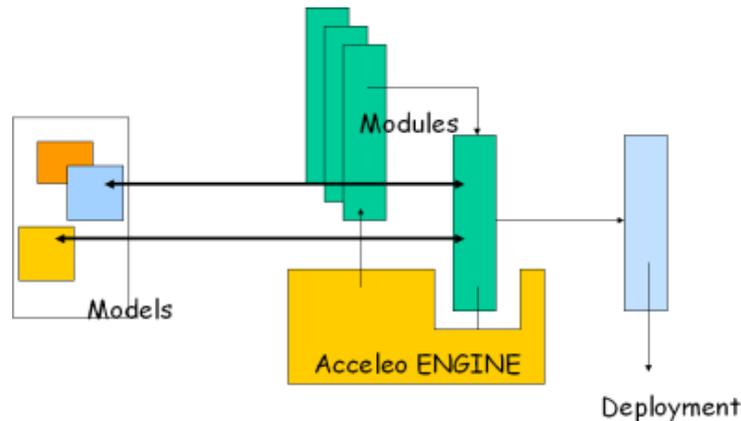
5. Haga clic en *Finalizar*, leer y aceptar la Licencia Pública de Eclipse, y ver la instalación de Acceleo en sí, junto con sus dependencias.
6. Luego descargar el SDK de Android e instalarlo.
7. Vincular con Eclipse a través del Android Development Tools (ADT) plugin en Eclipse, de la misma manera que se realizó para Rational Rhapsody.

6.5 El lenguaje Acceleo

El lenguaje Acceleo, se compone de dos tipos principales de estructuras (plantillas y consultas) en el interior de un módulo. En Acceleo, se pueden crear expresiones las cuales son usadas para seleccionar y extraer información del modelo. En Eclipse IDE dichas expresiones están basadas en la implementación del lenguaje OCL.

6.5.1 Módulos

Un generador Acceleo se compone de varios archivos llamados módulos. Los módulos son plugins Acceleo para la generación de código. Un módulo es un archivo .mtl, que se compone de varias plantillas y/o consultas.



Un módulo puede extender otro módulo, en cuyo caso sus plantillas podrán reemplazar sus plantillas "públicas" y "protegidas". También tendrá acceso a las plantillas y las consultas públicas y protegidas del módulo extendido. Un módulo también puede importar otro módulo para acceder a sus plantillas y consultas públicas.

6.5.2 Plantillas

Las plantillas son conjuntos de declaraciones Acceleo utilizados para describir la información necesaria para generar el código fuente a partir de un metamodelo. Ellos están delimitados por las etiquetas `[template...][template]`.

Una plantilla tiene una visibilidad (público | protegidos | privada) con un alcance similar a la visibilidad en lenguajes de programación orientados a objetos.

Dentro de una plantilla que puede utilizar dos tipos de expresiones para generar el código, primero, expresiones estáticas que se generarán sin transformaciones y expresiones Acceleo que utilizarán los elementos del modelo para calcular el texto generado.

```

generate.mtl
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/2.1.0/UML')]

[template public generate(c : Class)]

  [comment @main /]
  [file (c.name, false, 'UTF-8')]
  [c.name/]
  [/file]
[/template]

```

Las plantillas también pueden tener elementos opcionales:

- Anulaciones;
- Pre-condiciones (o condiciones de guarda);
- Post-Tratamiento;
- Inicializaciones de variables.

6.5.3 Anulaciones

Acceleo cuenta con dos sistemas diferentes con el fin de anular el comportamiento de una plantilla estática y una dinámica. Esos dos mecanismos están abordando diferentes tipos de problemas:

Anulado estático

Con el reemplazo estático, se puede fácilmente volver a utilizar el contenido de un módulo existente y cambiar un trozo de su comportamiento original.

El anulado estático existe en un lenguaje orientado a objetos como Java, pero aquí es diferente porque los módulos no son objetos, sino que son más similares a espacios de nombres. Para ver la diferencia, vamos a considerar los siguientes elementos:

```

--- Java ---

class A
methodA {}
method B {methodA()}

class B extends A
methodA overrides methodA {}
methodB overrides methodB {super.methodB}

```

En Java, llamando al methodB en un objeto B, estos son los métodos que serán llamados:

- B.methodB (punto de partida)
 - A.methodB (la llamada a super)
 - B.methodA (la llamada a MethodA en A.methodB se envía a la aplicación en B)
- Incluso si la clase A no conoce la clase B, la llamada del MethodA en a.methodB es enviado a la subclase B porque es nuestro contexto. Esa es la forma en que funciona la programación orientada a objetos.

Ahora vamos a considerar los módulos:

```

--- Acceleo ---

module A
templateA
templateB [templateA/]

module B extends A
templateA overrides templateA
templateB overrides templateB [super/]

module C
import module B

```

Si llama templateB desde el módulo C, éstas son las plantillas que se llamará:

- B.templateB (el punto de entrada)
- A.templateB (la llamada a super)
- A.templateA (la llamada no se distribuye a B.templateA, al contrario que en el "submódulo" para Java)

La palabra clave "extends" es sólo un modificador de la visibilidad de los elementos del módulo A en relación con el módulo B que en realidad no crean una relación entre un "módulo A" y un "módulo B", esos no son objetos, solo hay espacios de nombres.

Anulado dinámico

El reemplazo dinámico está en un nivel completamente diferente, ya que permite cambiar el comportamiento de un generador existente mediante la definición de las plantillas, que sustituirán a las plantillas existentes del generador. De esta manera, al igual que con la programación orientada a aspectos, puede parchear un generador existente ya construido, certificado y desplegado en su ordenador sin tener que cambiar el generador original.

Con el fin de hacer eso, sólo tienes que crear un nuevo proyecto Acceleo y utilizar el punto de extensión de la plantilla dinámica. El anulado dinámico se utiliza gracias al punto de extensión "org.eclipse.acceleo.engine.dynamic.templates". Si implementa el generador original, usted tendrá el comportamiento original pero si se implementa el nuevo generador con él, sus módulos dinámicos se "parchean" al generador original, cambiando dinámicamente su comportamiento. Por ejemplo, consideremos los módulos:

```
--- Acceleo ---  
  
module A  
  
templateA  
  
templateB  
  
  
  
module B  
  
imports moduleA  
  
  
  
-- The dynamic module in another plugin --  
  
module C extends moduleA  
  
templateA
```

Si usted está llamando en su generador existente del templateA desde el moduleB, la plantilla dinámica C.templateA será utilizado en su lugar. El generador existente no sabe sobre el nuevo generador y no tiene dependencias con este otro generador, pero desde que hemos instalado el nuevo generador como un plugin en la misma instancia de Eclipse como el generador existente y desde el nuevo generador utiliza el punto de extensión dinámico, todas sus plantillas que están anulando una plantilla desde el generador original se utilizarán, en lugar de las plantillas originales.

En resumen: Si bien el anulado estático consiste en referencias a las plantillas/consultas, el anulado dinámica anula las referencias a módulos completos en declaraciones de importación. Los dos métodos de anulado no son mutuamente excluyentes. Se podría hacer referencia a estos dos tipos de anulaciones como anulado de **espacio de nombres** versus al anulado **de importación**.

6.5.4 Pre-Condiciones

Imagínese que usted desea implementar un comportamiento diferente para una plantilla en función de determinadas precondiciones.

```

[comment Generates the java code for a class property that belongs to an association and is ordered /]
[template public genAssociation(p : Property) ? (owningAssociation <> null and isOrdered)]

[/template]

[comment Generates the java code for a class property that belongs to an association and is NOT ordered /]
[template public genAssociation(p : Property) ? (owningAssociation <> null and not isOrdered)]

[/template]

```

El ejemplo anterior muestra el ? (condición) que le dice a Acceleo que la plantilla sólo se debe ejecutar si la condición previa es satisfecha.

El orden de declaración de plantillas en un módulo es importante: Se ejecutará la primera plantilla para que la condición de guardia se evalúa como verdadera.

6.5.5 Post-Tratamiento

A menudo es útil, especialmente para el formato de código, aplicar ciertos tratamientos en el texto generado por una plantilla antes de que realmente escribirla en el archivo de salida.

```

[template public javaName (c : Class) post (trim()) ]
[name.toUpperFirst()]
[/template]

```

En el ejemplo anterior, sin el post-tratamiento `post (trim())`, la invocación a la plantilla escribiría el nombre seguido de un retorno de carro. Con el post-tratamiento, siempre que la plantilla se llama, se escribe el nombre esperado, sin un retorno de carro, que es probablemente lo que se necesita.

6.5.6 Consultas (Query)

Las consultas se utilizan para extraer información del modelo. Las consultas devuelven valores, o colecciones de valores. Utilizan OCL, encerrado en una etiqueta `[query... /]`.

```

[comment
  Renvoie les attributs public d'une classe.
/]
[query public getPublicAttributes(c : Class) : Set(Property) =
  c.attribute->select(visibility = VisibilityKind::public)
/]

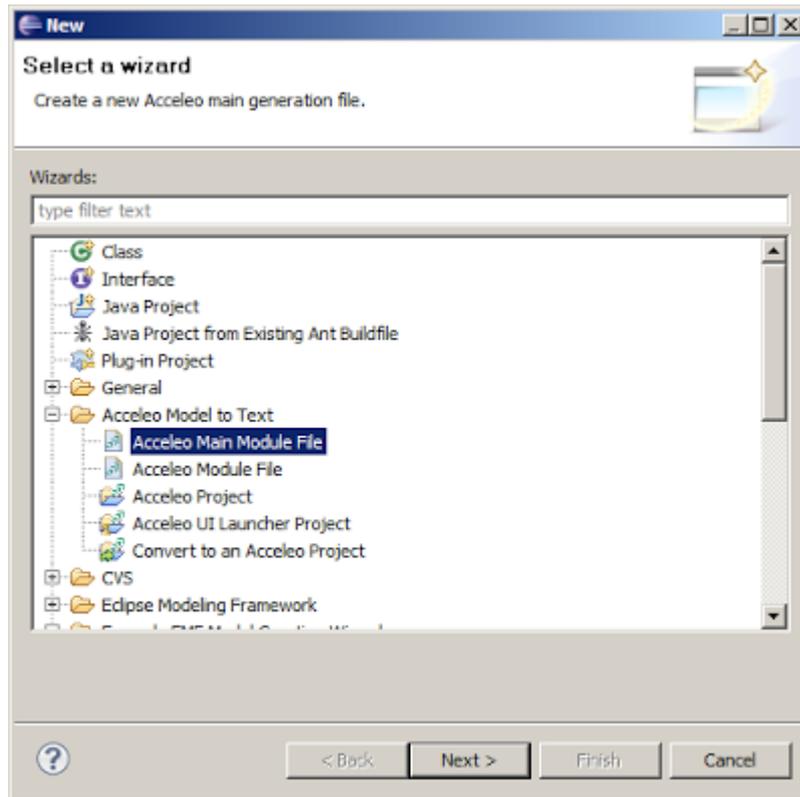
```

6.6 Elementos del lenguaje

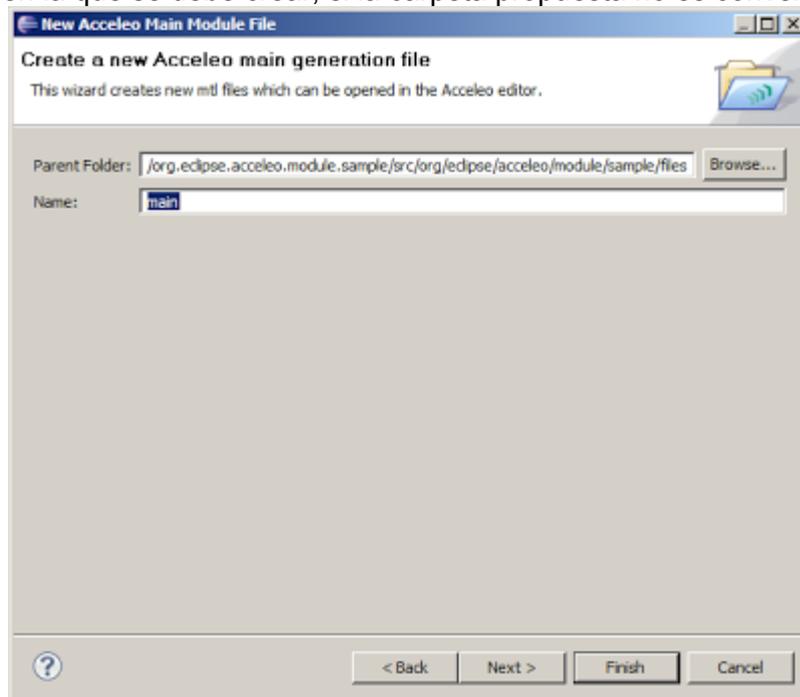
6.6.1 Módulos Main y clase Launcher

Los módulos "Main" de Acceleo son puntos de entrada, los cuales sólo tienen que contener la anotación "@main".

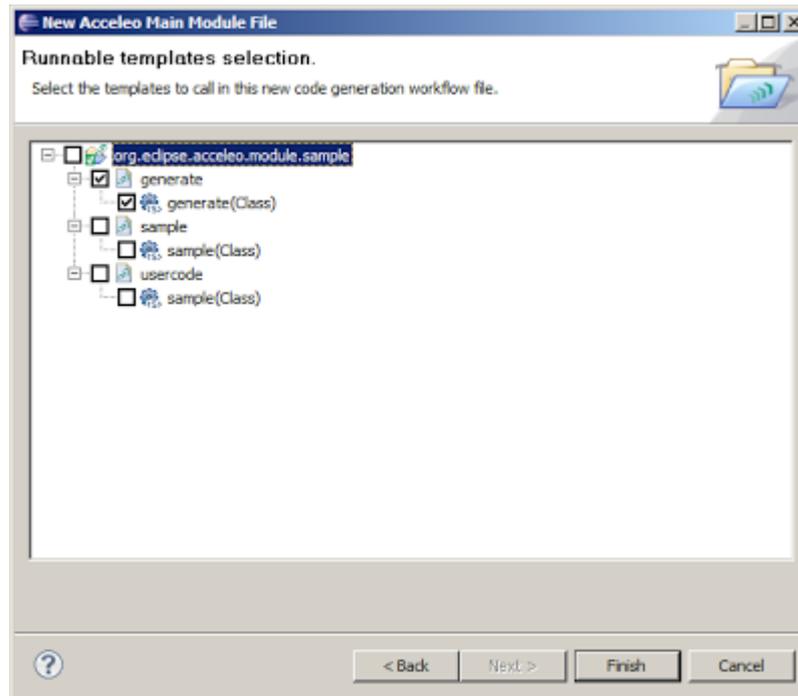
Para crear módulos Main seleccione `New > Other...` Seleccione `Acceleo Main Module File` en la categoría `Acceleo Model to Text`.



Haga clic en Next >. Introduzca el nombre del módulo para crear (sin la extensión .mtl) y la carpeta en la que se debe crear, si la carpeta propuesta no es conveniente.



Haga clic en Next >. Seleccione las plantillas que este módulo Main llamará para generar archivos.



Haga clic en Finish. Un nuevo módulo Acceleo se crea, que contiene la anotación Main y por consiguiente tiene un archivo Java generado adjunto, con el fin de iniciar la generación con esta plantilla. Este módulo importa los módulos que han sido seleccionados en la página del asistente anterior, y simplemente los llama uno tras otro.



Ahora usted puede lanzar este módulo principal simplemente haciendo clic derecho en el archivo main.mtl, seleccione RunAs > Launch Acceleo Application y seleccione el modelo de partida para generar y la carpeta de destino antes de golpear el botón "Run".

6.6.2 Generación de archivos

Las etiquetas de archivos se utilizan para indicar al motor Acceleo que debe generar el contenido de la etiqueta [file] en un archivo real.

La sintaxis es la siguiente:

[file (<uri_expression>, <append_mode>, <output_encoding>)] (...) [/ file]

- <uri_expression> expresión que devuelve un string que indica el nombre del archivo de salida;
- <append_mode> (opcional) booleano que indica si el texto de salida debe ser anexa al archivo o reemplazar su contenido;

- `<output_encoding>` (opcional) indica la codificación a utilizar para el archivo de salida. Esta codificación no necesita ser la misma que la codificación del módulo.

```
[module randomJavaFile('http://www.eclipse.org/emf/2002/Ecore')/]
```

```
[template public genRandomJavaFile(aParam: EPackage)]
```

```
[file ('myFile.java', false, 'UTF-8')]
```

```
Hello World
```

```
[/file]
```

```
[/template]
```

6.6.3 Estructuras de control

En las plantillas Acceleo se pueden usar expresiones Acceleo para manipular los elementos del modelo. Estas expresiones se basan en un súper conjunto del lenguaje OCL.

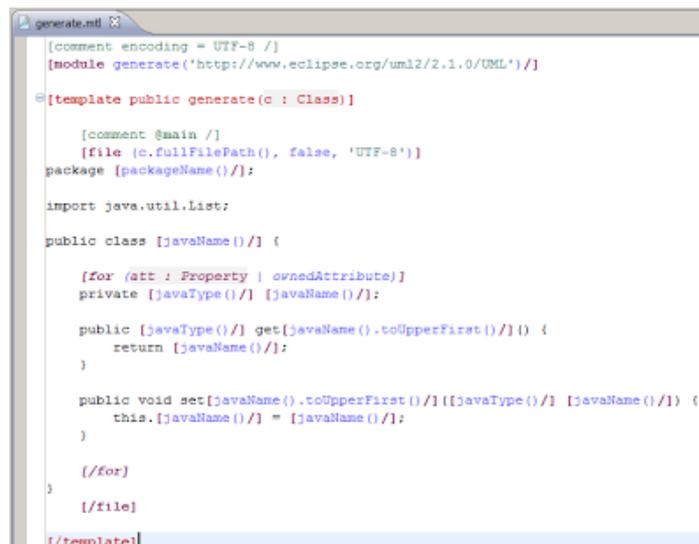
- **For Loops**

La sentencia For en Acceleo puede expresarse con dos sintaxis:

La sintaxis completa (conforme a la especificación MTL): `[for (i : E | e)]...[/for]`

La sintaxis resumida: `[for (<iterable _ expresión>)]...[/for]`

He aquí un ejemplo de un bucle for para generar algo de código para cada atributo de una clase UML:



```
generate.mtl
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/2.1.0/UML')/]
= [template public generate(c : Class)]
    [comment @main /]
    [file (c.fullFilePath(), false, 'UTF-8')]
package [packageName()]/:
import java.util.List:
public class [javaName()] / {
    [for (att : Property | ownedAttribute)]
    private [javaType()] [javaName()]/:
    public [javaType()]/ get([javaName().toUpperFirst()]/) () {
        return [javaName()]/:
    }
    public void set([javaName().toUpperFirst()]/) ([javaType()]/) [javaName()]/ {
        this.[javaName()]/ = [javaName()]/:
    }
    [/for]
}
[/file]
[/template]
```

- **Condición If**

La condición If se escribe así:

```
[if (condición)]...[/if]
```

- **Bloques Let**

Es importante entender que las variables en Acceleo son “final”, lo que significa que su valor no se puede cambiar después de que han sido inicializados.

La sintaxis es la siguiente:

```
[let (variableName : VariableType = expresión)]...[/let]
```

Dónde variableName es el nombre de la variable y variableType el tipo de la variable, y la expresión es una expresión cuyo valor será asignado a la variable si el tipo se corresponde (el bloque Let en Acceleo es equivalente a If (expression.oclsKindOf(VariableType))).

- **Comentario**

Los comentarios están inscritos en bloques `[comment]...[/comment]`.

6.6.4 Servicios Java

Los servicios Java deben utilizarse para consultar el modelo, filtrar uno o varios elementos del modelo y para implementar operaciones complejas. No se recomienda utilizar los servicios de Java para retornar gran cantidad de texto y generar código. El texto debe ser generado por las plantillas Aceleo.

Los servicios permiten una extensibilidad ilimitada, manteniendo las plantillas limpias y fáciles de leer.

Para utilizar un servicio, es necesario utilizar la operación Aceleo "invoke" a fin de decirle a Aceleo que llame a su método de Java y le dará el resultado. Los servicios Java se limitan a los parámetros y valor de retorno con el tipo de uno de los metamodelos utilizados en el generador o un "tipo primitivo" (String, Integer, Real, Boolean, etc.).

Usted puede llamar a los servicios de Java desde una plantilla o una consulta, sin embargo, se recomienda llamar desde consultas. He aquí un ejemplo de una llamada a un servicio de Java desde una consulta.

```
[module moduleName('http://www.eclipse.org/emf/2002/Ecore')]

[query reqMyQuery(aParam: EClass) : String =

    invoke('org.eclipse.acceleo.examples.uml2java.MyJavaClass',
'myMethod(org.eclipse.emf.ecore.EClass)', Sequence{aParam})

]
```

Se puede ver que vamos a llamar a la operación "myMethod" de la clase "MyJavaClass" con el argumento "aParam".

6.6.5 Archivos de propiedades

Los archivos de propiedades son archivos Java ".properties" que se pueden cargar con un generador Aceleo para personalizar una generación. Para utilizar un archivo de propiedades en su generador, hay que hacer referencia a él en el método "getProperties()" de su lanzador Java.

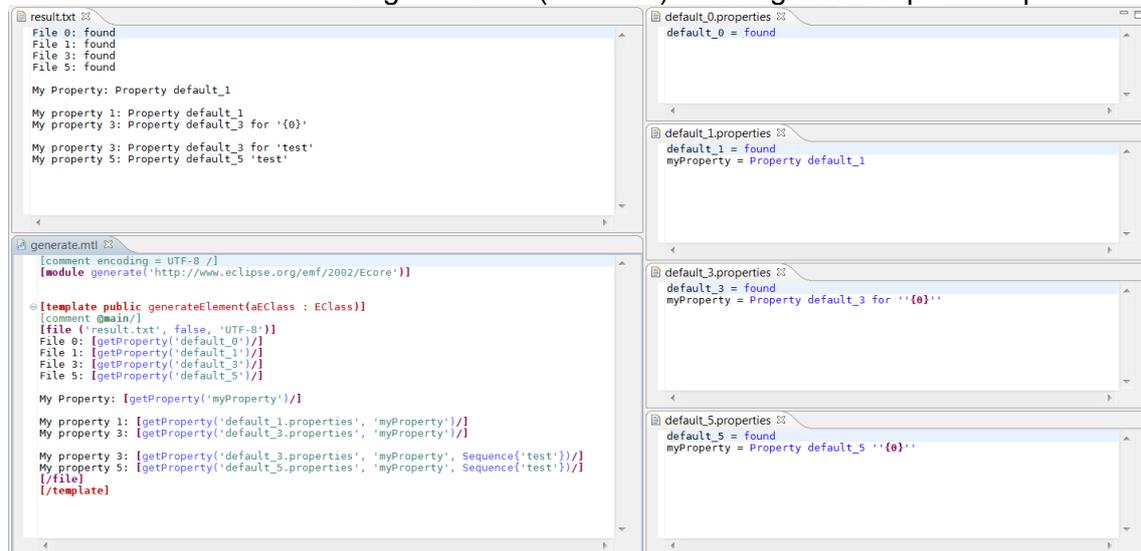
Property	Value
Info	
derived	false
editable	true

Una vez cargado un archivo de propiedades, es necesario utilizarlo en su generador. Con el fin de hacer eso, puede utilizar uno de las cuatro siguientes operaciones:

- "getProperty('myProperty')" devolverá el valor de la propiedad con la clave "myProperty". Archivos de varias propiedades pueden tener propiedades que compartan la misma clave, sólo el primer encontrado será devuelto con este método.
- "getProperty('default_1.properties', 'MyProperty')" devolverá el valor de la propiedad con la clave "myProperty" desde el archivo "default_1.properties".

- "getProperty('myProperty', Sequence{'my name'})" devolverá el valor de la propiedad con la clave "myProperty" parametrizado con el valor de la secuencia.
- "getProperty('default_3.properties', 'myProperty', Sequence {'my name'})" devolverá el valor de la propiedad con la clave "myProperty" parametrizado con el valor en la secuencia desde el archivo "default_3.properties".

Puede ver el resultado de la generación (result.txt) en la siguiente captura de pantalla.



6.7 Características

Acceleo proporciona herramientas para la generación de código basado en modelos compatibles con EMF definidos a partir de cualquier tipo de metamodelo como UML 1, UML 2 y hasta metamodelos personalizados (DSL). A partir de este metamodelo, el usuario puede definir un generador de código que va a producir cualquier tipo de lenguaje textual de un modelo usando el metamodelo.

	<pre> [comment encoding = UTF-8 /] [module generate('http://www.eclipse.org/emf/2002/Ecore')] [template public generateElement(aEClass : EClass)] [comment @main/] [file ('result.txt', false, 'UTF-8')] [file ('default_0', false, 'UTF-8')] File 0: [getProperty('default_0')] File 1: [getProperty('default_1')] File 3: [getProperty('default_3')] File 5: [getProperty('default_5')] My Property: [getProperty('myProperty')] My property 1: [getProperty('default_1.properties', 'myProperty')] My property 3: [getProperty('default_3.properties', 'myProperty')] My property 3: [getProperty('default_3.properties', 'myProperty', Sequence{'test'})] My property 5: [getProperty('default_5.properties', 'myProperty', Sequence{'test'})] [/file] [/template] </pre>	<pre> public class Cat { /** * The documentation of age */ private int age; /** * The documentation of size */ private int size; /** * The documentation of eat */ public void eat() { } /** * The documentation of sleep */ public void sleep() { } } </pre>
<p>Un modelo EMF sencillo</p>	<p>Un módulo Acceleo sencillo</p>	<p>El código generado</p>

Aquí estamos utilizando los siguientes elementos: EClass, EAttribute y EOperation de EMF. Gracias a este modelo de entrada y al módulo, Acceleo puede generar el código anterior. El módulo que se define en este ejemplo se parametriza para generar

Java pero el estándar MOFM2T es independiente del código generado, lo que habla de su versatilidad. Cuando se crea el generador, el usuario puede utilizar otro modelo para generar una pieza de código con una apariencia similar pero con un contenido diferente.

6.7.1 Dominio

La mayoría de los generadores de código o bien generan un tipo de tecnología (por ejemplo, Java) o usan un solo tipo de modelos (por ejemplo UML). Acceleo no le restringe a un dominio específico, permitiendo generar código a partir de:

- Modelos UML.
- Modelos Ecore.
- Los metamodelos en su espacio de trabajo con instancias dinámicas (".xmi").
- Un metamodelo hecho a medida.
- Un metamodelo hecho a medida con dependencia a UML.
- Modelos UML con perfiles.
- Varios metamodelos hechos a medida con dependencias de una a la otra.

6.7.2 Construido con las herramientas de su elección

Dado que Acceleo se basa en el popular framework EMF, puede crear su modelo con cualquier tipo de herramienta basada en EMF y utilizarlo como la entrada de su generación. Se puede generar código a partir de un modelo creado con:

- El editor basado en el árbol EMF.
- Un modelador gráfico.
- Un editor basado en texto realizado con xtext.
- Una aplicación basada en CDO (Connected Data Objects) utilizada para crear modelos remotos.

6.7.3 Tipo de tecnología a generar

Con su enfoque basado en plantilla que puede definir el tipo de código generado y personalizar su propio estilo de codificación. Se puede generar:

- Java.
- Scala.
- Javascript.
- HTML.
- Python.
- Acceleo no restringe el tipo de código generado, sólo hay una regla: si puede escribirlo, Acceleo puede generarlo.

6.7.4 Resumen de características

Entorno

Características	Acceleo
Entorno de edición	X
Vista previa de generación en tiempo real	X
Enfoque de plantilla	X
La integración completa dentro de Eclipse	X
Uso nativo de EMF	X

Energía

Características	Acceleo
Generación incremental	X
Extensibilidad Java	X
Reutilización de scripts	X
Sincronización de código-modelo	X
Control de consistencia	X
Apoyo PIM	X
Apoyo PSM	X
XMI 1.x	X
XMI 2	X
UML 1.x	X
UML 2	X
DSL	X
Cualquier tecnología de destino (C, Fortran, Java, Smalltalk, XML, etc)	X
Versionado para archivos generados	X

Pragmatismo

Características	Acceleo
Enfoque ascendente	X
Sintaxis legible y mantenible	X
100% modelos funcionales	X
Código abierto	X
Implementación rápida	X
Listo para usar módulos	X
Control de las evoluciones funcionales	X
Control de las evoluciones técnicas	X

Características en Acceleo 3.2

El resaltado de sintaxis	Terminación	Plegado de código	Compilador	Refactoring (plantilla extracto)
Plantillas de código personalizables	Declaración abierta	Perfilador	Depurador	Refactoring (renombrar)
Motor de generación	Esquema dinámico	Ejecución independiente	Detección de errores	Refactoring (consulta extracto)
Documentación de apoyo	Flotar	Soluciones rápidas	Trazabilidad	Refactoring (Pull up)

Espacios en blanco visibles	Esquema rápido	Buscar ocurrencias	Advertencias	Generación Incremental
Contenedor de path de clases	Apoyo Ant	Apoyo Maven		

6.7.5 Características del editor Acceleo

El editor Acceleo es el centro de la herramienta Acceleo. Se asocia con archivos de origen (archivos Acceleo .mtl) y ofrece todas las características de un editor de programación moderno para hacer más productivo el desarrollo de módulos de generación de código:

- **Resultado de sintaxis**

```

generate.mtl
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/2.1.0/UML')/]

[template public generate(c : Class)]

    [comment @main /]
    [file (c.name, false, 'UTF-8')]
    [c.name/]
    [/file]

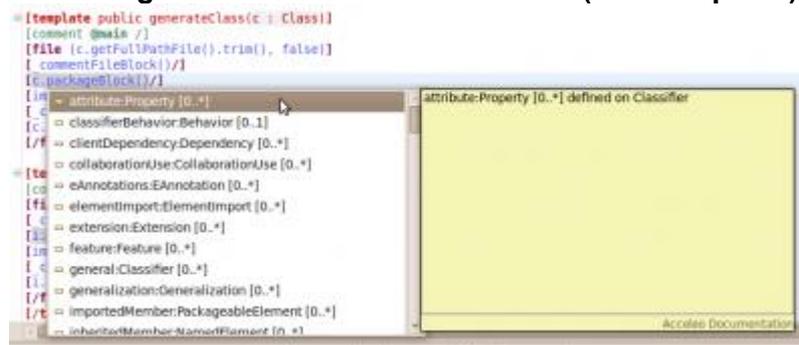
[/template]
    
```

El editor utiliza colores específicos para las plantillas Acceleo:

- rojo se utiliza para las etiquetas de plantilla;
- púrpura se utiliza para otras etiquetas (consultas, módulos, importaciones, ...);
- azul se utiliza para expresiones dinámicas de plantillas u otros lugares;
- verde se utiliza para comentarios y literales de cadena;
- negro se utiliza para cuerpos de texto o de consulta estáticas.

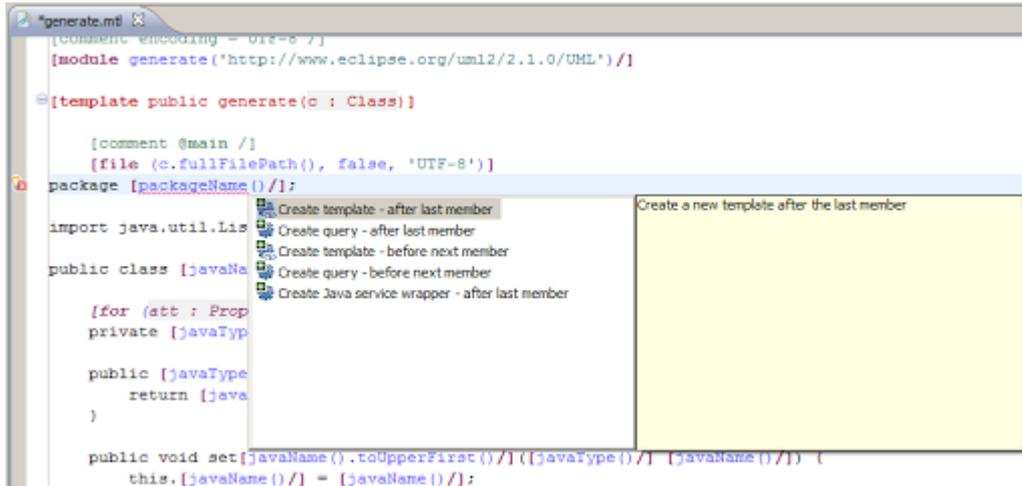
Usted puede cambiar los colores utilizados por el editor Acceleo gracias a un menú de preferencias accesible en el menú Windows -> Preferences.

- **Finalización inteligente o Asistente de contenido (Ctrl + Espacio)**



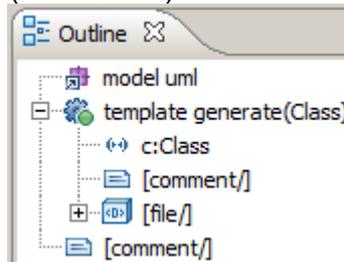
- **Soluciones rápidas (Ctrl + Mayús + 1)**

Actualmente, las soluciones rápidas proponen crear una plantilla o consulta, antes o después de la plantilla actual. En el siguiente ejemplo, acabamos de escribir la llamada a una plantilla que aún no existe, y el uso de la solución rápida para crear inmediatamente.



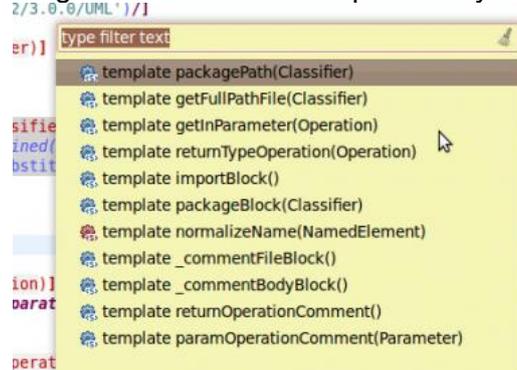
- **Esquema dinámico**

La estructura del módulo, las importaciones, las plantillas, las consultas se pueden ver allí. Al hacer doble clic sobre cualquiera de ellos coloca el cursor en la posición correspondiente en el módulo (en el editor).



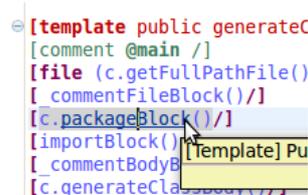
- **Esquema rápido (Ctrl + O)**

Muestra una ventana emergente con una lista de plantillas y consultas.



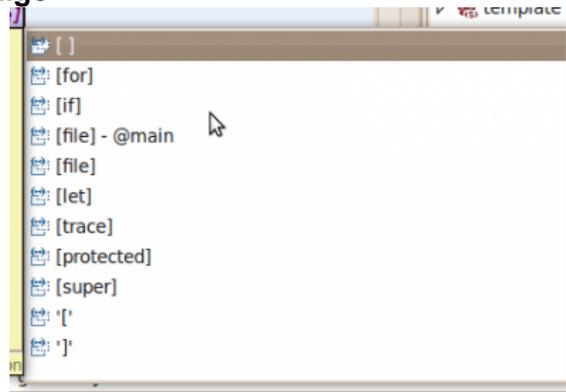
- **Declaración abierta o Navegación (Ctrl + click izquierdo o F3)**

Te lleva a la declaración del elemento seleccionado.



Los hipervínculos en el editor Acceleo están disponibles para navegar rápidamente a las definiciones de los elementos y operaciones del modelo.

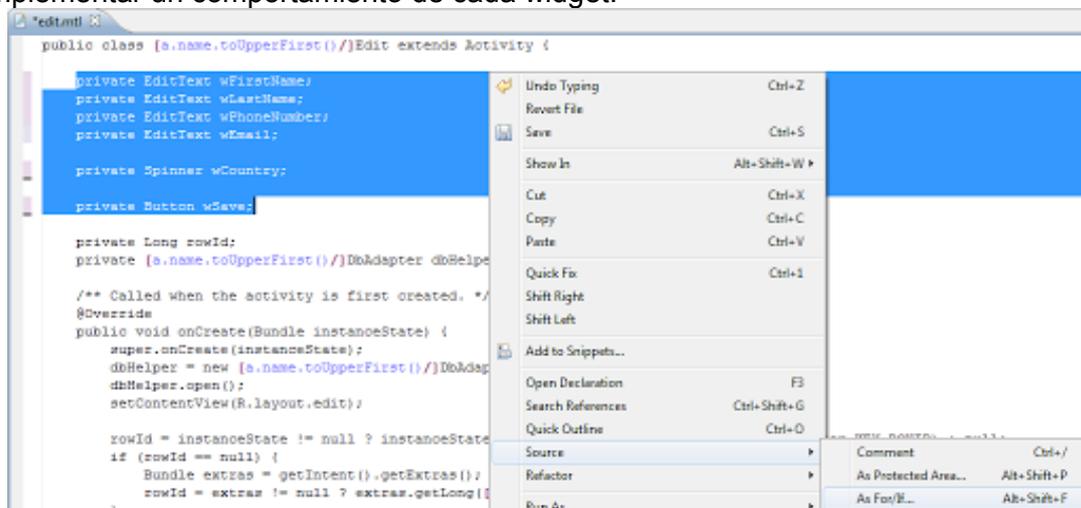
- Plantillas de código



- **Detección de errores** en tiempo de compilación,
- **Plegado de código,**
- **Buscar referencias (Ctrl + Shift + G)** obteniendo todos los elementos que hacen referencia a un elemento dado,
- **Advertencia e informaciones de apoyo** (advertencias, TODO y FIXME),
- **Documentación flotada**
- **Bloques If/For rápidos**

Se podría tener múltiples tipos de widgets: Texto, Spinner, Button... y, a partir de un ejemplo, se quiere personalizar el módulo de generación para cada uno de estos widgets.

El ejemplo a continuación muestra un ejemplo de un lugar donde queremos implementar un comportamiento de cada widget:



Una vez que la acción se ejecutó, terminamos con esto:

```

public class [a.name.toUpperFirst()]Edit extends Activity {

    [for (w : Widget | widgets)
        [if (oclIsKindOf())
            private EditText wFirstName;
            private EditText wLastName;
            private EditText wPhoneNumber;
            private EditText wEmail;

            [elseif (oclIsKindOf())
                private Spinner wCountry;

            [elseif (oclIsKindOf())
                private Button wSave;

        [endif]
    [endif]
}

```

Por supuesto, ésto significa que todavía tenemos que cambiar las condiciones de estas sentencias If; pero simplifica la transformación de la plantilla en lo que necesitamos.

• Refactorización

Acceleo contiene varios procesos de refactorización con el fin de ayudar al usuario a editar sus módulos Acceleo.

- **Renombre** (Alt + Shift + R),
- **Extraer como plantilla o como consulta,**
- **Pull up**

La operación "pull up" permite al usuario sacar algunas plantillas o consultas en otro módulo y tener todos los enlaces de herencia conectados.

- **Transformar a Área Protegida**

Del mismo modo, es muy útil para marcar algún área de código como protegida. Por ejemplo, en el siguiente código, puede ser útil proteger el área de las importaciones con el fin de mantener las importaciones requeridas por el código de usuario después de cada regeneración.

```

[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/2.1.0/UML')]

[template public generate(c : Class)]

    [comment @main /]
    [file (c.name.concat('.java'), false, 'UTF-8')]
    package [newClassTemplate()];

import java.util.List;

    pu
    first() {

    }

    [template (c : Class) ]
    [c.ancestors(Package).name->sep('.')/]
    [template]

    [comment
        Renvoie les attributs public d'une classe.
    /]
    [query public getPublicAttributes(c : Class) : Set(Property) =
        c.attribute->select(visibility = VisibilityKind::public)
    /]

```

El texto seleccionado está rodeado luego con marcadores [protected/], con un id automáticamente inferido.

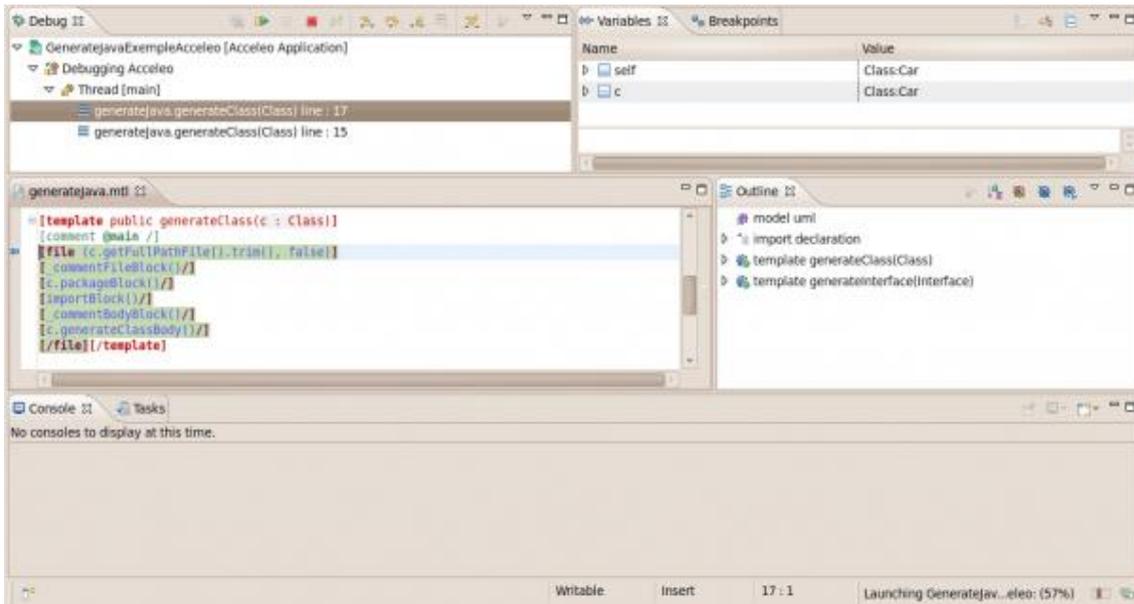
6.7.6 El lado oscuro del estándar

- La navegación en OCL es compleja.
- No existen los servicios de java, todos se describen en OCL.

- El manejo de cadenas de texto es menos potente que en `acceleo.org` Unicode.

6.7.7 Depurador

Acceleo contiene un depurador que se puede utilizar con el lanzamiento de la generación con el menú "Debug As...". Si está utilizando el lanzamiento Acceleo Plug-In Application, se utilizará el depurador Acceleo, el cual funciona como el depurador de Java. Como tal, se le permite colocar puntos de interrupción y mover instrucciones por instrucciones en la ejecución de su módulo. También puede ver el estado de todas sus variables.

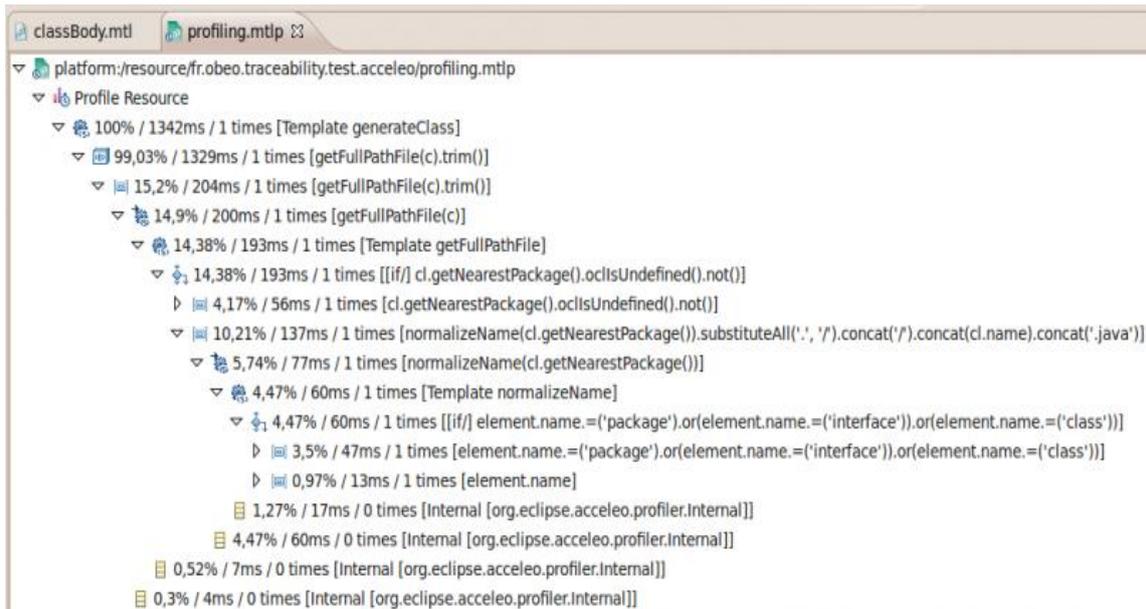


6.7.8 Perfilador

El generador de perfiles permite al usuario averiguar las instrucciones ejecutadas durante la generación, el número de veces que esas instrucciones se han ejecutado y el tiempo empleado por cada una de esas instrucciones, por lo que es más fácil de identificar cuellos de botella.

El perfilador es accesible gracias al menú "Profile As..." (en lugar de "Run As..."). Sólo hay una información adicional para configurar, que es la ruta de acceso al archivo de resultados de perfiles donde Acceleo almacenará la información de perfiles de ejecuciones posteriores. Los archivos del perfil deben tener la extensión `.mtlp`. Estos archivos son en realidad sólo un modelo EMF serializado.

Se puede ver en esta imagen el resultado de una generación con el perfil activado y utilizarlo para crear un informe.



6.7.9 Trazabilidad (localizar los orígenes de su código generado)

El motor Acceleo puede calcular la información de los elementos que intervienen en la generación de un archivo. Este sistema permite, por ejemplo, determinar los elementos de los modelos de entrada que se han utilizado para generar una pieza específica de texto y la parte del generador de código que ha estado involucrado. Se puede activar en la configuración de lanzamiento y el resultado se puede ver en la vista de resultados.

6.7.10 Stand-alone

El analizador y el motor de generación, componentes críticos de Acceleo, también se pueden utilizar en "stand-alone", sin ser desplegado en Eclipse. Acceleo genera una clase Java para iniciar la generación de programación permitiendo así la integración de un generador de Acceleo en cualquier aplicación Java. Este lanzador Java también puede llamarse desde Ant o Maven.

Compilación construir su generador fuera de Eclipse

Para compilar su generador Acceleo, puede utilizar el plug-in Maven o puede hacerlo mediante programación.

- Programación: puede utilizar la clase Java, `org.eclipse.acceleo.internal.parser.compiler.AcceleoParser` para lanzar la compilación mediante programación sólo tiene que crear una instancia de un `AcceleoProject` y entonces usted tiene que llamar a `buildAll (new BasicMonitor())`.
- Maven: la integración Acceleo Maven evoluciona de forma independiente de los lanzamientos de Acceleo, se recomienda comprobar la wiki Acceleo para una puesta al día de la documentación sobre el tema: <http://wiki.eclipse.org/Acceleo/Maven>.

Ejecución lanzar su generador fuera de Eclipse

Un generador Acceleo puede ser lanzado desde cualquier aplicación J2SE-1.5+.

Con el fin de poner en marcha el generador, es necesario crear una instancia de ella, para darle el URI de su modelo y la carpeta de salida donde se generará el código.

```
String path = "C:/Users/anUser/Desktop/generators/uml2java/model/example.uml";
```

```
URI modelURI = URI.createFileURI(URI.decode(path));
```

```
File targetFolder = new File("C:/Users/anUser/Desktop/generators/uml2java/result");
```

```
GenerateJava generator = new GenerateJava(modelURI, targetFolder, new ArrayList<Object>());
generator.doGenerate(new BasicMonitor());
```

En este ejemplo, "GenerateJava" es el lanzador Java de nuestro generador Acceleo.

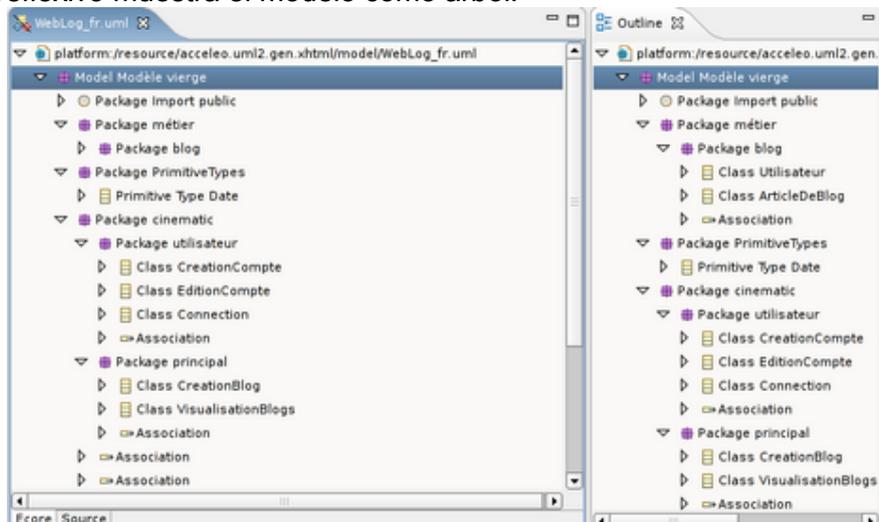
6.7.11 Editor reflexivo

Para facilitar el diseño de la plantilla de generación, el editor de reflexión ofrece una previsualización de tiempo real del resultado de la generación.

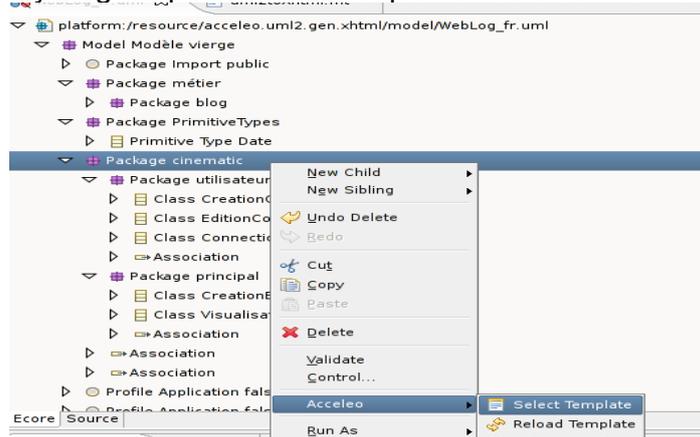
Desde un modelo exportado, el editor reflexivo muestra una vista de árbol.

Entonces, el editor reflexivo se utiliza para probar la validez de una plantilla de generación directamente en un modelo, y obtener una vista previa del resultado. Si la plantilla o modelo ha cambiado, la vista previa se actualiza automáticamente para visualizar los impactos.

El editor reflexivo muestra el modelo como árbol.



Una vez que el modelo se abre en el editor de reflexión, se puede asociar un generador de plantilla con el modelo. Haga clic en Acceleo -> Select Template en el editor de reflexión y elegir la plantilla vacía o que hemos creado anteriormente.



Selección de un generador de plantilla

Ahora vamos a llenar o modificar la plantilla con el fin de generar texto.

La siguiente captura de pantalla muestra una plantilla que genera una descripción de HTML para cada clase, con sus atributos y comentarios.

```

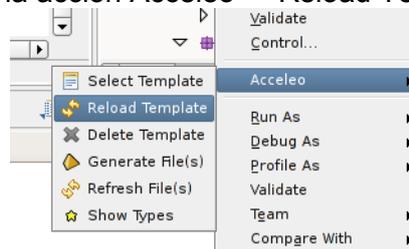
WebLog_fr.uml | uml2toXhtml.mt
<%%
metamodel http://www.eclipse.org/uml2/2.0.0/UML
%%>
<%%script type="uml.Class" name="uml2toXhtml" file="<name%>.html"%%>
<html>
  <head/>
  <body>
    <h1>Class Description</h1>
    <p>Name of class : <name%></p>
    <p>Comment : <ownedComment.body%>

    <h1>Attributes</h1>
    <%%if (attribute.nSize() == 0){%>
    <p>No attributes.</p>
    <%%else{%>
    <ul>
      <%%for (attribute){%>
      <li><name%> : <type.name%></li>
      <%%}%>
    </ul>
    <%%}%>
  </body>
</html>

```

Generador de plantilla a XHTML

Una vez que la plantilla se escribe y se guarda puede actualizar la vista previa mediante un clic derecho y la acción Acceleo -> Reload Template.



Recarga de una plantilla

El editor reflectante muestra una vista previa del código generado en la ficha Source.

```

WebLog_fr.uml | uml2toXhtml.mt
<html>
  <head/>
  <body>
    <h1>Class Description</h1>
    <p>Name of class : Utilisateur</p>
    <p>Comment :

    <h1>Attributes</h1>
    <ul>
      <li>email : String</li>
      <li>prenom : String</li>
      <li>nom : String</li>
      <li>login : String</li>
      <li>motDePasse : String</li>
    </ul>
  </body>
</html>

```

Puede repetir este paso tanto como sea necesario con el fin de ajustar el generador de plantilla.

6.7.12 Bloques de código de usuario

Bloques de código de usuario delimitan porciones de texto que sólo se generan una vez, y quedan preservados para las generaciones posteriores. Ésto es muy útil para permitir a los usuarios añadir código en algunas áreas de los archivos generados, manteniendo el resto del archivo bajo el control del generador.

Acceleo soporta dos formas de especificar bloques de código de usuario en el código generado.

La primera forma de especificar bloques de código de usuario es utilizar el constructor estándar `[protected (id)] ... [/protected]`, como se muestra a continuación:

```

generate.mtl X
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/3.0.0/UML' /)]

[template public generate(aClass : Class)]
[file (aClass.classFileName(), false)]
package [aClass.containingPackages().name->sep('.')]

// [protected ('imports')]
// [/protected]

public class [aClass.name.toUpperFirst()] {
[for (p: Property | aClass.attribute) separator('\n')]
private [p.type.name/] [p.name/];
[/for]

[for (p: Property | aClass.attribute) separator('\n')]
public [p.type.name/] get[p.name.toUpperFirst()]() {
return this.[p.name/];
}
[/for]

[for (o: Operation | aClass.ownedOperation) separator('\n')]
public [o.type.name/] [o.name/]() {
// [protected (o.name)]
// TODO should be implemented
// [/protected]
}
[/for]
}
[/file]
[/template]

```

El módulo anterior produce el siguiente código:

```

generate.mtl Person.java X
package org.eclipse.acceleo.java

// Start of user code imports
// End of user code

public class Person {
private String emails;

private Date dateOfBirth;

private String name;

private String firstname;

public String getEmails() {
return this.emails;
}

public Date getDateOfBirth() {
return this.dateOfBirth;
}

public String getName() {
return this.name;
}

public String getFirstname() {
return this.firstname;
}

public Integer getAge() {
// Start of user code getAge
// TODO should be implemented
// End of user code
}

public String getFullName() {
// Start of user code getFullName
// TODO should be implemented
// End of user code
}
}

```

La segunda forma, llamada JMerge, que es específico a los generadores del lenguaje Java. Con la anotación JMerge "@generated" puede indicar que este componente se ha generado y que debe ser eliminado y se regenera mientras que una etiqueta "@generated NOT" indicará que todo el contenido del elemento documentado se ha cambiado por el usuario y no debe ser sobrescrito por la nueva generación incluso fuera de un área protegida.

```

generate.mtl Person.java
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/3.0.0/UML')]

[template public generate(aClass : Class)]
[file (aClass.classFileName(), false)]
package [aClass.containingPackages().name->sep('.')]

/**
 * @generated
 */
public class [aClass.name.toUpperFirst()] {
[for (p: Property | aClass.attribute) separator('\n')]
/**
 * @generated
 */
private [p.type.name/] [p.name/];
[/for]

[for (p: Property | aClass.attribute) separator('\n')]
/**
 * @generated
 */
public [p.type.name/] get[p.name.toUpperFirst()]() {
return this.[p.name/];
}
[/for]

[for (o: Operation | aClass.ownedOperation) separator('\n')]
/**
 * @generated NOT
 */
public [o.type.name/] [o.name/]() {
// TODO should be implemented
}
[/for]
}
[/file]
[/template]

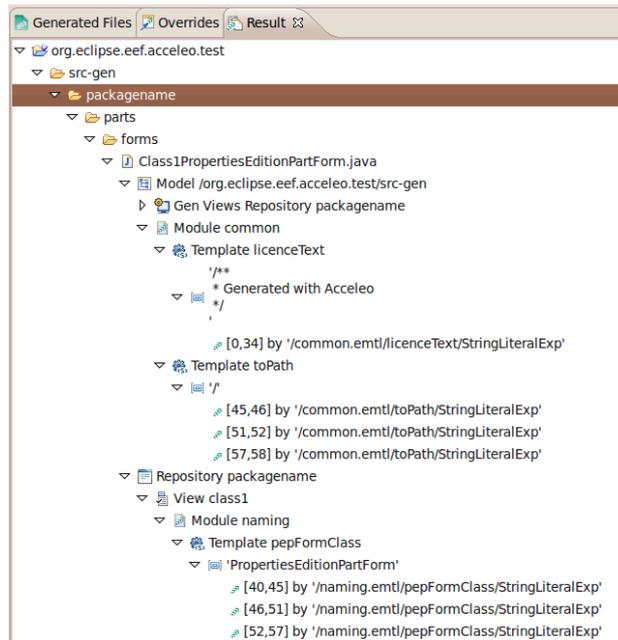
```

En ambos ejemplos, el código se encuentra en áreas "protegidas" nunca será reemplazado por las generaciones posteriores, ya sea dejado como está o modificado por el usuario fuera del módulo (directamente en el archivo generado).

6.8 Vistas de la perspectiva Acceleo

6.8.1 Vista Result (muestra la sincronización del modelo, los módulos y el código generado)

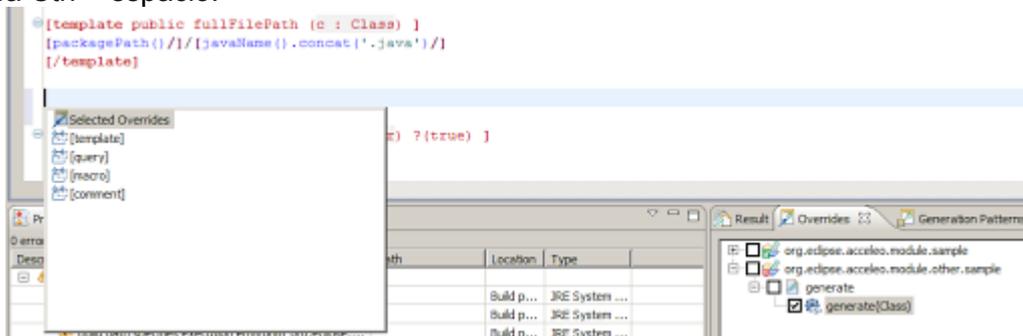
La vista Result permite obtener la información de trazabilidad que se produce durante la generación. En la vista Result, se puede ver todos los archivos generados y los elementos a partir de los modelos de entrada utilizados para generar cada una de las regiones del archivo con la región desde el generador.



6.8.2 Vista Overrides (para anular un comportamiento existente en una plantilla)

La vista Overrides se puede utilizar para crear fácilmente otras plantillas o módulos de sobreescritura. Se permite al usuario ver todos los módulos Acceleo disponibles en su área de trabajo y en sus plug-ins.

Para anular una o varias plantillas existentes, sólo tienes que seleccionar en esta vista cada checkbox. A continuación, edite el módulo en el que va a reemplazar las plantillas, colocando el cursor donde desea insertar las plantillas reemplazadas, y pulsa Ctrl + espacio.



Seleccione la primera opción ("Selected Overrides"). Se crean entonces las plantillas de reemplazo.

```

[template public fullPath (c : Class) ]
[packagePath()]/[javaName().concat('.java')]
[/template]

[comment @Override generate.generate /]
[template public generate(c : Class) overrides generate ]

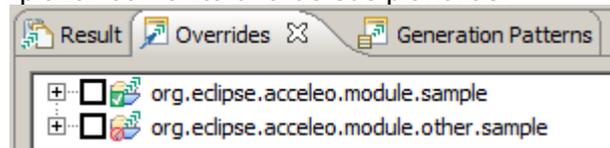
    [comment @main /]
    [file (c.name, false, 'UTF-8')]
    [c.name/]
    [/file]

[/template]

[template public javaName(e : Classifier) ?(true) ]

```

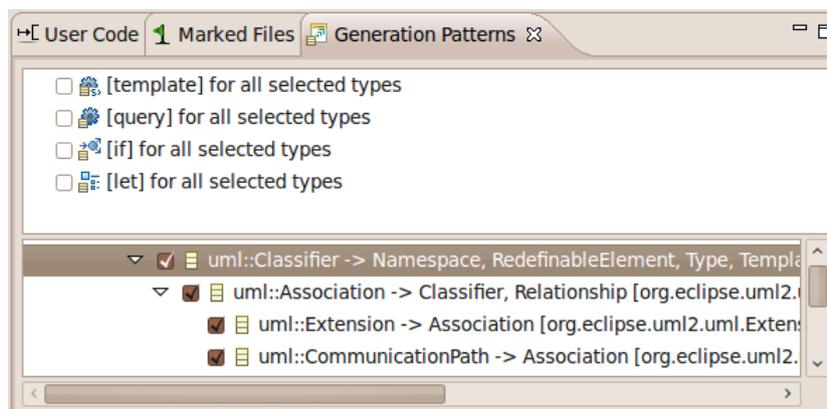
Nota: Un marcador indica si un determinado proyecto es accesible desde el suyo. Si éste es el caso, una marca verde indica que todo está bien. De lo contrario, un marcador rojo indica que es necesario importar el proyecto en el suyo para poder anular una plantilla que contiene. Por ejemplo, en la pantalla de abajo, org.eclipse.acceleo.module.other.sample se necesita importar en su proyecto actual antes de poder reemplazar con éxito una de sus plantillas.



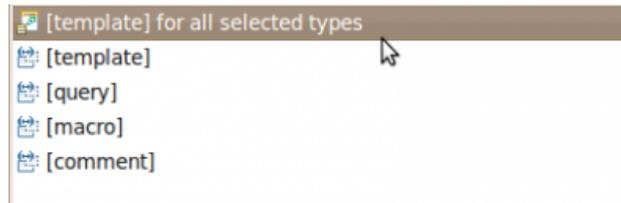
6.8.3 Vista Generation Patterns (para definir su propia propuesta de finalización)

La vista Generation Patterns le permite al usuario seleccionar uno de los patrones de diseño de generación de código y elegir los elementos en los que se aplicará el patrón. Un ejemplo común es el uso de polimorfismo que se puede lograr muy fácilmente con esta vista.

Tenga en cuenta que la parte inferior de la vista presenta una vista jerárquica del metamodelo que está utilizando. Cada nodo representa un tipo del metamodelo y contiene todos sus subtipos. Seleccione los tipos de los que desea crear plantillas "name".



Después de esta selección, vaya al editor y active el mecanismo de finalización pulsando Ctrl + Espacio. Seleccionar la primer opción que debe ser "[template] for all selected types".



Y aquí se puede ver el resultado de un patrón de diseño que se utiliza para crear una nueva plantilla para todas las clases de UML "classifier".

```
[template public name(e : Classifier) ]
[comment TODO Auto-generated template stub/]
[/template]

[template public name(e : Association) ]
[comment TODO Auto-generated template stub/]
[/template]

[template public name(e : Extension) ]
[comment TODO Auto-generated template stub/]
[/template]

[template public name(e : CommunicationPath) ]
[comment TODO Auto-generated template stub/]
[/template]

[template public name(e : AssociationClass) ]
[comment TODO Auto-generated template stub/]
[/template]

[template public name(e : Artifact) ]
[comment TODO Auto-generated template stub/]
[/template]

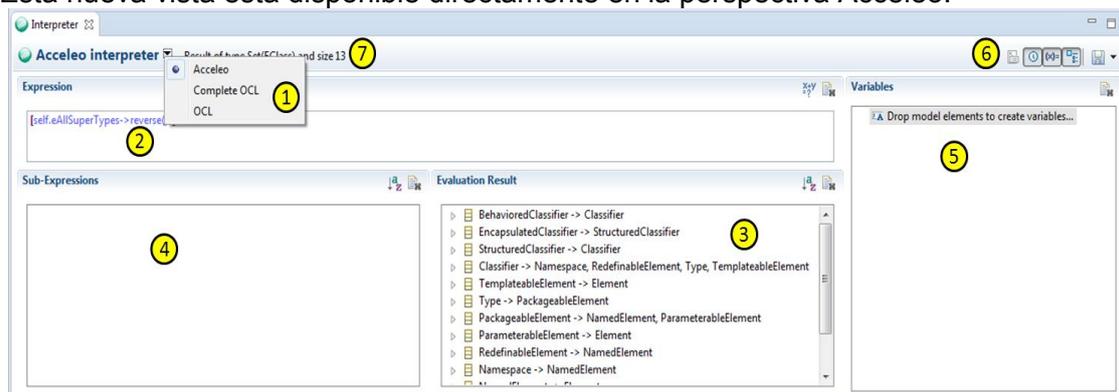
[template public name(e : DeploymentSpecification) ]
[comment TODO Auto-generated template stub/]
[/template]
```

Aquí, Aceleo ha hecho su trabajo, ahora es el momento para que usted haga el suyo: implementar estas nuevas plantillas.

6.8.4 Vista Interpreter

Esta vista permite al usuario introducir expresiones Aceleo y ejecutarlas sin la necesidad de lanzar una generación.

Esta nueva vista está disponible directamente en la perspectiva Aceleo.



- Selección del lenguaje
- Expresión
- Resultado de la evaluación
- Sub-Expresiones
- Variables
- Acciones
- Retroalimentación

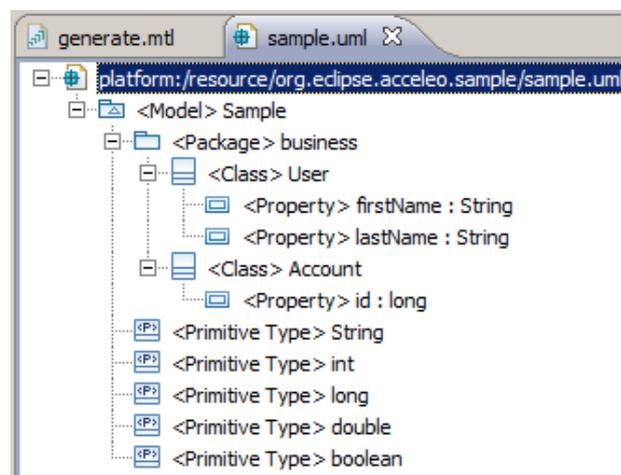
6.9 Proyecto Acceleo

6.9.1 Dependencias

Un proyecto Acceleo, siendo un proyecto plugin de Eclipse, maneja sus dependencias gracias a OSGI. Todos los proyectos Acceleo contienen el archivo MANIFEST.MF en la carpeta META-INF. Una vez abierto, puede ir a la pestaña "dependencias" para agregar una dependencia a otro plug-in Eclipse necesario para su generador. Por ejemplo, si va a crear un generador utilizando modelos UML, es necesario añadir la dependencia al metamodelo de UML "org.eclipse.uml2.uml".

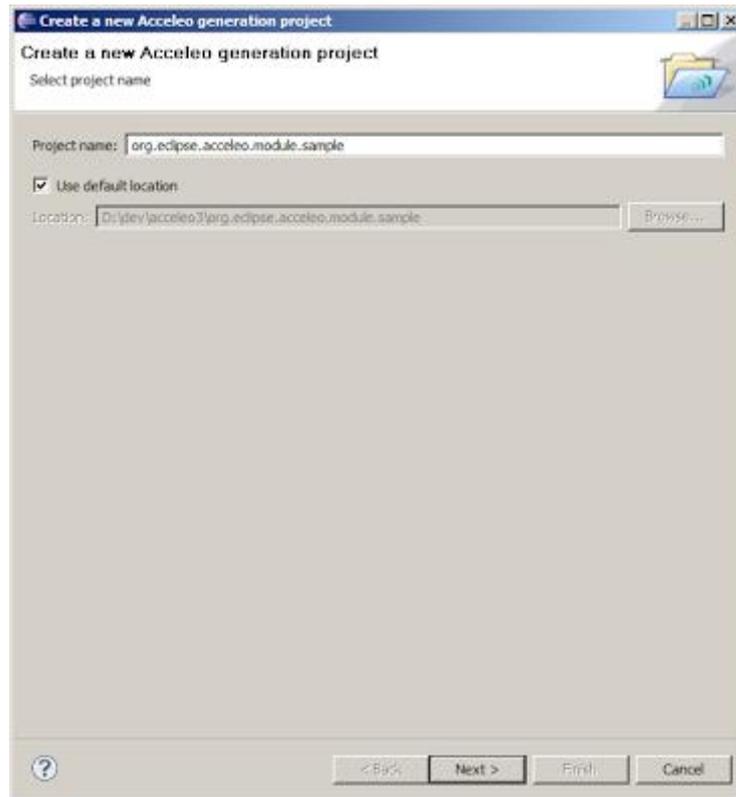
6.9.2 Creación de un Proyecto Acceleo "desde cero"

El objetivo de un proyecto Acceleo es generar texto de un modelo (o de un conjunto de modelos). Vamos a crear un nuevo módulo Acceleo para generar Java Beans desde un modelo UML.



Para crear un nuevo proyecto Acceleo, haga clic derecho en la vista Explorador de paquetes y a continuación, seleccione New -> Acceleo Project.

Elige un nombre para el proyecto (en este caso, org.eclipse.acceleo.module.sample), a continuación, haga clic en Next.



Esta segunda página del asistente le permite inicializar el proyecto mediante la creación de uno o varios archivos de módulo Acceleo.

- Seleccione la carpeta en la que desea crear el nuevo archivo de módulo.
- Rellene el nombre del módulo (lo dejaremos como generate).
- Seleccione el metamodelo de la que su archivo de generación tomará sus tipos (en este ejemplo, UML).
- Por último, elegir la metaclase que se utilizará para generar el archivo (en este ejemplo, Class). Ésta se puede modificar más adelante en cualquier momento directamente en los archivos de módulo.



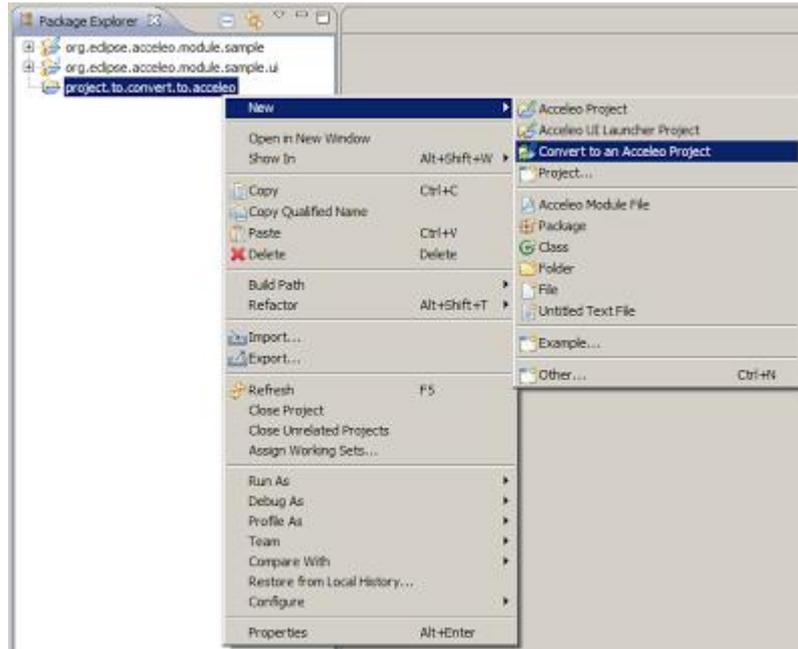
Usted puede crear más de un archivo de módulo en este proyecto mediante el botón "Add" en la izquierda.

Al hacer clic en Finish creará el archivo(s) del módulo, y algunos archivos generados automáticamente de ella.

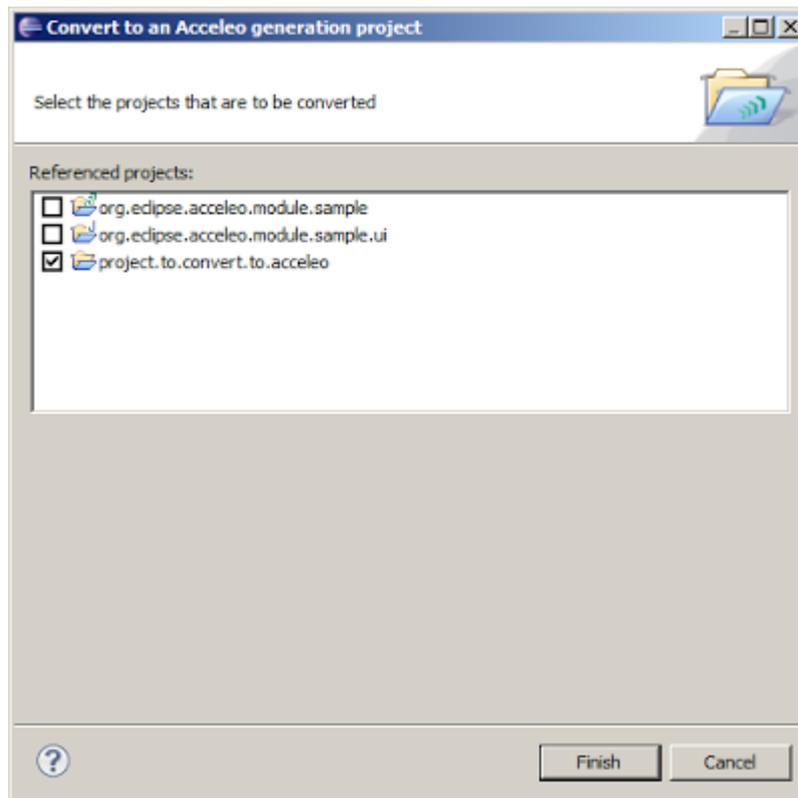


6.9.3 La transformación de un proyecto existente en un proyecto Acceleo

Ésto se puede lograr haciendo clic derecho en el explorador de paquete, a continuación, seleccionar New > Convert to an Acceleo Project.



Seleccione el proyecto(s) que debe ser convertida en proyecto Acceleo(s), y haga clic en Finish.



Nota: La naturaleza Acceleo puede retirarse de un proyecto Acceleo simplemente haciendo clic derecho sobre el proyecto y seleccionando Acceleo> Remove Acceleo Nature.



6.9.4 Creación de un Proyecto Acceleo UI

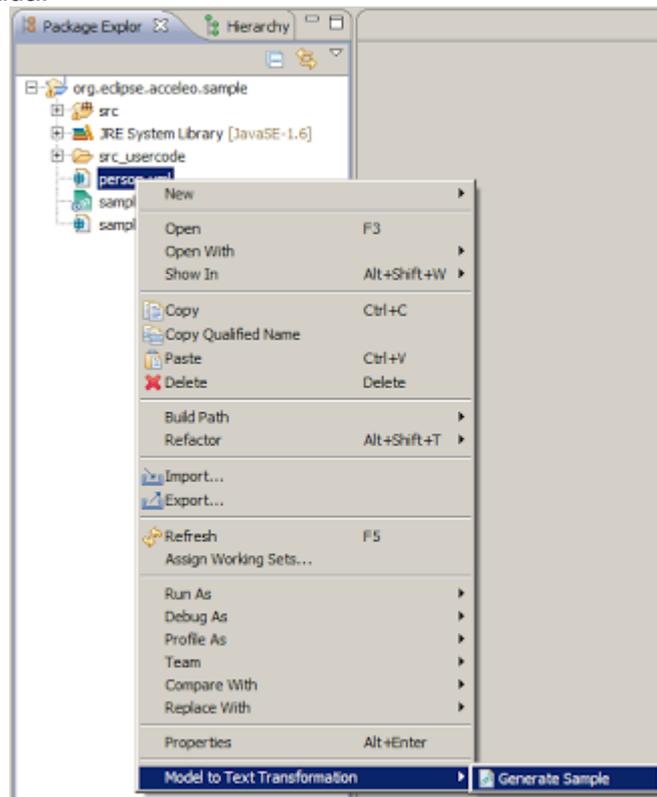
Ahora que sus módulos de generación están listos, es posible que desee tener algunos asistentes para lanzar la generación desde dentro de Eclipse. Para no tener que manipular un proyecto Acceleo, puede utilizar un asistente proporcionado por Acceleo para crear una interfaz de usuario de Eclipse básica para su generador. Este asistente creará un nuevo proyecto Eclipse que le permitirá iniciar la generación con una acción del botón derecho sobre cualquier modelo apropiado.

Haga clic derecho en su proyecto Acceleo (org.eclipse.acceleo.module.sample) a continuación, seleccione New > Acceleo UI Launcher Project.

Este asistente requiere la siguiente información:

- nombre del proyecto generador UI,
- nombre del proyecto de referencia,
- nombre generador (este nombre se mostrará a los usuarios en la interfaz de usuario),
- tipo de archivo que filtra en qué archivo de extensiones aparecerá el menú emergente (*.uml para un generador UML),
- carpeta donde se generará el código, por defecto, se genera en una carpeta llamada "src-gen".

El asistente creará un nuevo plugin con todo el código necesario para mostrar una nueva acción para el archivo de modelo seleccionado que va a generar el código en la carpeta especificada.



7 Comparación de las herramientas

7.1 Aspectos teóricos

	App Inventor	Rational Rhapsody	Acceleo
Creador	Google Labs (2010)	I-Logix Inc (1996)	Obeo (2006)
Desarrollador actual	MIT	IBM (Rational Software)	Fundación Eclipse
Licencia	Freeware	Shareware	Licencia Pública Eclipse
SO	Windows, Linux y Mac OS X	Windows y Linux	Windows, Linux y Mac OS X
Última versión	App Inventor 2 (06/12/2013)	8.1.5 (25/04/2016)	3.5.1 (05/03/2015)

Código fuente: Acceleo es un proyecto de la Fundación Eclipse mayormente desarrollado en Java, su código fuente está disponible en Github...mientras que el editor de bloques de App Inventor utiliza la librería Open Blocks de Java para crear un lenguaje visual a partir de bloques. Estas librerías están distribuidas por MIT bajo su licencia libre (MIT License). El compilador que traduce el lenguaje visual de los bloques para la aplicación en Android utiliza Kawa como lenguaje de programación, distribuido como parte del sistema operativo GNU de la Free Software Foundation. La familia de programas IBM Rational® cubre un amplio espectro en lo que a ámbitos de trabajo se refiere, haciendo un uso intensivo de los lenguajes UML y SysML.

Metamodelos y lenguajes: En lugar de escribir código, App Inventor diseña visualmente la forma en que la aplicación se ve, enlazando bloques para especificar el comportamiento de la aplicación...mientras que Acceleo genera código de cualquier metamodelo, ya sea Ecore, UML o DSL, con tal de que se pueda leer a través de EMF (Framework de Modelado en Eclipse), para generar cualquier tipo de tecnología (Java, C, PHP...), sólo hay una regla: si puede escribirlo, Acceleo puede generarlo...con Rational Rhapsody se puede modelar en diagramas UML, SysML o DSL a partir del que se puede crear código para los lenguajes C, C++, C#, Java y Ada...por su parte App Inventor crear aplicaciones de software exclusivamente para el sistema operativo Android.

Entorno de desarrollo: Tanto Rational Rhapsody como Acceleo utilizan un plug-in para integrarlo dentro del entorno Eclipse, mientras que App Inventor es una herramienta basada en la Internet, mediante la que se construyen aplicaciones desde un navegador web Drag & Drop para ir enlazando los bloques.

Independencia del entorno de desarrollo: Acceleo viene integrado como plugin de Eclipse, pero puede ser utilizado, compilado y ejecutado fuera de Eclipse al igual que Rhapsody el cual dispone de un software totalmente independiente y además una interfaz de línea de comandos, mientras que App Inventor también es una herramienta independiente, pero que se utiliza desde un navegador web.

Entradas y salidas: Un generador Acceleo se compone de varios archivos llamados módulos. Un módulo es un archivo .mtl, que se compone de varias plantillas y/o consultas. Las consultas se utilizan exclusivamente para extraer información del modelo. Una plantilla contiene expresiones estáticas (generadas sin transformaciones) y expresiones Acceleo, basadas en OCL, que utilizarán las consultas, los elementos del modelo de entrada y los archivos de propiedades Java para generar el código.

Algo similar sucede en Rational Rhapsody donde las entradas al generador de código son el modelo y las propiedades de generación de código. Las salidas del generador de código son archivos de código fuente en el lenguaje de destino.

En cambio App Inventor cuenta con el Diseñador y el Editor de Bloques. En el Diseñador se construye la interfaz de usuario de la aplicación, seleccionando los componentes y editando sus propiedades de visualización. Mientras que en el Editor de Bloques, se configura el comportamiento de la aplicación, editando las propiedades de comportamientos de cada componente, mediante el ensamblado de bloques de programa que especifican cómo los componentes deben comportarse antes los eventos. El equipo de la aplicación Inventor ha creado bloques para casi todo lo que puedes hacer con un teléfono Android.

Repositorio: A diferencia de las demás herramientas analizadas los servidores de App Inventor pueden almacenar sus proyectos y permiten compartir sus aplicaciones creadas.

Depuración: Acceleo contiene un depurador que se puede utilizar con el lanzamiento de la generación. Rational Rhapsody cuenta con una perspectiva que le permite abrir diagramas simulados para observar el modelo y llevar a cabo la depuración a nivel de diseño en la perspectiva de depuración. En la depuración de App Inventor se puede ir testeando la aplicación en el teléfono a medida que se la construye, al probar la opción Do It en un bloque se observará el comportamiento en el teléfono y se mostrará el resultado en un globo del bloque, además de los diferentes valores que van tomando las variables con la opción "watch" activada.

Trazabilidad: En Rational Rhapsody se pueden crear enlaces de trazabilidad a partir del modelo hacia los requisitos. En Acceleo se puede ver la relación entre modelo de entrada, módulo generador y código generado desde la vista Result. Mientras que App Inventor no dispone de trazabilidad.

Facilidad de uso: App Inventor está dirigida a personas que no están familiarizadas con la programación informática, las cuales pueden tener su primera aplicación en funcionamiento en una hora o menos, y se pueden programar aplicaciones más complejas en mucho menos tiempo que con los lenguajes más tradicionales, basados en texto. Mientras que para utilizar Acceleo y Rational Rhapsody se necesitan conocimiento sobre modelado y lenguajes de programación.

Sincronización: Rational Rhapsody le permite trabajar en el modelo o código y mantener la sincronización entre cada uno para que los cambios en una se reflejan en la otra de forma automática (DMCA). Roundtripping es un método utilizado para actualizar el modelo con pequeños cambios introducidos al código generado previamente. Este mecanismo no es aplicable en Acceleo ni App Inventor.

Uso del emulador: A medida que construye su aplicación App Inventor, se pueden hacer "pruebas reales" en un emulador mediante el programa aiStarter o en un dispositivo Android. Cuando se desarrolla en Android, tanto en Acceleo como en Rhapsody, el SDK nos permite instalar diferentes versiones de Android y el AVD

Manager nos da la posibilidad de administrar las características de los emuladores con los cuales podemos correr nuestros demos.

Generación incremental: Tanto en Rhapsody como en Acceleo se cuenta con generación de código de forma incremental, donde se genera una pieza de código, luego se modifica el código generado y finalmente se regenera el código una vez más sin perder las modificaciones anteriores.

En Rhapsody, el comando Generate genera código solamente para los elementos que se han modificado desde la última generación, mientras que el comando Re Generate se genera el modelo entero. La generación de código por defecto es usar el procesamiento en paralelo. En Acceleo se usan bloque de código de usuario tanto a través de la etiqueta [protected] o con la anotación JMerge. En App Inventor, se trabaja directamente en el teléfono, donde se tiene la posibilidad de ir desarrollando y probando las aplicaciones de forma incremental, lo que permite aislar errores más rápidamente y corregirlos más fácilmente.

Previsualización: El editor de reflexión de Acceleo facilita el diseño de la plantilla de generación, permitiendo la previsualización en tiempo real de la generación. En Rhapsody se puede visualizar el modelo a través de la simulación de diagramas (animación) lo que permite controlar el diseño a medida que se crea.

Al mismo tiempo, en App Inventor a través de una conexión en directo entre el ordenador y el teléfono, la aplicación aparece en el teléfono paso a paso a medida que añade piezas a la misma, por lo que puede poner a prueba su trabajo a medida que construye.

Perspectivas y vistas: La plataforma de integración de Rhapsody dentro de Eclipse añade las perspectivas de modelado Rational Rhapsody y de depuración Rational Rhapsody. La primera de ellas cuenta con un editor que contiene las herramientas para modelar diagramas, mientras que la segunda se utiliza para depurar el modelo. Para la versión Acceleo integrada en Eclipse tenemos la perspectiva Acceleo la cual contiene diferentes vistas entre ellas Interpreter, Result, Overrides, etc. y el editor donde se desarrollan módulos de generación de código. La estructura de App Inventor se ejecuta en un navegador para visualizar el Diseñador, donde se seleccionan los componentes para su aplicación y el Editor de Bloques, donde se ensamblan bloques de programa que especifican cómo los componentes deben comportarse.

Generación de archivos: Mediante las etiquetas de archivos [file] cada plantilla le indica al motor Acceleo que debe crear un archivo real con el código generado. Entre los parámetros de la etiqueta, está el nombre del archivo (con extensión) y su codificación.

Estos archivos se crearán bajo la propiedad Target (ej. la ruta raíz de su proyecto Android) que será configurada a la hora de generar el código.

Para editar una aplicación App Inventor se puede compartirla en forma de código fuente (.aia) y para ejecutarla sin estar conectado a App Inventor, se debe empaquetar la aplicación para producir un paquete de aplicación en formato ejecutable (.apk, variante del formato JAR de Java).

En Rational Rhapsody cuando se crea un proyecto, una carpeta que contiene los archivos del proyecto se crea automáticamente en la ubicación que usted especifique. En la versión para desarrolladores de Java, mediante la propiedad JAVA_CG::Configuration::JarFileGenerate, se puede crear un archivo JAR cuando se genera el proyecto. A la hora de la generación del código se debe configurar el componente activo.

Sobreescritura: Con el fin de anular el comportamiento de una plantilla, Acceleo cuenta con dos maneras de sobreescritura, la estática y la dinámica. Para manejar esto, Acceleo dispone de la vista Overrides. En Rational Rhapsody la sobreescritura se aplica para los estereotipos, los cuales heredan y pueden agregar nuevos elementos a los que se puede aplicar, sus propiedades (con sus correspondientes valores) y las etiquetas. En App Inventor este concepto no es necesario.

Estructuras de control: En Acceleo las estructuras de control se basan en un súper conjunto del lenguaje OCL. Entre éstas se cuenta con bucles For, condiciones If, bloques Let, comentarios y sus variables son constantes.

El equipo de la aplicación Inventor ha creado bloques para casi todo lo que puedes hacer con un teléfono Android, así como bloques de control para programar (if, for each, while, etc.), comentarios, variables globales y locales.

En Rational Rhapsody a la hora de implementar una operación en el modelo o desde el código, se programa usando las mismas estructuras de control que provee el lenguaje del producto, en este caso Java.

Resaltado de sintaxis: El editor Acceleo contiene resaltado de sintaxis, Rhapsody lo hereda de Eclipse mientras que en App Inventor cada tipo de bloque posee diferentes formas y colores.

Detección de errores: En Acceleo se cuenta con detección de errores en tiempo de compilación, en Rhapsody se ven errores y advertencias durante la comprobación del modelo en la solapa Check Model de la ventana Output. Mientras que en App Inventor 1 aparece un icono de advertencia de color amarillo en la esquina izquierda de todo bloque que le falte enchufar algún componente y en App Inventor 2 aparece un signo de exclamación en el bloque con error, además de un contador de advertencias y errores en la vista del editor.

Plegado del código o modelo: El editor Acceleo dispone de plegado de código, mientras que en App Inventor se puede tener más espacio disponible en la pantalla al condensar un grupo de bloques. En un diagrama de Rhapsody se pueden configurar las operaciones, atributos, etc. para que aparezcan en pantalla a través del cuadro de diálogo "Opciones de visualización".

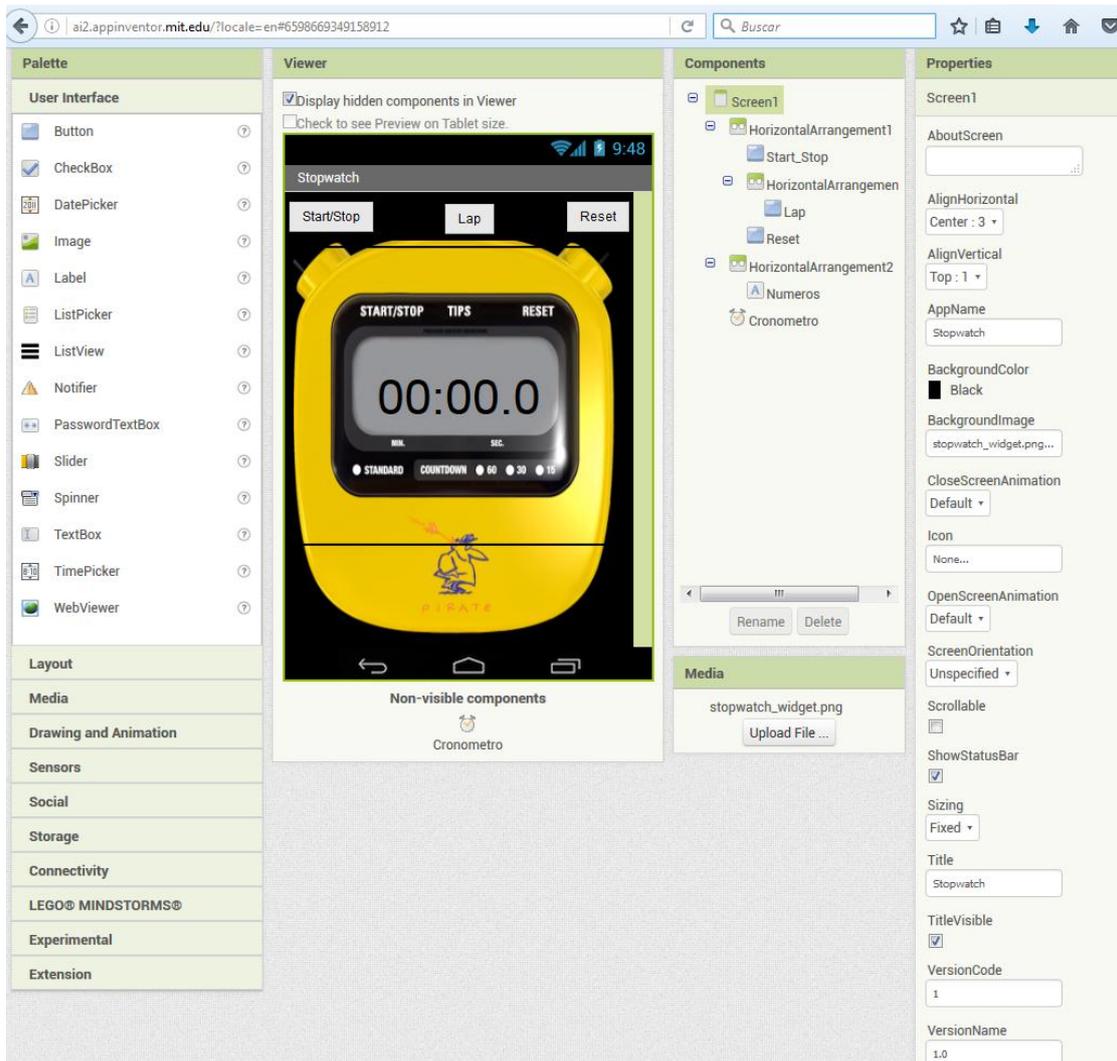
7.2 Ejemplo práctico concreto "Stopwatch"

Para evidenciar claramente las diferencias a continuación se tomó un ejemplo concreto donde se implementa un cronómetro y se generó el código Android para cada una de las herramientas.

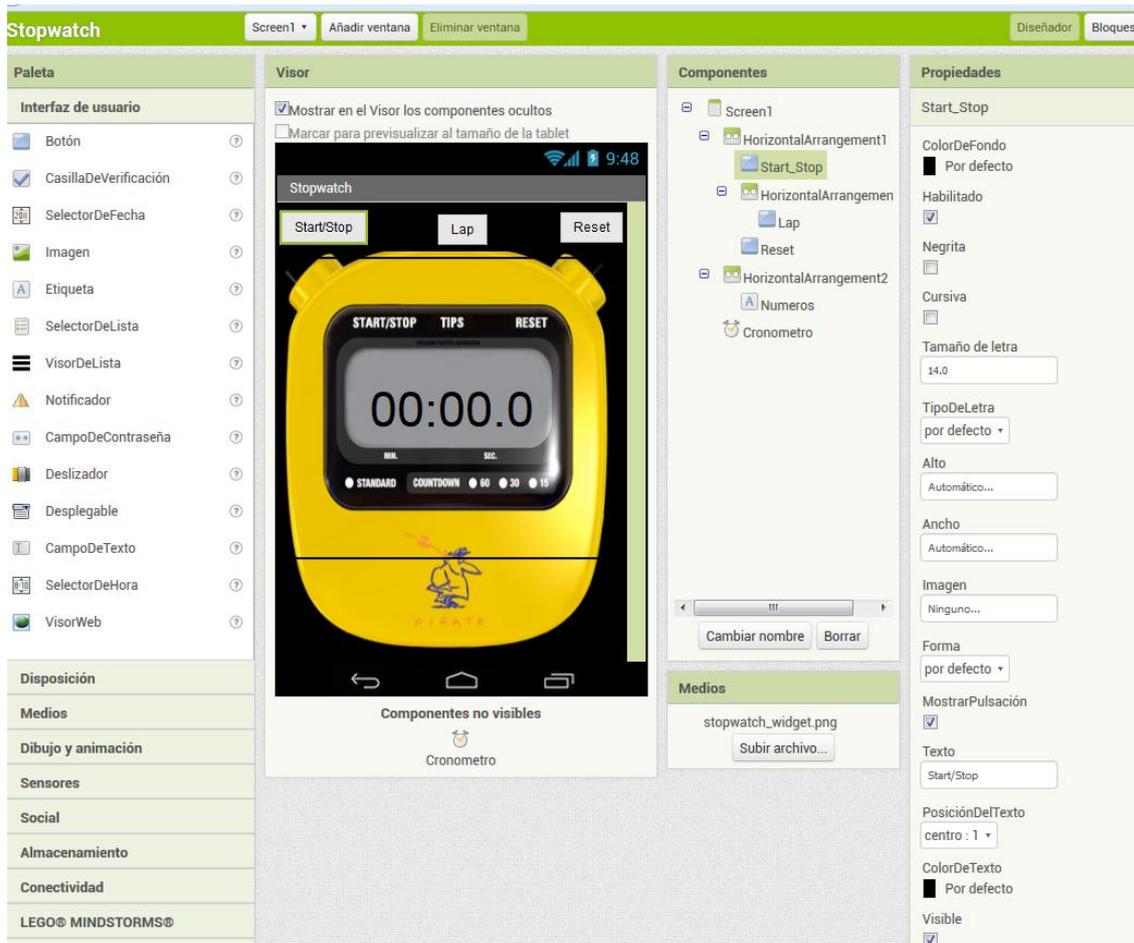
7.2.1 Implementación con App Inventor 2

Configuración en el Diseñador

La aplicación Stopwatch tendrá un componente Screen (Screen1) a la que se le asignarán las propiedades `AppName` -> `Stopwatch`, `BackgroundColor` -> `Black`, `BackgroundImage` -> `stopwatch_widget.png`, `VersionCode` -> `1`, etc., como se muestra en la siguiente figura.



Además se tienen que agregar los componentes visuales a la aplicación. Para lograr esto: desde la paleta básica, arrastre y suelte por ejemplo el componente Button en el Visor. Luego se le podrá asignar un nombre "Start_Stop" y desde el panel de Propiedades, se asignará el tamaño, color y tipo de letra, texto "Start/Stop", etc.



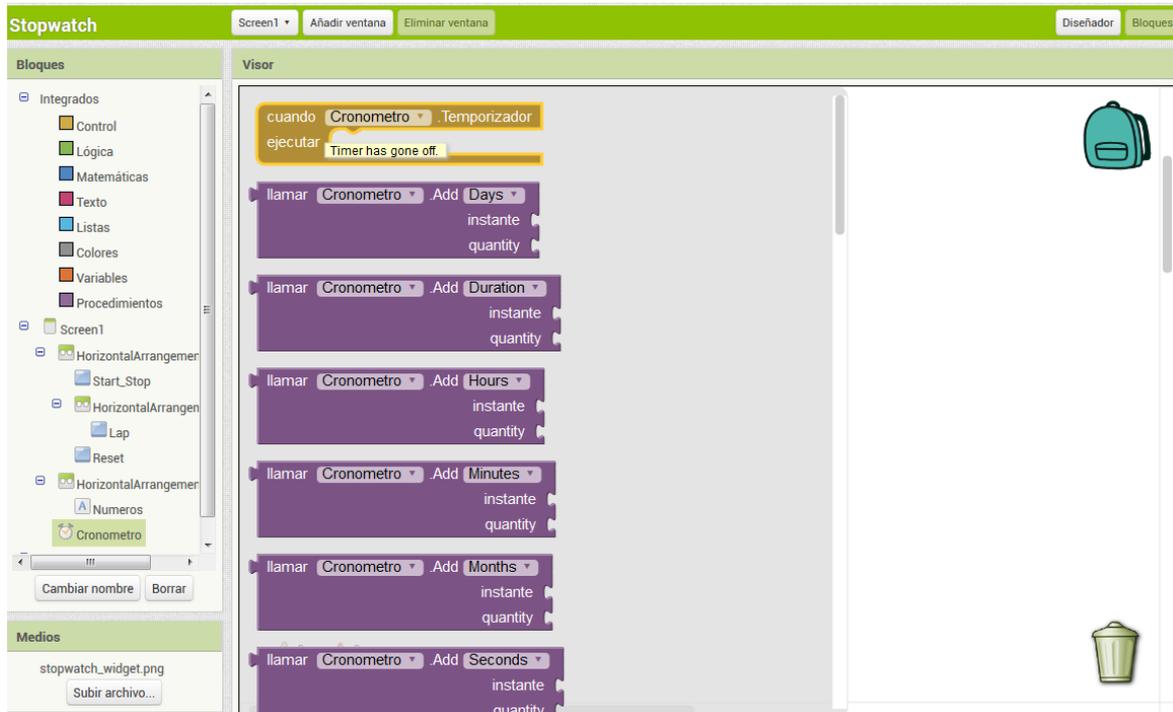
Luego se debería agregar y realizar el mismo procedimiento para todos los componentes de la aplicación, en este caso, HorizontalArrangement, Button, Label y el Clock.

Programación con el editor de bloques

Hasta ahora se vio la organización de la pantalla y los componentes de su aplicación en el *Diseñador*. Para empezar a programar el comportamiento de la aplicación, es necesario ir al *Editor de bloques*.

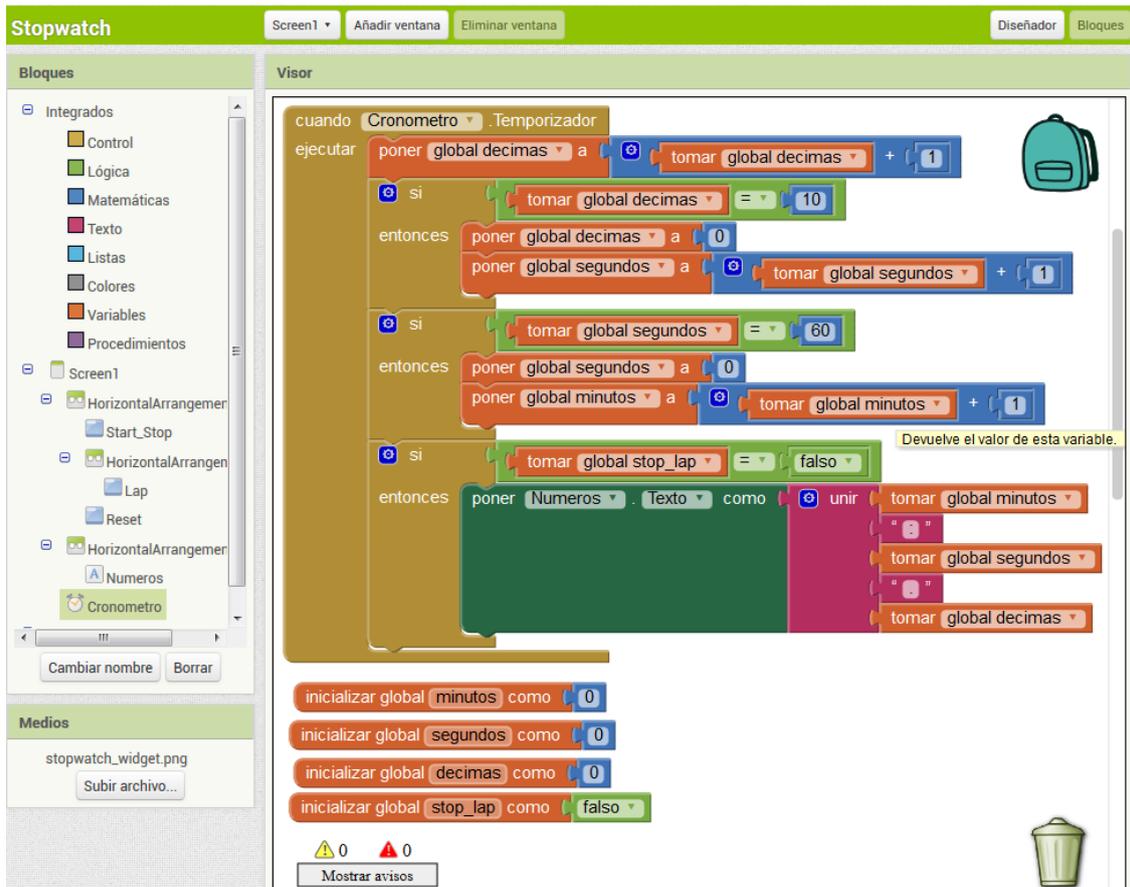
Implementar el comportamiento del cronometro

Paso 1. Bajo la paleta Bloques en el lado izquierdo del Editor de bloques, haga clic en el cajón Cronometro para abrirlo. Arrastre y suelte el bloque Cronometro.Temporizador en el área de trabajo (la zona abierta a la derecha).

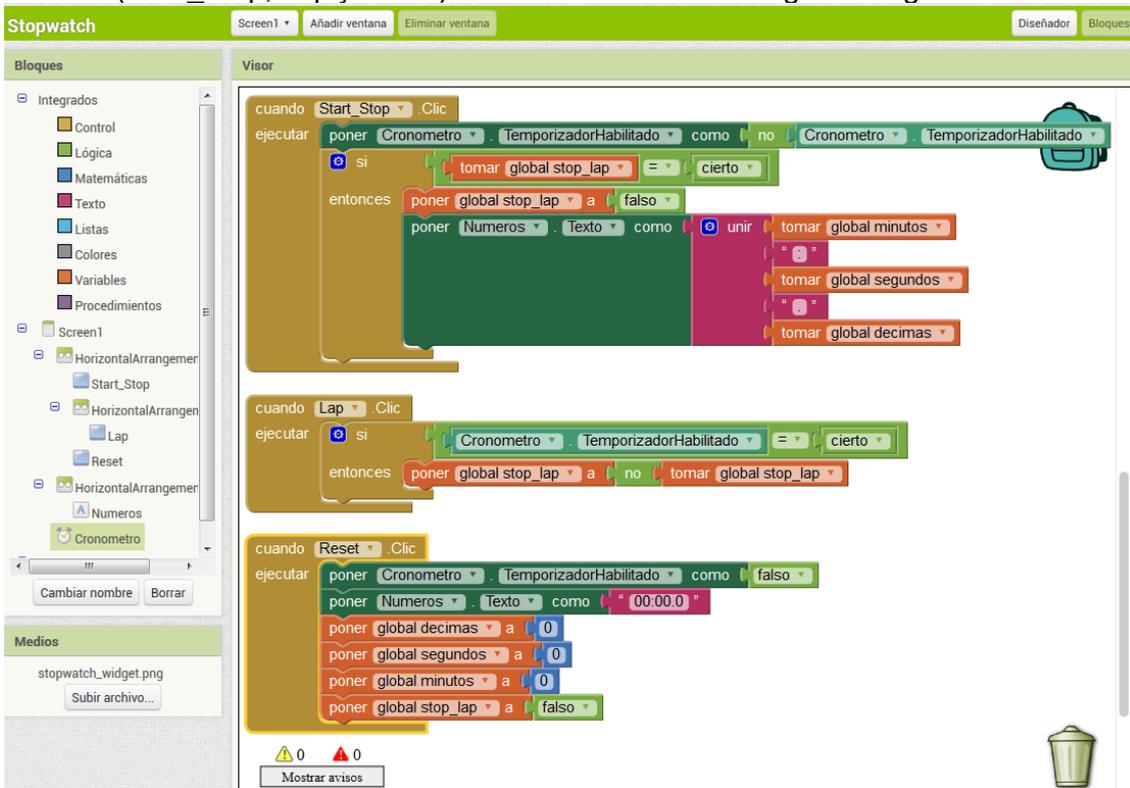


Esos bloques verdes se denominan bloques **de control de eventos**. Los bloques de control de eventos especifican cómo el teléfono debe responder a ciertos eventos: un botón se ha pulsado, el teléfono está siendo sacudido, el usuario está arrastrando su dedo sobre un lienzo, etc. Los bloques de control de eventos son de color verde y el uso de la palabra *cuando*. Por ejemplo, *cuando Cronometro.Temporizador* es un controlador de eventos.

Paso 2. Una vez elegido el bloque, debemos implementar su comportamiento. Haga clic en el bloque que necesita, arrástrelo y conéctelo a la sección "ejecutar" del bloque *cuando Cronometro.Temporizador*. Los bloques se conectan entre sí como piezas de rompecabezas y se puede oír un sonido de clic cuando se conecten. A continuación se puede observar la implementación completa y la declaración e inicialización de las variables globales necesarias.



Siguiendo el mismo procedimiento se debería implementar el comportamiento de los botones (Start_Stop, Lap y Reset) como se muestra en la siguiente figura.



7.2.2 Implementación con Rational Rhapsody

Crear proyecto Rhapsody

- a) En el Package Explorer, click-derecho, **New** → **Rhapsody Project**.
- b) En el wizard, elija el nombre, lenguaje Java, tipo de proyecto Android, el path donde ubicarlo y el botón finish.
Además del Package Explorer el nuevo proyecto aparecerá en el Model Browser.

Asociarlo a un nuevo proyecto Android

- a) En el Model Browser, abra el árbol en “nombre del nuevo proyecto” → Components → DefaultComponent → Eclipse Configurations → DefaultConfig.
- b) En DefaultConfig click-derecho, **Create IDE Project**.
- c) Cuando aparezca la ventana Asistente para configuración de Rhapsody, elija **New Project** y finish.
- d) En el cuadro de diálogo New Project, seleccione la opción **Android Application Project** y complete todos los campos requeridos del wizard.

El Package Explorer mostrará tanto el proyecto de Rational Rhapsody y el proyecto Android Eclipse, mientras que en el Model Browser se observa que el proyecto Rational Rhapsody ahora contiene lo siguiente:

- Un componente con el estereotipo AndroidComponent.
- Visualización de la biblioteca Android que incluye un diagrama de modelo de objetos.
- El perfil AndroidProfile.

Nota: el archivo R.java no se crea automáticamente, porque no compilan los archivos styles.xml. Para solucionarlo se debe cambiar cada “Theme.AppCompat.Light” por “android:Theme.Light” y cada “Theme.AppCompat.Light.DarkActionBar” por “android:Theme.Holo.Light.DarkActionBar”.

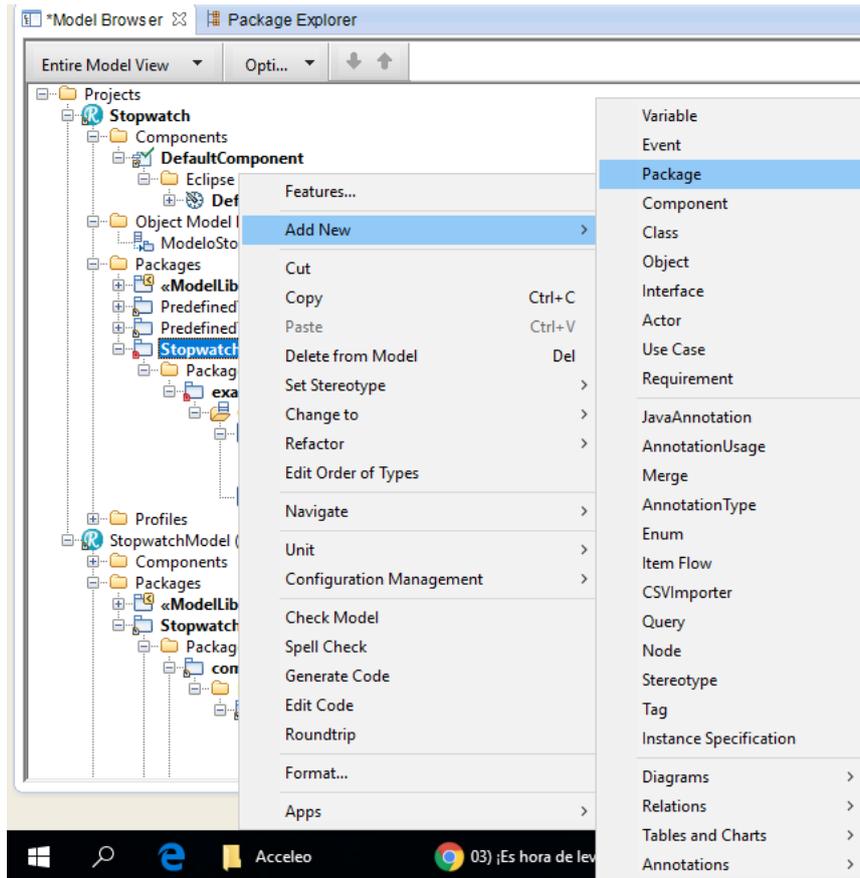
Para poder modificar el proyecto Rhapsody creado, se lo debe setear como el proyecto activo. Para eso desde el Model Browser, click derecho en el proyecto Rhapsody y darle a Set as Active Project.

Ahora se puede comenzar a agregar elementos al proyecto Android.

Agregando elementos al modelo Rational Rhapsody

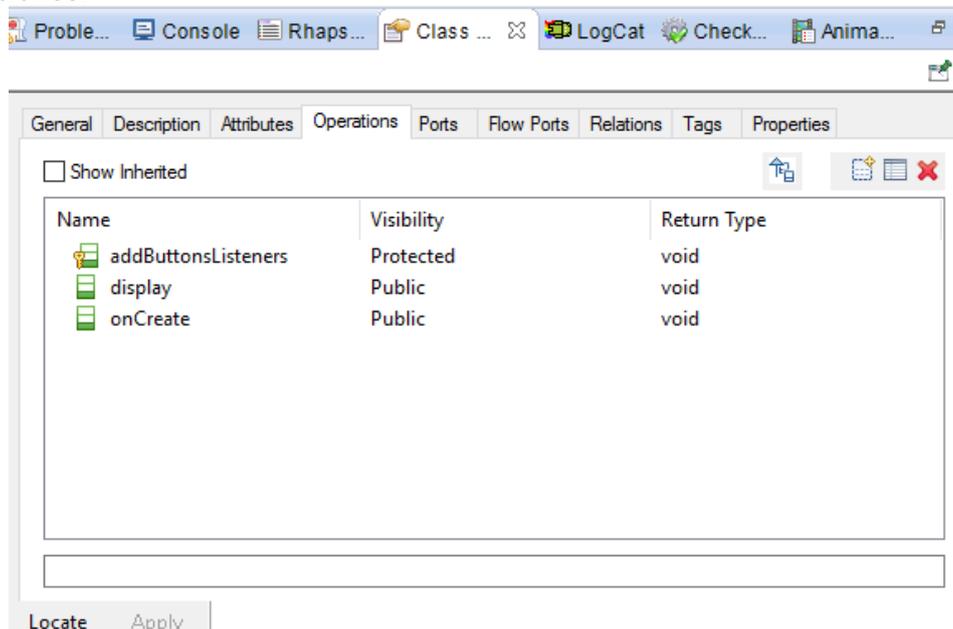
Usando la perspectiva de Modelado de Rhapsody en Eclipse, se pueden utilizar los modelos del menú expandible para añadir elementos a tu modelo.
Procedimiento:

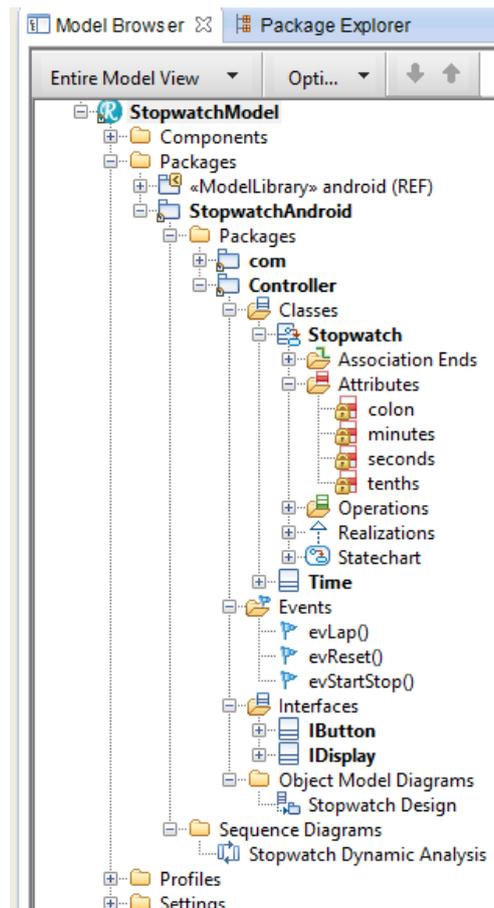
1. Con Eclipse Modelo Browser abierto, haga clic derecho en un elemento para el que desea agregar un elemento.
2. Seleccione **Add New** (tipo de elemento) en el menú en función del tipo de elemento seleccionado. Por ejemplo, **Add New > Package**.



Bajo un paquete se pueden crear varios tipos de elementos como diagramas, relaciones, clases, dependencias, etc.

Para modificar un elemento, puede editar el nombre, descripción, tipo y código de implementación mediante la ventana **Features**, que se muestra a continuación. En este caso se muestran las características de una clase, más precisamente sus operaciones.





Modelo creado por completo

Una vez que tenemos el modelo completo podremos generar y compilar el código Rational Rhapsody, como se muestra en la sección 5.12 y ejecutar la aplicación Android generada.

7.2.3 Implementación con Acceleo

Para crear el metamodelo Android se usará el Framework Xtext. En base a este metamodelo se creará el modelo de la aplicación, en el cual se basará y será leído por Acceleo para implementar el código de generación.

Xtext: breve descripción

Xtext es un framework de código abierto para el desarrollo de lenguajes de programación y lenguajes DSL.

A diferencia de los generadores de analizador sintácticos estándar, Xtext no sólo genera un analizador, sino también un modelo de clase para el árbol de sintaxis y un completo y personalizable IDE basado en Eclipse.

Xtext se está desarrollando en el proyecto Eclipse como parte del proyecto Eclipse Modeling Framework y está disponible bajo la Licencia Pública Eclipse.

Para especificar un lenguaje, un usuario tiene que escribir una gramática en el lenguaje de la gramática Xtext.

Instalación y configuración del software

- Instalar Eclipse.
- Instalar el SDK de Android.
- Agregar en Eclipse el plugin ADT para vincular con el SDK de Android.
- Agregar en Eclipse el plugin para trabajar con Acceleo.

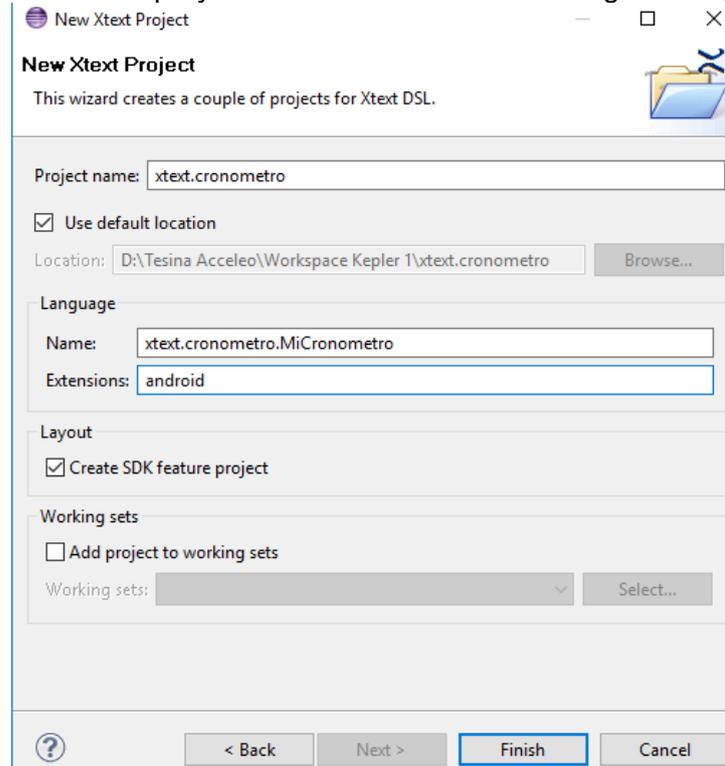
- Instalar plugin "xtext SDK" en el sitio de actualización de Eclipse, por ejemplo desde <http://download.eclipse.org/releases/helios>, en la categoría "Modeling".

Proyecto Xtext

1. Creación del proyecto xtext

- a) Abrir el menú File > New > Project...
- b) Elija xtext > Project xtext.

Y llenar las propiedades del proyecto como se muestra en la siguiente figura:



2. Modificamos el archivo de gramática

Al archivo de gramática MiCronometro.xtext se le agrega el concepto de Application e introducimos algunos conceptos más a nuestra DSL: Screan, Layout, Chronometer, View, Widget, Action, etc.

El DSL quedará como se muestra a continuación:

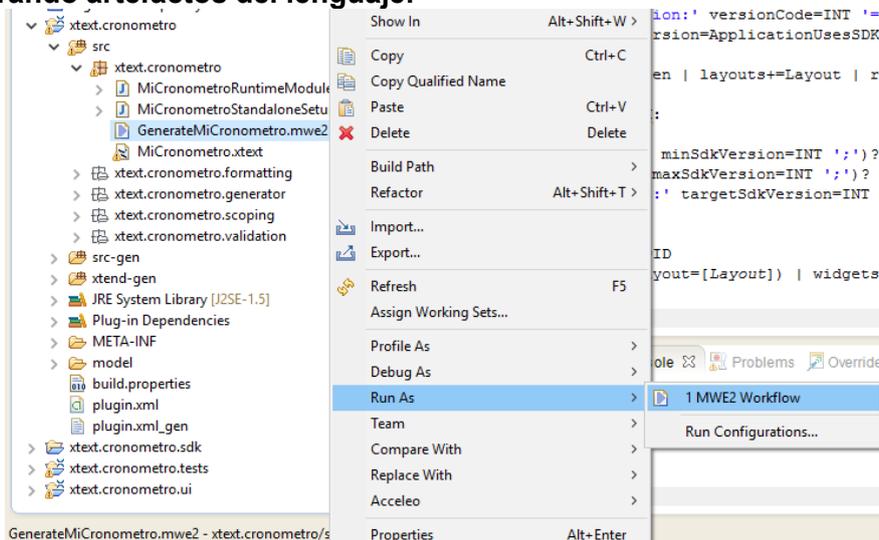
```

1 grammar xtext.cronometro.MiCronometro with org.eclipse.xtext.common.Terminals
2 generate miCronometro "http://www.cronometro.xtext/MiCronometro"
3
4 Model: applications+=Application*;
5
6 Application:
7     'application' name=STRING
8     '=>' packageName=PackageName
9     (
10         ('version:' versionCode=INT '=>' versionName=STRING)?
11         & (sdkVersion=ApplicationUsesSDK)?
12     )
13     (screens+=Screen | layouts+=Layout | resources+=Resource | classes+=Class)+;
14
15 ApplicationUsesSDK:
16     'sdk:'
17     '{' ( ('min:' minSdkVersion=INT ';')?
18         & ('max:' maxSdkVersion=INT ';')?
19         & ('target:' targetSdkVersion=INT ';')? ) '}' ;
20
21 Screen:
22     'screen' name=ID
23     '{' (('show' layout=[Layout]) | widgets=ViewCollection) '}' ;
24
25 Class: 'class' name=ID '{' '}' ;
26
27 PackageName: ID ('.' ID)*;
28
29 Layout:|
30     (isLinear?='linear')?
31     'layout' name=ID
32     '{'
33     (

```

MiCronometro.xtext

3. Generando artefactos del lenguaje.

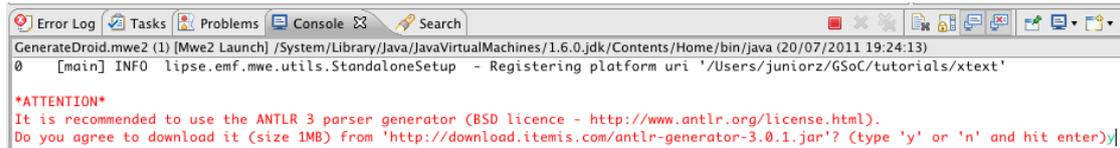


Con el fin de utilizar el DSL que se necesita para generar los componentes del lenguaje.

- a) Haga clic derecho en el archivo de flujo de trabajo *GenerateMiCronometro.mwe2*.
- b) Desde el menú contextual, seleccione "Run As > MWE2 Workflow".

4. Instalar el generador de analizadores sintácticos ANTLR 3

Ir a la *consola*, escriba 'y' y presionar "Enter". Ésto permite descargar el generador de analizadores sintácticos basados en ANTLR 3.



```

GenerateDroid.mwe2 (1) [Mwe2 Launch] /System/Library/Java/JavaVirtualMachines/1.6.0.jdk/Contents/Home/bin/java (20/07/2011 19:24:13)
0 [main] INFO lipse.emf.mwe.utils.StandaloneSetup - Registering platform uri '/Users/juniorz/GSoC/tutorials/xtext'

*ATTENTION*
It is recommended to use the ANTLR 3 parser generator (BSD licence - http://www.antlr.org/license.html).
Do you agree to download it (size 1MB) from 'http://download.itemis.com/antlr-generator-3.0.1.jar'? (type 'y' or 'n' and hit enter)|

```

El flujo de trabajo va a generar todos los archivos necesarios para crear un plugin de Eclipse con un editor de texto a tu DSL. Cada vez que haga algún cambio en el archivo de gramática (MiCronometro.xtext) debe regenerar los artefactos del lenguaje.

Exportar e instalar la DSL como un plugin de Eclipse

Antes de crear el proyecto Acceleo debe exportar e instalar el plugin que contiene su metamodelo DSL. Por lo tanto, debe exportar e instalar el plugin xtext (xtext.cronometro) generado.

Exportar la DSL

1. File > Export...
2. Elija Plug-in Development > Deployable plug-ins and fragments.
3. Seleccionar los proyectos xtext.cronometro, xtext.cronometro.tests y xtext.cronometro.ui
4. Elija una carpeta de destino.

Instalando su plugin generado

1. Copie los plugins (xtext.cronometro_1.0.0.jar, xtext.cronometro.tests_1.0.0.jar y xtext.cronometro.ui_1.0.0.jar) generados desde la carpeta elegida en el paso anterior a la carpeta plugins (dentro de su instalación de Eclipse).
2. Reinicie Eclipse.

Diseño proyecto Acceleo

Los módulos Acceleo se dividen en dos sub-paquetes de *org.acceleo.example.droid* situado en la carpeta *src*:

- subpaquete *main*: contiene el módulo principal que será lanzado como una Acceleo Application.
- subpaquete *files*: contiene los módulos restantes, que serán utilizados para generar el código.

Creación del proyecto Acceleo

1. Abrir el menú *File > New > Project...*
2. Elija *Acceleo Model to Text > Acceleo Project*
3. Cambie el Project name a *acceleo.crono*.

Adición de módulos para el proyecto

Ahora vamos a añadir varios módulos de plantilla.

En la sección "Module Files", seleccione *generate* módulo y cambiar sus propiedades utilizando los campos en el lado derecho de la ventana.

1. En la sección "Metamodel URIs"
 - a) Haga clic en el botón +
2. Seleccione el paquete URI *http://www.cronometro.xtext/MiCronometro*
3. Cambie las propiedades del módulo seleccionado utilizando los campos de la parte derecha de la ventana:
 - a) Marque "*Initalize Contents*" y seleccione "*Copy example content*"
 - b) Archivo: Busque "*AndroidManifest.xml*"
 - c) Nombre del módulo: "*application*"
 - d) Nombre de la plantilla: "*generateApplication*"
 - e) Tipo: "*Aplication*"

- f) Marque la opción "*Template*"
- g) Marque "*Generate file*"

Se va a crear el módulo *application*. Repita estos pasos para crear los módulos necesarios (*screen.mtl*, *layout.mtl*, *resource.mtl*, *class.mtl*) y el módulo principal generador (*main.mtl*).

Agregar la dependencia Metamodel

1. Abra el archivo MANIFEST.MF dentro de la carpeta META-INF.
2. Seleccione la ficha Dependencies y agregar el plugin que contiene su metamodelo DSL (*xtext.cronometro*) a la lista de plugins necesarios.
3. Guarde el archivo.

Nota: Si tiene proyectos Xtext abiertos en el workspace debe cerrarlos y refrescar el proyecto Acceleo para que este tome los cambios recientemente ingresados desde las dependencias.

Creación del módulo de plantilla

MTL define las transformaciones utilizando plantillas. Así pues, si queremos generar algo, tenemos que ponerlo en una plantilla.

El proceso de generación de Acceleo comienza invocando una plantilla **Main**, que se encuentra en el archivo *main.mtl* y contiene un *[comment @main/]* indicando el punto de partida. El elemento raíz de nuestra DSL MiCronometro es una *Application* y así nuestra plantilla main debe generar todo el código de un objeto Application (AndroidManifest.xml), además de Screens (MainActivity.java), Layout (activity_main.xml), Resource (nameApplication.xml y strings.xml) y Classes (CronometroPropio.java).

```

1  [comment encoding = UTF-8 /]
2  [module main('http://www.cronometro.xtext/MiCronometro')]
3  [import acceleo::crono::files::application /]
4  [import acceleo::crono::files::screen /]
5  [import acceleo::crono::files::layout /]
6  [import acceleo::crono::files::resource /]
7  [import acceleo::crono::files::class /]
8
9  [template public main(aApplication : Application)]
10 [comment @main/]
11   [for (aScreen : Screen | aApplication.screens)]
12     [aScreen.generateScreen(aApplication.classes->first())/]
13   [/for]
14   [generateAndroidManifest(aApplication)/]
15   [for (aLayout : Layout | aApplication.layouts)]
16     [aLayout.generateLayout()/]
17   [/for]
18   [aApplication.generateResourceFile()/]
19   [for (aClass : Class | aApplication.classes)]
20     [aClass.generateClassCronometroPropio()/]
21   [/for]
22 [/template]
23

```

main.mtl

Hay algunos puntos importantes en el código anterior:

- *[import acceleo::crono::files::application /]*
Aquí estamos importando plantillas (entre otras cosas) en el módulo actual.
- *[generateAndroidManifest(aApplication)/]*
Aquí estamos llamando una plantilla. MTL soporta tanto la sintaxis *generateAndroidManifest(aApplication)* como *aApplication.generateAndroidManifest()*.

De esta manera delegamos la generación del archivo manifest a otra plantilla.

```

1 [comment encoding = UTF-8 /]
2 [module application('http://www.cronometro.xtext/MiCronometro')]
3
4 [template public generateAndroidManifest(aApplication : Application)]
5   [file ('AndroidManifest.xml', false, 'UTF-8')]
6 <?xml version="1.0" encoding="utf-8"?>
7 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
8     package="[aApplication.packageName]"
9     android:installLocation="preferExternal"
10
11 -- Mas Codificación Acceleo
12
13 </manifest>
14   [//file]
15 [//template]

```

application.mtl

Creación del proyecto Android

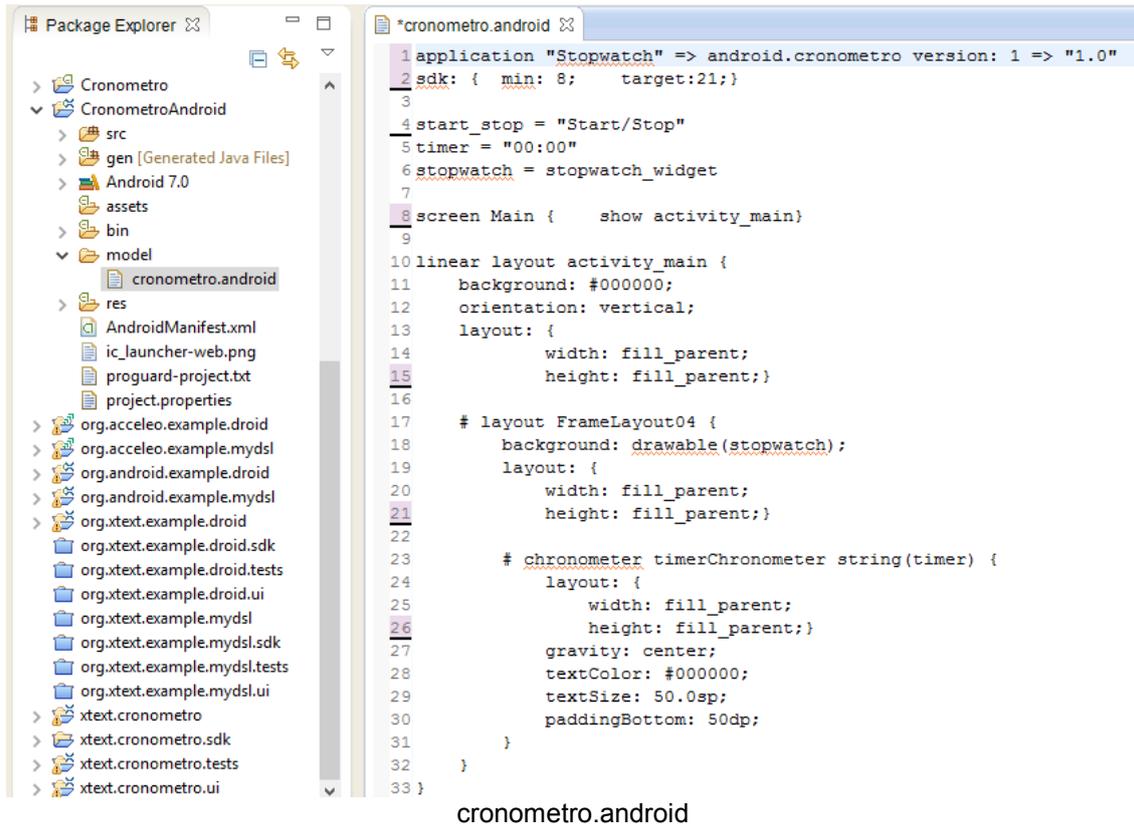
Este proyecto será una guía en el proceso de creación del generador y una prueba del mismo.

1. Abrir el menú *File > New > Project...*
2. Elija *Android > Android Application Project*
3. Llenar las Opciones de proyecto de la siguiente manera:
 - a) Nombre de la aplicación: *Stopwatch*
 - b) Nombre del proyecto y el nombre del paquete: *CronometroAndroid*, SDK mínima, Target SDK y compilación con: *API 21: Android 4.X (L Preview)*
 - c) Crear Actividad: *MainActivity*

Creación de un archivo modelo MiCronometro

Con el fin de probar el generador, necesitamos tener algún archivo fuente escrito en nuestro lenguaje DSL.

1. Haga clic derecho en el *CronometroAndroid*
2. Elija *New > Folder*
3. Crear una carpeta con el nombre *model*
4. Haga clic derecho en la carpeta creada *model*
5. Elija *New > File*
6. Cree un archivo denominado *cronometro.android*
7. Pegue el código siguiente en el contenido del archivo:



Generación de código

Con el fin de ejecutar la generación del código:

1. Haga clic derecho en el paquete *acceleo.crono.main*
2. Elija *Run As > Launch Acceleo Application*
La primera vez se verá la ventana **Run Configurations**.
3. Establecer la propiedad *Model* hasta el archivo de origen DSL (*cronometro.android*)
4. Establecer la propiedad *Target* a la ruta raíz de su proyecto Android (*CronometroAndroid*).
5. Chequear la opción *Contribute traceability information to Result View*.
6. Haga clic en *Apply* y luego en *Run*.

Probando el código generado

Una vez que generemos algún código, éste es el procedimiento para probar la aplicación para Android:

1. Haga clic derecho en el proyecto Android *CronometroAndroid*
2. Elija *Run As > Android Application*

La AVD se pondrá en marcha y la aplicación se cargará. La aplicación se llamará "Stopwatch".

7.2.4 Conclusiones Prácticas

Al desarrollar la aplicación en App Inventor 2 resultó fácil y rápida la realización tanto de su interfaz gráfica como la implementación del comportamiento mediante sus bloques de construcción. Aunque a la hora de querer centrar los números digitales dentro de la figura del cronómetro, se encontraron ciertas dificultades debido a que no se tiene acceso al XML resultante.

La experiencia con Rational Rhapsody fue un tanto más compleja, ya que además de la implementación de clases, relaciones y eventos, se deben combinar diagramas de

estado, modelo de objetos y de secuencia, entre otros, para poder tener una aplicación en funcionamiento.

Al crear la aplicación con Acceleo, se necesitó usar el Framework Xtext para la creación del metamodelo Android, el cual se podría ampliar y estandarizar para ser utilizado para implementar todo tipo de aplicaciones Android. Una vez que se obtuvo un modelo adecuado a partir del metamodelo Android, la implementación de la generación del código Android con Acceleo resultó fácil y ágil, al tener integración con el entorno Eclipse.

8 Conclusiones y trabajos futuros

8.1 Conclusiones

A lo largo de la historia de la programación de computadoras se ha evolucionado con la creación de herramientas que nos permiten abstraernos del mundo binario.

Antiguamente se programaba en lenguajes de bajo nivel o ensambladores los cuales son dependientes de la arquitectura de la computadora. Hoy en día se usan los lenguajes de alto nivel, que si bien pierden velocidad al no tener un acceso directo a los recursos de la máquina, ganan en abstracción de la arquitectura siendo en su mayoría portátiles (independientes de la plataforma y/o sistemas operativos) y teniendo mayor facilidad de uso.

Actualmente siguiendo el paradigma de MDD, se están desarrollando nuevas herramientas para abstraerse un paso más, es decir, que se busca generar sistemas que sean independientes de cualquier tecnología o lenguaje de implementación, ignorando hardware, sistemas operativos y lenguajes de programación. Dentro de estas herramientas se encuentran las descritas en esta tesina las cuales tienen en común estar destinadas a la creación de aplicaciones que ejecuten sobre un sistema operativo Android.

Al adentrarme en este mundo, además de la escasa cantidad de herramientas que están destinadas a generar código para el moderno y frecuentemente utilizado SO Android, pude observar que las diferentes herramientas ofrecen insuficiente información y ejemplos de uso. Lo que da la pauta de lo mucho que queda para investigar y dedicarle si es que se quieren aplicar este tipo de herramientas en el mercado actual.

Convencer a los empresarios y desarrolladores de software será una tarea que demandará años, hasta tanto no se confíe en la fiabilidad de la automatización de la generación de código de los nuevos productos, los cuales tendrán que usar estándares industriales, mantener una consistencia y robustez en las transformaciones entre los modelos y el código generado, demostrar un aumento en la velocidad y reducir los costos de producción. Además se debería difundir e informar ampliamente el desarrollo de estas nuevas tecnologías, extendiéndolas al ámbito educativo, para que desde épocas tempranas los estudiantes dispongan de un abundante conocimiento sobre este paradigma sus características y el uso de estas herramientas.

En la actualidad, al desarrollar una aplicación que se ejecute para el SO Android muy escasamente se aplican metodologías dirigidas por modelos, pero se puede utilizar App Inventor para aplicaciones simples, aprovechando su facilidad de uso al estar dirigida a personas que no están familiarizadas con la programación informática y el hecho de contar con bloques para un gran número de las necesidades básicas que se realizan con un dispositivo Android. Además de contar con Google Play, el centro de distribución de aplicaciones para Android donde cualquier usuario puede compartir sus creaciones libremente.

Si se desea desarrollar una amplia y compleja aplicación que necesite de la integración del modelo, el código y la depuración en un entorno de desarrollo, como Eclipse, se debería utilizar productos del estilo de Rational Rhapsody o Acceleo.

Principalmente para sistemas embebidos de tiempo real y porque no para interfaces de usuario en PC, se podría utilizar Rational Rhapsody, el cual gracias a la simulación de los diagramas UML, permite visualizar el modelo, chequearlo estáticamente, para mejorar su consistencia y depurar el diseño a medida que se crea, siempre manteniendo la sincronización de los cambios en el modelo o el código. Además, se trata de una herramienta muy completa y estable que dispone de facilidades en su uso para el modificado y creación de nuevos diagramas aunque el entorno grafico no sea muy amigable para el usuario y no sea un software libre.

Acceleo fue diseñado para los desarrolladores de las tecnologías MDA con el objetivo de incrementar su productividad de desarrollo de software. Con Acceleo se garantiza un tiempo de ejecución relativamente corto pero supone dificultades con las transformaciones de modelos ya que provee un listado finito de posibles modelos con los que es compatible y con los que nos permite trabajar de forma rápida en sus plantillas de transformación. Carece de esa versatilidad, pero no deja de ser un generador muy interesante para transformar a texto modelos UML, EMF, XMI y JAVAXMI. Además posee una integración completa tanto con Eclipse como con el framework EMF, eficacia en proyectos a gran escala, depuración, trazabilidad, generación incremental, fácil actualización y manejo de templates, coloreo de sintaxis, autocompletado y se compila en tiempo real con detección de errores. Asimismo, los modelos de entradas son independientes de los lenguajes de salida utilizados para generar el código. Nos permite realizar llamadas al sistema y librerías externas a través del lenguaje Java. Claramente, requiere poseer un conocimiento previo tanto en Java, como en modelado (por ejemplo, UML) y en la plataforma Eclipse. En resumen es una herramienta muy completa que posee el punto fuerte de basarse en un estándar internacional de la OMG.

8.2 Trabajos futuros

A raíz de los temas como así también de las herramientas descritas y desarrolladas en la presente tesina se derivan los siguientes puntos que pueden ser considerados trabajos futuros:

- Al momento en que se comenzó con esta tesina y se decidió incluir a Rational Rhapsody como una de las herramientas para investigar, la misma se integraba con el entorno Eclipse para la generación de código Android. Actualmente Eclipse ya no tiene soporte por parte de Google y en su lugar se creó un nuevo IDE oficial llamado Android Studio, pero este último no se integra con Rational Rhapsody. El diseño y generación automática de código Android a través de modelos para Android Studio podría ser un futuro trabajo de investigación.
- Al usar App Inventor tenemos una cierta cantidad de bloques. Estos bloques vienen encapsulados y ya listos para ser utilizados en nuestras aplicaciones. Se podría necesitar de una funcionalidad que requiera la utilización de un nuevo tipo de bloque. Un nuevo trabajo sería producir la creación de nuevos bloques o investigar como lo hace Google/MIT con App Inventor.
- En Rational Rhapsody se dispone del modelo y ciertas propiedades de generación que dan como resultado el código fuente Android. La muestra de la relación de trazabilidad faltante entre los elementos mencionados se podría crear.
- Desarrollar y producir algún mecanismo de roundtripping en Acceleo.

Referencias bibliográficas

- Desarrollo de software dirigido por modelos: conceptos teóricos y su aplicación práctica. Claudia Pons; Roxana Giandini; Gabriela Pérez. 1a ed. - La Plata: Universidad Nacional de La Plata, 2010.
- MDD of Android Applications [Online] / Autor: Prezi - <http://prezi.com/iuftagvwwcma/mdd-of-android-applications/>
- Bitreactive [Online] / Autor: Bitreactive AS - <http://bitreactive.com>
- Universidad Noruega de Ciencia y Tecnología [Online] / Autor: NTNU - <https://www.ntnu.edu/>
- Developing Android Applications with Arctis. Stephan Haugsrud - Departamento de Telemática de la Universidad Noruega de Ciencia y Tecnología, Junio de 2010 - [Online] : <http://www.diva-portal.org/smash/get/diva2:348806/FULLTEXT01.pdf>
- EclipseCon Europe [Online] / Autor: Frank Alexander Kraemer [Bitreactive AS] - <http://www.eclipsecon.org/europe2012/sessions/arctis-modeling-reactive-java-components.html>
- Arctis Orientation and Manual. Frank Alexander Kraemer - NTNU Technology Transfer AS, Octubre de 2010 - [Online]: <http://www.item.ntnu.no/~kraemer/arctis/orientation/arctis-orientation-2010-10.pdf>
- MOF based code generation method for android platform. Son, H.S. , Kim, W.Y. , Kim, R.Y.C. - International Journal of Software Engineering and its Applications, 7 (3), pp. 415-426, Mayo de 2013 - [Online]: http://www.sersc.org/journals/IJSEIA/vol7_no3_2013/37.pdf
- Mobia Modeler: Easing the creation process of mobile applications for non-technical users. Balagtas-Fernandez, F., Tafelmayer, M., Hussmann, H. - International Conference on Intelligent User Interfaces, Proceedings IUI, pp. 269-272, Noviembre de 2010 - [Online]: <https://www.medien.ifi.lmu.de/pubdb/publications/pub/balagtasfernandez2010iui/balagtasfernandez2010iui.pdf>
- Wikipedia App Inventor [Online] / Autor: Comunidad Wikipedia - https://es.wikipedia.org/wiki/App_Inventor
- App Inventor [Online] / Autor: Massachusetts Institute of Technology - <http://appinventor.mit.edu>

- Aprendé App Inventor [Online] / Autor: Google - <https://sites.google.com/site/aprendeappinventor>
- IBM Knowledge Center [Online] / Autor: IBM - http://www.ibm.com/support/knowledgecenter/SSB2MU/rhapsody_family_welcome.html
- Rational Rhapsody User Guide. IBM – 2009 [Online]: ftp://public.dhe.ibm.com/software/rationalsdp/documentation/product_doc/Rhapsody/version_7-5/UserGuide.pdf
- Rational Rhapsody API Reference Manual. IBM – 2009 [Online]: <http://www-01.ibm.com/support/docview.wss?uid=swg27017178&aid=1>
- Java Tutorial for Rational Rhapsody. IBM – 2009 [Online]: http://ai.ia.agh.edu.pl/wiki/_media/pl:dydaktyka:miw:2010:telelogic:tutorialjava.pdf
- Instalar el SDK de Android y vincularlo con Eclipse [Online] / Autor: Ramón Invarato - <http://jarroba.com/instalar-el-sdk-de-android-y-vincularlo-con-eclipse/>
- Android Developers [Online] / Autor: Google - <https://developer.android.com/index.html>
- Rational Rhapsody Developer [Online] / Autor: IBM - <http://www-03.ibm.com/software/products/ar/es/ratirhap/>
- Sistema avanzado de prototipado rápido para control en exoesqueletos y dispositivos mecatrónicos. Antonio Flores Caballero - Departamento de ingeniería de sistemas y automática, Universidad Carlos III de Madrid, Leganés, 15 de Diciembre 2014 - [Online]: http://e-archivo.uc3m.es/bitstream/handle/10016/20665/tesis_antonio_flores_caballero_2014.pdf
- Acceleo [Online] / Autor: Fundación Eclipse - <http://www.eclipse.org/acceleo/>
- Acceleo [Online] / Autor: Fundación Eclipse - <http://projects.eclipse.org/projects/modeling.m2t.acceleo>
- Obeo Network [Online] / Autor: Obeo - <http://www.obeonetwork.com/>
- Stephane Begaudeau [Online] / Autor: Stephane Begaudeau - <http://sbegaudeau.tumblr.com/>
- Eclipse Documentation [Online] / Autor: Fundación Eclipse - <http://help.eclipse.org/>
- Wikipedia Acceleo [Online] / Autor: Comunidad Wikipedia - <http://en.wikipedia.org/wiki/Acceleo>
- Ingeniería dirigida a modelos. Sistemas de transformación modelo a texto. Complemento de refactorización de código C++N a C++11 para Eclipse. Alejandro

Tovar Moreno – Departamento de informática, Universidad Carlos III de Madrid,
Colmenarejo, 25 de Junio de 2013 - [Online]: [http://e-
archivo.uc3m.es/bitstream/handle/10016/19209/TFG Alejandro Tovar Moreno.pdf](http://e-archivo.uc3m.es/bitstream/handle/10016/19209/TFG_Alejandro_Tovar_Moreno.pdf)

- Android code generation with Acceleo [Online] / Autor: GitHub - <https://github.com/eclipse-soc/amalgamation-examples-acceleo/wiki/Android-code-generation-with-Acceleo>
- MOF Model To Text Transformation Language™ (MOFM2T™), 1.0 [Online] / Autor: Object Management Group - <http://www.omg.org/spec/MOFM2T/1.0/>

Apéndice: Glosario

MOFM2T es un estándar que define un lenguaje de generación de código llamado MTL (modelo a texto lenguaje de transformación). El lenguaje utilizado en Acceleo es una implementación de esta norma.

EMF: Acceleo se basa en el proyecto EMF que es un framework de modelado y generación automática de código para construir herramientas y otras aplicaciones a partir de modelos de datos estructurados. Gracias a este uso, Acceleo puede generar código a partir de modelos gráficos creados con GMF o representación textual creada con xtext.

OCL es un lenguaje estándar de la OMG para navegar en modelos y definir restricciones sobre los elementos de un modelo Ecore para luego evaluarlas y determinar si el modelo está correctamente formado.

Es un lenguaje libre de efectos laterales. Esto significa que ninguna expresión OCL puede modificar elementos del modelo.

Se usa frecuentemente para especificar pre y post condiciones para las operaciones.

El lenguaje Acceleo es un súper conjunto del lenguaje OCL y como tal el proyecto Acceleo depende de la implementación del lenguaje OCL realizada en la fundación Eclipse.

MOF (Meta-Object Facility), lenguaje estándar de transformación de modelo a texto de la OMG.

Ecore es un lenguaje basado en EMOF que es parte de la especificación MOF. El meta-metamodelo Ecore es usado por EMF para la definición de metamodelos. Los metamodelos y modelos usados por EMF se representan con documentos XML.

DSL (Domain Specific Languages) es cualquier lenguaje que esté especializado en modelar o resolver un conjunto específico de problemas. Este conjunto específico de problemas es el llamado dominio de aplicación o de negocio. La mayor parte de los lenguajes de programación no se pueden considerar DSL ya que no están diseñados para resolver un conjunto específico de problemas, sino para resolver cualquier tipo de problema.

UML Lenguaje de modelado unificado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, respaldado por el OMG.

ANSI Instituto Nacional Americano de Estándares es una organización sin fines de lucro que supervisa el desarrollo de estándares para productos, servicios, procesos y sistemas en los Estados Unidos.

SysML Lenguaje de especificación de sistemas, el cual es un subconjunto ampliado de UML 2.0, y un estándar de la OMG, construido por una comunidad de ingenieros de sistemas.