Máster Universitario de Investigación en
Ingeniería de Software y Sistemas Informáticos

Generador Automático de Código ABAP IV para sistemas SAP R3

Trabajo Fin de Máster
Itinerario Ingeniería de Software (Código 31105128)

Alumno: David Rueda Barrón
Director: Ismael Abad Cardiel

Curso 2015/2016
Convocatoria Septiembre 2016

Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos

ITINERARIO: Ingeniería de Software

CÓDIGO DE ASIGNATURA: 31105128

TÍTULO: Generador Automático de Código ABAP IV para sistemas SAP R3

TIPO DE TRABAJO: Tipo B, trabajo específico propuesto por el alumno

Alumno: David Rueda Barrón
Director: Ismael Abad Cardiel

IMPRESO TFDM05_AUTOR
AUTORIZACIÓN DE PUBLICACIÓN
CON FINES ACADÉMICOS

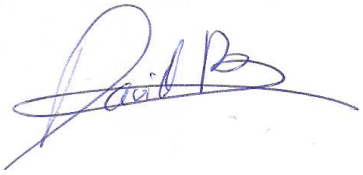
Impreso TFdM05_Autor. Autorización de publicación

y difusión del TFdM para fines académicos

Autorización

Autorizo/amos a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firma del/los Autor/es



Juan del Rosal, 16
28040, Madrid

Tel: 91 398 89 10
Fax: 91 398 89 09

www.issi.uned.es

AGRADECIMIENTOS

Quiero agradecer a mi esposa Sonia Arilla todo su apoyo y estimulo a la hora de realizar este Máster. Gracias por haber estado siempre a mi lado estos años en los que solo tenía tiempo para trabajar y estudiar, por tener siempre una sonrisa y las palabras adecuadas para levantarme el ánimo y continuar hasta el final.

RESUMEN

El presente trabajo aborda el desarrollo de un generador de código automático ABAP IV como posible herramienta para el diseño, implementación y mantenimiento de procesos de negocio y aplicaciones a medida para el sistema de gestión empresarial SAP R3. El diseño, implementación y mantenimiento de las aplicaciones a medida y procesos de negocio suelen ser las tareas que más tiempo y dinero consumen normalmente en un proyecto de implantación del sistema SAP R3. El presente trabajo analiza las posibilidades que ofrece el enfoque de la arquitectura dirigida por modelos frente al desarrollo tradicional de software.

Definiremos un metamodelo a medida que pueda ser usado para el diseño de modelos que representen aplicaciones SAP R3. Estos modelos servirán como modelos de entrada para el generador de código que realizara las trasformaciones necesarias para generar código ABAP IV ejecutable sobre sistemas SAP R3.

El metamodelo será diseñado dentro del marco de la arquitectura dirigida por modelos MDA y siguiendo el estándar de OMG de cuatro niveles capas de modelado. Para el diseño de metamodelo se utilizara el lenguaje de metamodelado Ecore disponible en la herramienta EMF de la plataforma de desarrollo ECLIPSE. Con el lenguaje de modelado Ecore, definiremos los elementos y relaciones de un metamodelo propio. A partir de este metamodelo, se podrá modelar aplicaciones SAP R/3 que sirvan como modelos de entrada un sistema de trasformación de modelo a texto. El trasformador de modelo a texto, diseñado con la herramienta Acceleo para la plataforma Eclipse, generara el código ABAP IV necesario para implementar el modelo de entrada. . Con esta herramienta se espera poder mejorando la productividad, calidad, mantenimiento y seguridad del software para sistemas SAP R3.

Por último, compararemos los resultados obtenidos con el generador de código automático de código ABAP IV con las herramientas disponibles para la generación de código automático ABAP IV basadas en UML 2.0.

LISTA DE PALABRAS CLAVE

MDA, CIM, PIM, PSM, Abap, SAP, Eclipse, EMF, Acceleo, Metamodelo, Ecore

Contenido

Lista de imágenes

1 INTRODUCCIÓN

SAP R/3 es un sistema de software ERP (Enterprise Resource Planning). Los sistemas de software ERP están destinados a la administración/optimización de los recursos y procesos de negocio de una organización o empresa. La implantación de un sistema SAP R3 por parte de las empresas es una tarea costosa, compleja y requiere un gran esfuerzo de tiempo y dinero. Una de las tareas más costosas es el desarrollo de aplicaciones a medida del modelo de negocio de la empresa. Estos desarrollos a medida son necesarios porque no todos los procesos de negocio se encuentran cubiertos por el estándar del sistema SAP R/3. Para el desarrollo de aplicaciones a medida, los sistemas SAP R3 utilizan su propio lenguaje de programación, denominado ABAP IV. Los sistemas SAP R3 tienen su propio entorno de desarrollo y que no permite la generación automática de código basada en modelos generados con lenguajes de metamodelado como UML. Esta situación obliga a implementar cualquier diseño que se realice directamente "a mano" por parte de los desarrolladores.

En los últimos años, SAP ha iniciado una migración de su entorno de desarrollo para que pueda ser ejecutado desde el IDE Eclipse. También ha liberado varios plugins para esta plataforma de desarrollo de software que permiten utilizarla como entorno de desarrollo de aplicaciones ABAP IV. La integración de ABAP IV en la plataforma Eclipse ha permitido abrir una puerta a la generación de código ABAP IV dirigido por modelos. La plataforma Eclipse dispone de herramientas propias de para el modelado de software como Eclipse modeling Framework (EMF) y herramientas para las transformaciones de modelo a texto (M2T) como Acceleo y que pueden integrase con los plugins desarrollados por SAP para Eclipse.

El presente trabajo se centra el desarrollo y construcción de un generador automático de código ABAP IV dirigido por modelos dentro del marco de la arquitectura MDA (Model Driven Arquitecture). Utilizando las herramientas de modelado y generación de código existentes en Eclipse y el lenguaje de metametamodelado ECore, generaremos un metamodelado a medida para modelar aplicaciones ABAP IV. El metamodelo permitirá diseñar modelos abstractos por parte de los desarrolladores para desarrollar aplicaciones SAP que satisfagan los requerimientos del proyecto. Los modelos desarrollados servirán como modelos de entrada al generador automático de código, transformando los modelos a código ABAP IV. Con esta herramienta se espera poder mejorando la productividad, calidad, mantenimiento y seguridad del software para sistemas SAP R3.

Debido a los diferentes tipos de desarrollo posibles en SAP, el trabajo se centrara en la generación de programas ABAP Report. Los ABAP Report o reportes ABAP son los desarrollos a medida más solicitados durante una implantación de un sistema SAP R3. Sin embargo, el metamodelo presentado en el presente trabajo, podría ampliarse a nuevas tecnologías que está siendo adaptadas por SAP como la computación ubicua o la integración con sistemas externos que envíen información a sistema y puedan ejecutar remotamente pequeñas secciones de código (Remote función cal o servicios web) como por ejemplo, un sistema que reconocimiento de matrículas de los camiones que entran al almacén y envié los números de matrícula a un servicio web. El generador de código podría ser utilizado para desarrollar y mantener el código que será ejecutado remotamente por sistemas externos.

Por último, compararemos los resultados obtenidos con las soluciones actuales, basadas en el lenguaje de metamodelado UML 2.0, y propondremos futuras líneas de trabajo para continuar con el desarrollo del generador automático de código ABAP.

2 AMBITO DE APLICACIÓN

Cuando la implantación de un sistema SAP R3 por parte de una empresa requiere el desarrollo de aplicaciones a medida para la gestión de su modelo de negocio, el proyecto se aborda en diferentes fases:

1. Especificación de requisitos
2. Diseño del software
3. Implementación
4. Pruebas validaciones
5. Despliego en el entorno de producción
6. Mantenimiento

El presente trabajo está ambientado en la fase de diseño y los primeros pasos de la fase de implementación. Actualmente el entorno de desarrollo de SAP para ABAP IV, no posee ninguna herramienta de diseño gráfica que permita generar modelos. Tampoco existen herramientas que permitan la generación de código ABAP a partir de modelos realizados con herramientas externas o de terceros. En resumen, durante la fase de implementación, los modelos generados durante la fase de diseño, han de ser implementados “a mano” por los desarrolladores.

En la actualidad, se están desarrollando aplicaciones para la generación de código ABAP IV a partir de modelos UML. Este trabajo, pretende analizar la posibilidad de generar el código a partir de un metamodelo propio definido con el lenguaje de metamodelado Ecore. El objetivo es demostrar que es posible desarrollar modelos dependientes de la plataforma y que puedan ser utilizados como modelos de entrada para la generar código ABAP IV para programas tipo report y orientados a objetos.

Una vez realizadas las pruebas con el metamodelo diseñado, compararemos los resultados obtenidos con las iniciativas disponibles para la generación de código ABAP IV. Estas iniciativas también están siendo desarrolladas para el IDE Eclipse y se centran en la utilización de modelos UML para generar diagramas de clases que después serán utilizados como modelos de entrada para generar las clases del modelo en ABAP IV.

3 DESCRIPCION DEL PROBLEMA

3.1 Sistema SAP R/3

SAP son las siglas de Systems Applications Products in Data Processing. Es una empresa multinacional alemana, fundada por antiguos empleados de IBM, que tomaron el nombre de la división en la que trabajaban en IBM [1]. SAP está dedicada al diseño de productos informáticos de gestión empresarial, tanto para empresas como para organizaciones y organismos públicos. Sus principales productos son SAP ERP para la administración y gestión, SAP Business Warehouse (SAP BW) para análisis, reporte y data warehousing, SAP BusinessObjects para el análisis y presentación de datos, SAP HANA la implementación de SAP AG de la tecnología de base de datos en memoria.

Ilustración 1 Sistema SAP R/3 y módulos

SAP R3 es un software ERP (Enterprise Resource Planing) que corresponde a un tipo de sistema de cómputo integrado de gestión. SAP R/3 permite integrar los datos de negocio para coordinar todos los procesos de negocio que se llevan a cabo actualmente en organizaciones o empresas. La principal ventaja del sistema SAP R/3, es que se encuentra organizado en un conjunto de módulos de software cliente/servidor claramente diferenciados. Cada modulo de un sistema SAP R/3 está enfocado en un área o funcionalidad concreta de las empresa u organizaciones (compras, ventas, gestión de almacén, finanzas…). Los módulos están completamente integrados gracias a la puesta en común de toda la información de cada modulo y por qué se alimentan de los datos almacenados en una base de datos común. La integración entre todos sus módulos es la característica más importante SAP R/3s. Una vez la información es almacenada, está disponible a través de todo el sistema facilitando los procesos de negocio y el manejo de la información y permitiendo la adaptación del sistema a los requerimientos individuales de cada empresa.

FI	CONTABILIDAD FINANCIERA
FI-GL	Cuentas de Mayor
FI-LC	Consolidación Sociedades
FI-AR	Cuentas a Cobrar
FI-AP	Cuentas a Pagar
FI-AA	Gestión de Activos
FI-SL	Special Ledger
	Cierres

IM	INVERSIONES
	Gestión de Inversiones

TR	TESORERIA
	Programa Conciliación
	Provisiones Posicionamientos
	Control de Fondos

CO	CONTROLLING
CO-CCA	Contabilidad por Centros Coste
	Contabilidad Presupuestaria
CO-PC	Control de Costes del Producto
CO-PA	Analisis de Rentabilidad
CO-OPA	Ordenes Internas
CO-ABC	Costes Basados en Actividades

HR	GESTION DEL PERSONAL
HR-PA-EMP	Datos Maestros de Personal
HR-PA-PAY	Nómina
HR-PA-TRV	Gastos de Viaje
HR-PD-OM	Organización y Planificación
HR-PD-PD	Desarrollo de Personal
HR-PD-SCM	Gestión de la Formación
HR-PA-APP	Selección de Personal
HR-PA-TIM	Gestión de Tiempos

MM	GESTION DE MATERIALES
MM - MRP	Planificación Necesidades Materiales
MM-PUR	Gestión de Compras
MM-IM	Gestión de Inventarios
MM-WM	Gestión de Almacenes
MM-IV	Verificación de Facturas
MM-IS	Sistema de Información
MM-EDI	Intercambio Electrónico de Datos
	Sistema Clasificación
	Gestión de Lotes

QM	CALIDAD
QM-PT	Herramientas de planificación
QM-IM	Proceso de Inspección
QM-QC	Control de Calidad
QM-CA	Certificados de Calidad
QM-QN	Notificaciones de Calidad

PP	PRODUCCION
PP-BD	Datos Basicos
PP-SOP	Gestión de la Demanda
PP-MP	Plan Maestro
PP-CRP	Plan de Capacidades
PP-MRP	Plan de Materiales
PP-SFC	Ordenes de Fabricación
PP-PC	Costes de producto
PP-IS	Sistema de Información
PP-PI	Industria de procesos
PP-CFG	Configuración de Producto

LO	GESTION DATOS GENERALES DE LOGISTICA
LO-MD	Datos Básicos
LO-VC	Gestión Variantes de Productos
LO-PR	Modelos Previsión y Comportamientos
LO-ECH	Cambios Ingeniería Objetos SAP

SM	GESTION DEL MANTENIMIENTO

EC	ENTERPRISE CONTROLLING
EC-PCA	Contabilidad Centros Beneficio
EC-BP	Planificación del Negocio
EC-MC	Consolidación a Nivel Directivo
EC-EIS	Executive Information System

SD	VENTAS Y DISTRIBUCION
SD - MD	Datos maestros
SD-SLS	Gestión de Ventas
SD-GF	GestiónTarifas y Condiciones de Precio
SD-SHP	Gestión de Expediciones
SD-BIL	Facturación
SD-IS	Sistemas de Información
SD-EDI	Intercambio Electrónico de Datos

PS	GESTION DE PROYECTOS
PS-BD	Datos Basicos
PS-OS	Planificación del proyecto
PS-PLN	Plan de Costes
PS-APM	Proceso de Aprobación
PS-EXE	Seguimiento y Progreso del Proyecto
PS-IS	Sistema de Información

IS-R	INDUSTRY SOLUTION RETAIL
IS-R	Planificación de Surtidos
IS-R	Reaprovisionamiento
IS-R	Formatos de presentación
IS-R	Sales Retail
CP	Inventario de proveedores
MM	Compras Retail
SD	Transporte
RIS	Sistema de Información Retail

PM	GESTION DEL MANTENIMIENTO
PM-EQM	Identificación Descripción
PM-PRM	Mantenimiento Preventivo
PM-WOC	Ordenes de Mantenimiento
PM-PRO	Proyectos de Mantenimiento
PM-SM	Gestión del Servicio

Ilustración 2 Lista de módulos de SAP R/3

El sistema SAP R/3 está basado en una arquitectura cliente/servidor de 3 capas:

- **La capa de presentación:** A través de un cliente de entorno gráfico (SAP GUI) esta capa hace de interfaz entre el usuario y el sistema. La capa de presentación envía la información del usuario al sistema y muestra al usuario la respuesta del sistema.
- **La capa de aplicación:** En esta capa, están los servidores que procesan las aplicaciones ABAP (o la parte de la aplicación que no es ejecutada en el cliente final) y un servidor de comunicaciones que permite la comunicación entre ellos.
  Dentro de esta capa, en el servidor de aplicación, se encuentra el diccionario ABAP. El diccionario ABAP es una base de datos abstracta dentro de la cual se definen tipos de campos, tablas, estructuras de campos (registros) y demás objetos de base de datos. Su función principal es "aislar" a las aplicaciones ABAP de la base de datos real. Con esto se hace posible crear una aplicación ABAP que opere igualmente sin importar cual motor de base de datos se está empleando.
- **La capa de base de datos:** Incluye una base de datos centralizada y un sistema gestor de la base de datos. Todos los datos del servidor se almacenan allí, incluido los datos de configuración del servidor o los programas ABAP. SAP R3 es compatible con la mayoría de bases de datos disponibles en el mercado como Oracle, MySQL, SQLServer, DB2 o Informix [2].

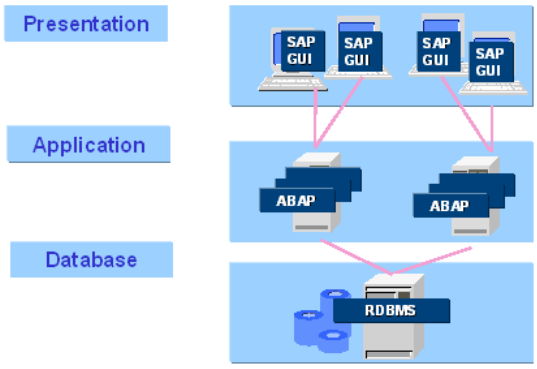


Ilustración 3 Arquitectura de 3 capas de un sistemas SAP R/3

Esta arquitectura cliente/servidor de 3 capas confiere a los sistemas SAP R/3 las siguientes características:

- **Escalabilidad:** Permite la adición de nuevos equipos en cualquiera de sus 3 niveles para acomodarse a los requerimientos dinámicos del sistema.
- **Portabilidad:** El software normalmente continúa en vigencia más tiempo que el hardware que lo soporta, es por ello por lo que el software SAP R/3 se caracteriza por su portabilidad a través de distintos tipos de hardware, sistemas operativos y RDBMS.
- **Apertura:** Todos los datos están almacenados en tablas que son accesibles sin necesidad de instrucciones complejas de recuperación de datos.
- **Parametrizabilidad:** SAP R/3 es un software estándar que dispone de herramientas específicas para la adaptación del software a las necesidades de la empresa. Estas herramientas se conoce como el customizing de un sistema SAP R/3.

Como se ha comentado anteriormente, los usuarios interactúan con el sistema SAP R/3 a través del cliente gráfico SAP GUI en la capa de presentación. SAP es una aplicación distribuida, donde se utiliza un software de cliente (SAPGUI) instalado en la estación de trabajo de un usuario, para acceder al servidor central de SAP remotamente a través de la red de la empresa.

Los usuarios tienen que autenticarse al acceder a SAP. Una vez identificados, pueden ejecutar los programas y aplicaciones disponibles en los sistemas SAP R3. Cada programa o aplicación en SAP R3 tiene asociado un nombre o alias que al ser ejecutado, llama al programa asociado. Cada alias se denomina "transacción" y es posible agrupar en un menú un conjunto de programas afines y sus respectivas transacciones. Por ejemplo, los usuarios pueden ejecutar transacciones como y agruparlas bajo un menú representativo:

- **ME21N:** Para la creación de pedidos de compra a proveedores
- **MIGO:** Para gestionar la recepción de las mercancías de comprada
- **VA01:** Creación de pedidos de venta a clientes
- **MM02:** Gestión de materiales
- **FB02:** Modificar documentos contables

Para simplificar su uso, el cliente SAP GUI muestra una pantalla de inicio con menú configurable con las transacciones y su descripción dependiendo de la parametrización del usuario, permitiendo la creación de menús personalizados para cada departamento de la empresa.

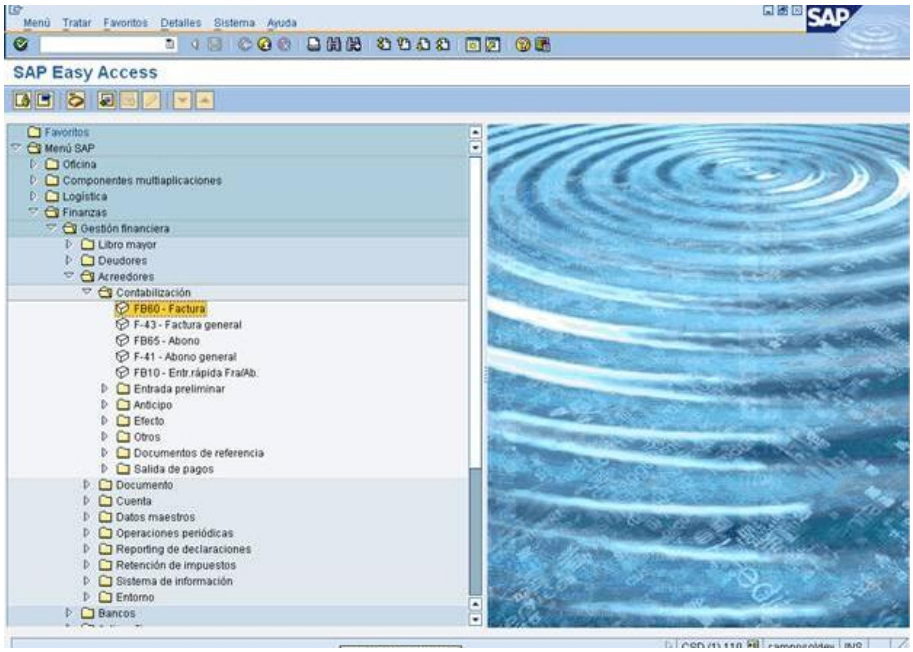


Ilustración 4 Menú con los accesos a cada transacción organizado por departamentos

Es posible crear programas y transacciones nuevos para solucionar problemas específicos que no estén cubiertos por alguno de los programas propios de SAP R3. Para el desarrollo de aplicaciones a medida, SAP R3 incluye el lenguaje de programación ABAP IV y un entorno de desarrollo de aplicaciones completo. El lenguaje de programación ABAP IV y su entorno de desarrollo se describen en detalle en el capítulo 3 del presente trabajo.

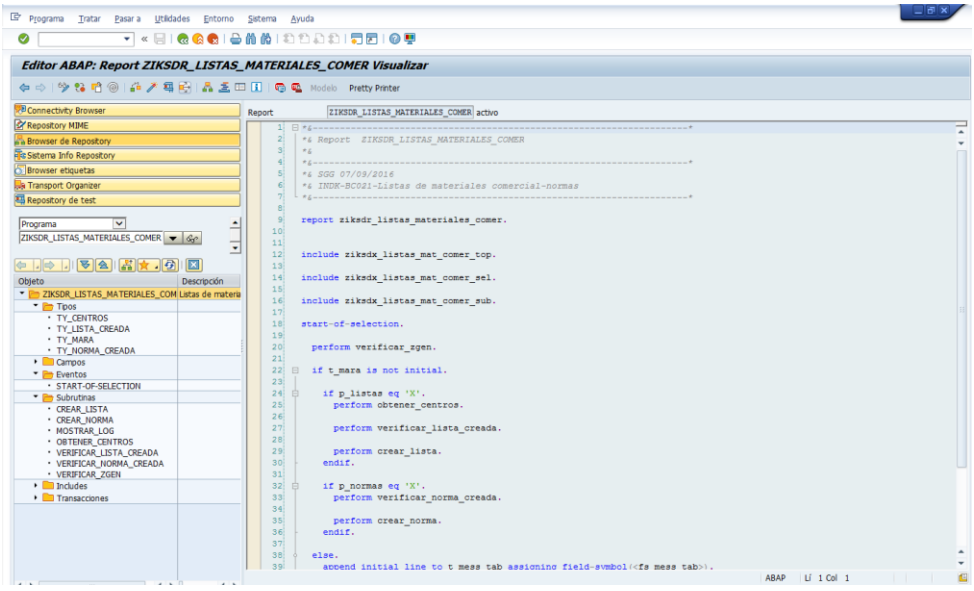


Ilustración 5 Entorno de desarrollo ABAP IV en SAP

3.1.2 Lenguaje de programación ABAP IV

ABAP IV (Advanced Business Application Programming) es un lenguaje de programación de cuarta generación orientado al desarrollo de aplicaciones de negocio, propiedad de SAP. ABAP IV se utiliza específicamente para el desarrollo de aplicaciones a medida sobre la mayoría de los productos de SAP, como su sistema SAP R/3. A diferencia de otros lenguajes de programación como C/C++, VisualBasic o Java con los que podemos desarrollar programas que después puedan ejecutarse en cualquier equipo o sistema, los programas desarrollados con ABAP IV son solamente ejecutables dentro de un sistema SAP.

Originalmente ABAP IV fue definido como un lenguaje estructurado de cuarta generación (de ahí el antiguo nombre de ABAP IV) orientado a eventos. Posteriormente ha incorporado elementos propios del paradigma de programación orientada a objetos. Para acceder a los datos de las tablas, ABAP IV utiliza sentencias de Open SQL que le permiten conectarse con prácticamente cualquier base de datos disponible en el mercado. Además cuenta con miles de funciones para el manejo de archivos, bases de datos, fechas, etc y por último, permite conexiones RFC (Remote Function Call) para conectar a los sistemas SAP con cualquier otro sistema o lenguaje de programación.

Las características principales de ABAP IV son:

- Contiene sentencias y palabras clave propias del lenguaje.
- Es un lenguaje orientado a eventos bien definidos, la secuencia de instrucciones depende del cumplimiento de una condición o evento.
- Es interpretado, no compilado.
- Se utiliza tanto en programación de informes como en programación de diálogo o Module Pool para SAP.
- Esta completamente integrado dentro del entorno de desarrollo de SAP.

Y como lenguaje orientado a objetos contiene:

- Objetos.
- Clases.
- Atributos.
- Métodos
- Interfaces.

El formato básico de las reglas ABAP IV es simple:

- Cada declaración ABAP debe terminar en un punto.
- Los elementos dentro de cada declaración deben estar separados por lo menos por un espacio.
- El final de línea es equivalente al espacio.
- Las declaraciones y palabras clave no distinguen entre mayúsculas y minúsculas.
- Si un texto en una declaración ABAP ocupa más de una línea, podemos usar el carácter & para combinar la sucesión de las mismas como si fuera una sola.

La entorno de desarrollo de ABAP IV integrado en los sistemas SAP r3 contiene diferentes herramientas para la edición de programas. Las más importantes son:

- Editor para escribir informes, module pools, includes y subroutine pools. También incluye una función de edición que se encarga de colocar correctamente el sangrado (indentación). Esta función ayuda a que el código sea más legible. También ofrece la posibilidad de escoger entre algunos modelos estandarizados acerca de las mayúsculas (por ejemplo todo mayúsculas, todo minúsculas, mayúsculas para declaraciones y palabras clave, etc.).
- Diccionario de base de datos para el procesamiento de las definiciones de tabla y recuperar tipos globales.
- Menú Painter para el diseño de la interfaz de usuario (barras de menú, de aplicaciones, asignación de teclas, pulsadores, campos de entrada de datos en pantalla, etc.)
- Screen Painter para diseñar pantallas y flujos lógicos
- Constructor de funciones para los módulos de funciones
- Constructor de clases para las clases de objetos de ABAP e interfaces

El navegador de objetos (transacción SE80) proporciona una interfaz integrada simple y unificada para poder utilizar todas estas herramientas.

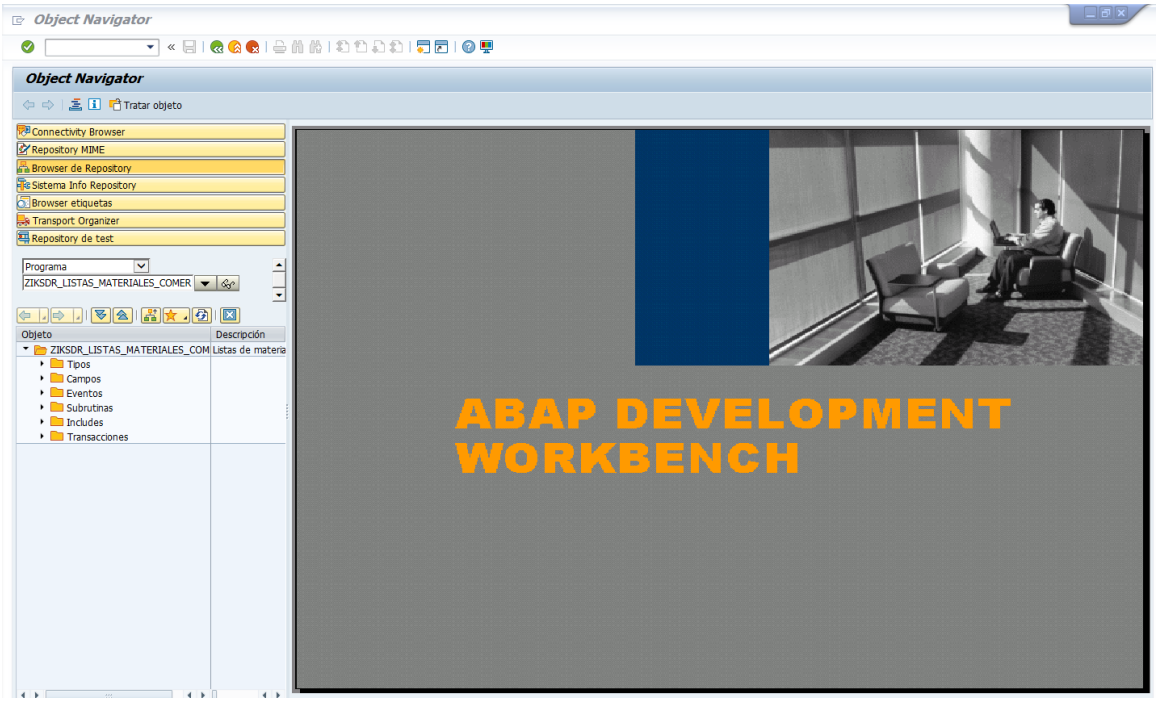


Ilustración 6 Navegador de objetos (transacción SE80)

3.1.3 Tipos de desarrollos ABAP IV

Existen 4 tipos de programas que pueden ser desarrollados con ABAP IV:

- **Report ABAP:** También llamados reportes ABAP. Son los programas más comunes y se utilizan mayormente para generar informes o listados de datos a partir de selección de datos que realiza el usuario. Estos listados pueden mostrar información de ventas, compras, finanzas, inventario, stock…etc. Los listados generados pueden ser interactivos y responder a acciones del usuario. Por ejemplo, un listado de pedidos de compra a proveedores puede permitir seleccionar pedidos al usuario para cancelar o para confirmar su entrega.

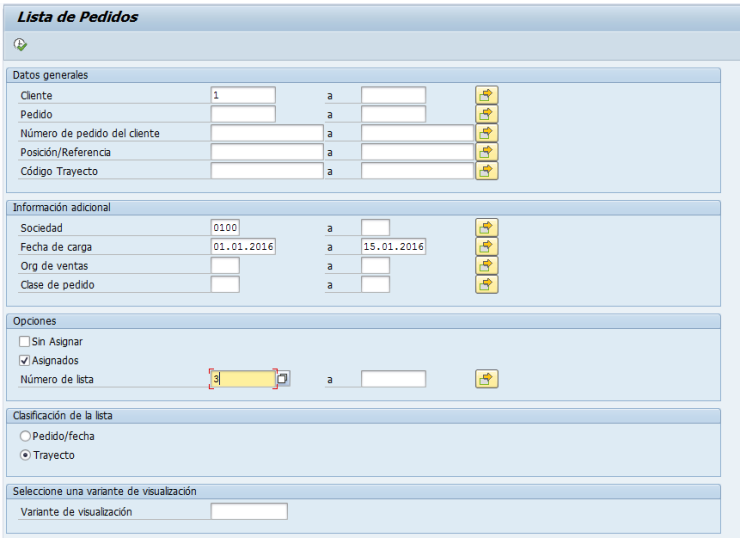


Ilustración 7 Pantalla de selección reporte ABAP IV

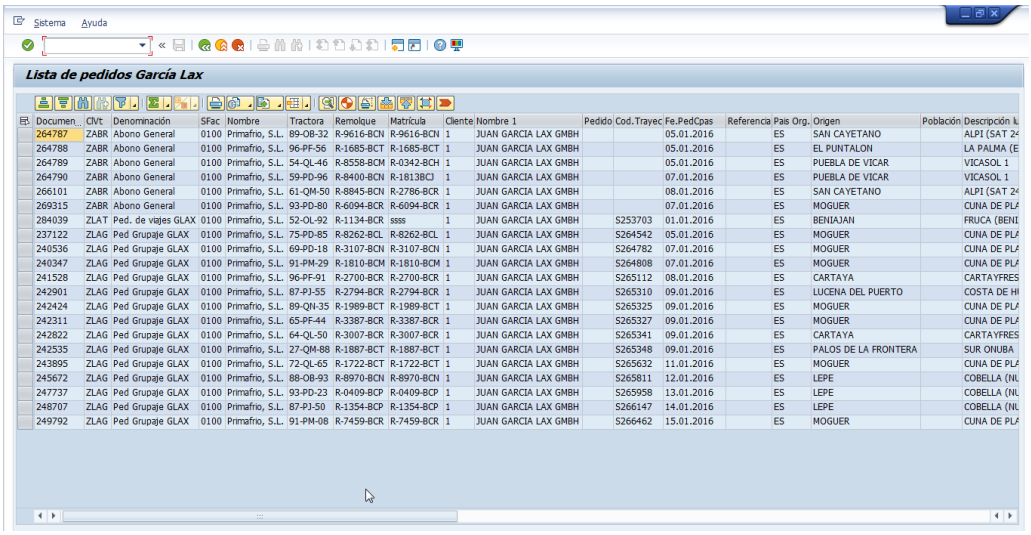


Ilustración 8 Listado reporte ABAP IV

- **Module pool:** También llamados programas de diálogo. Son pantalla sobre las que subyace algún tipo de evento como introducir datos o realizar alguna acción. Este tipo de pantallas con eventos definidos se conocen en SAP como dynpros. En las dynpros se definen atributos de la pantalla como el titulo o tamaño y se definen los elementos que la forman como campos de entrada/salida, checkbox, radiobuttons o pulsadores. Por último se definen los eventos asociados a los elementos o acciones del usuario. Por debajo de la secuencia de dynpros existe un programa ABAP que contiene los métodos y funciones con el código asociado a cada evento de la pantalla.

  Su función es representar modelos de negocios más complejos que un reporte o listado ABAP, sobre todo cuando es necesario implementar un proceso de negocio que requiere la interactuación de varios módulos de SAP o que requiere que le usuario interactué en el proceso a través de varias dynpros. Por ejemplo, la siguiente imagen muestra un module pool diseñado para una empresa de transporte que muestra los pedidos de venta, la información detallada del pedido de venta seleccionado, la mercancía que transporta y su ruta. Toda la información y acciones posibles están disponibles a las necesidades del cliente e integra la funcionalidad de varios módulos del sistema SAP R/3.

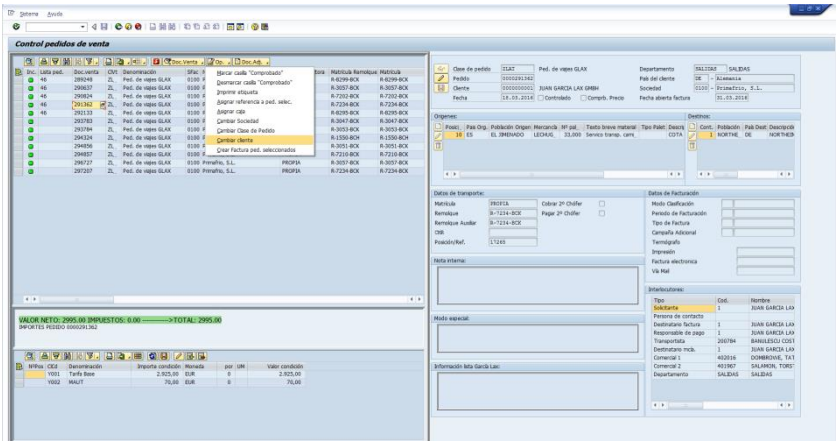


Ilustración 9 Module pool Control de transportes

Como hemos comentado, un module pool pueden estar formado por una o varias dynpros para definir un proceso lógico o proceso de negocio, cada evento definido en una pantalla puede actuar sobre si misma o llamar a otra dynpro. Todas las dynpros tienen definidas por defecto dos eventos:

  - **PBO Process Before Output:** Contiene todos los eventos y procesos que deben ejecutarse ANTES de mostrar la dynpro al usuario. Puede utilizarse para configurar la dynpro o acceder a la base de datos para mostrar los datos requeridos por el usuario

  - **PAI Process Afer Input:** Contiene todos los eventos y procesos que actúan según las acciones del usuario sobre la dynpro, ya sean introducir datos en los campos de entrada /salida o pulsar un botón en la misa. Puede redirigir el proceso y llamar a otras dynpros.

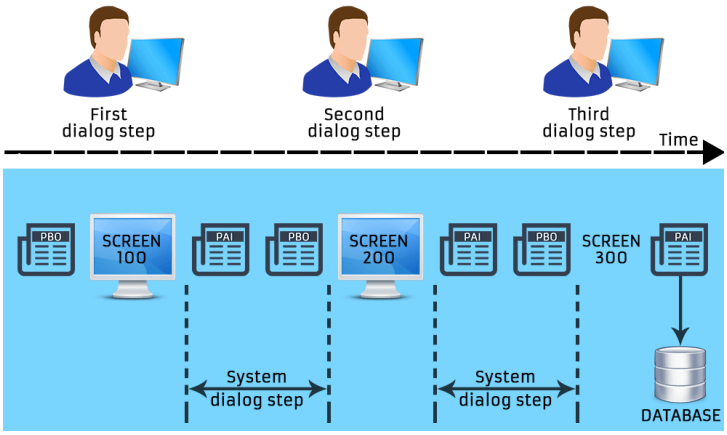


Ilustración 10 Secuencia de dynpros Module Pool Abap

- **BAPIS:** Son funciones globales con parámetros de entrada/salida y que soportan el protocolo Remote Function Call (RFC) esto permite que puedan ser expuestas para ser consumidas como servicios web. Su función principal es permitir la integración del software entre SAP y otros fabricantes de software o servicios web.

  Todas las BAPI's cumplen las siguientes reglas:
  - Soportan el protocolo Remote Function Call (RFC)
  - Son métodos de un Objeto de Negocios
  - Se procesan sin devolver ventanas de diálogo al programa que las invoca

  Normalmente las BAPIs encapsulan procesos de negocio como por ejemplo la creación de pedidos de compra, pedidos de venta, recepción de mercancías o expedición. El objetivo es facilita integración de procesos de partes del proceso de negocio con software externo. También se utiliza para que varias aplicaciones a medida que tengan que desarrollarse en los sistemas SAP R/3 y requieran ejecutar el mismo proceso de negocio comparta la misma BAPI que encapsula el proceso de negocio.

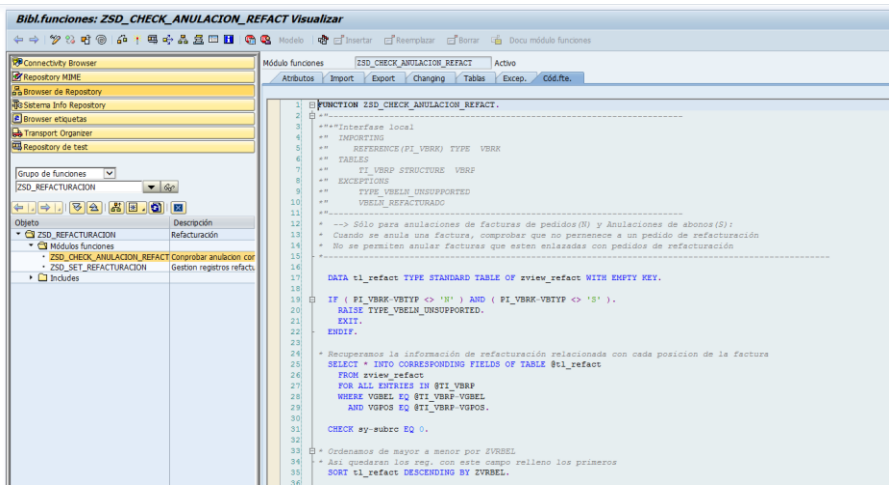


Ilustración 11 Entorno de desarrollo para Bapis ABAP

- **ABAP Object:** En 1999, con el lanzamiento de la versión 4.6 de R/3, SAP se incluyo una extensión para el lenguaje de programación ABAP IV. Esta extensión ampliaba el lenguaje ABAP IV incluyendo la programación orientada a objetos que fue denominada ABAP Object [3]. Hasta esta ampliación, el paradigma de la programación orientada a objetos solo se usaban exclusivamente en el diseño.

  ABAP Object, como paradigma de programación orientada a objetos, incluye las siguientes propiedades:
  - Encapsulamiento
  - Abstracción
  - Polimorfismo
  - Herencia

  Comparado con otros lenguajes orientados a objetos como Java, ABAP no es un lenguaje "puro" orientado a objetos. Es posible usar la orientación a objetos en programas ya existentes peor también combinar sentencias ABAP convencionales en programas ABAP orientados a objetos. El runtime de ABAP soporta tanto el modelo de programación procedural como el modelo ABAP Objects en cualquier programa ejecutable, module pool o bapis.

CARACTERISTICA	ABAP	JAVA	C++
Herencia múltiple	No	No	Si
Lenguaje "puro" orientado a objetos	No	Si	No
Liberación de objetos de manera automática	Si	Si	No
Existe el concepto "interfaz"	SI	SI	NO
Hereda de la clase OBJECT	Si	SI	NO
Manejo de parámetros opciones de forma explicita	SI	NO	NO
Los programas se ejecutan sobre…	Servidor de aplicaciones ABAP	Java Virtual Machine	Sobre el sistema operativo

  Las clases en ABAP Object pueden declararse globales o locales:
  - Clases Globales: Es posible definir clases e interfaces globales que son almacenadas en la biblioteca de clases dentro del repositorio de SAP y todos los programas ABAP en un sistema SP podrán acceder a ellas
  - Clases Locales: Son clases e interfaces que se definen dentro de un programa ABAP y solamente pueden ser usadas en el programa en donde están definidas.

  Cuando se usa una clase en un programa ABAP, el sistema primero busca una clase local con el nombre especificado. Si no se encuentra ninguna, busca una clase global.

3.2 Arquitectura Dirigida por Modelos (MDA)

La arquitectura MDA, propuesta y patrocinado por el Object Management Group (OMG) [4], se basa en separar claramente la especificación de la funcionalidad del negocio de la especificación de esa funcionalidad en una plataforma tecnológica determinada. Para conseguir esta separación se propone un proceso de desarrollo basado en la realización y transformación de modelos. En este proceso, cada etapa produce los modelos que sirven como punto de partida o entrada para la siguiente fase, siendo el objetivo final, la generación del código sobre una plataforma especifica. Con independencia del enfoque dado al ciclo de vida del software, la diferencia fundamental del ciclo de vida de desarrollo MDA con el tradicional es que, la comunicación entre las fases del proceso de desarrollo, se realiza a través de modelos que un computador es capaz de analizar y comprender.

Para tal propósito MDA caracteriza los siguientes tipos de modelos:

- **CIM:** Representa los modelos independientes de la computación (Computationally-Independent Model). Estos modelos se caracterizan en el dominio del problema. Este tipo de modelos surge ante todo en procesos de modelado de negocio e idealmente se conciben antes del levantamiento de requisitos para una aplicación particular.
- **PIM:** Representa los modelos que describen una solución de software pero no contiene detalles de la plataforma concreta en la que solución va a ser implementada, de ahí su nombre de modelos independientes de la plataforma (Platform-Independent Models). Estos modelos surgen como resultado del análisis y diseño.
- **PSM:** Son los modelos derivados de la categoría anterior, contienen los detalles de la plataforma o tecnología con que se implementará la solución, de ahí su nombre de modelos específicos de la plataforma (Platform-Specific Models).

Por último, a partir de los modelos PSM se generaría el código de la aplicación.

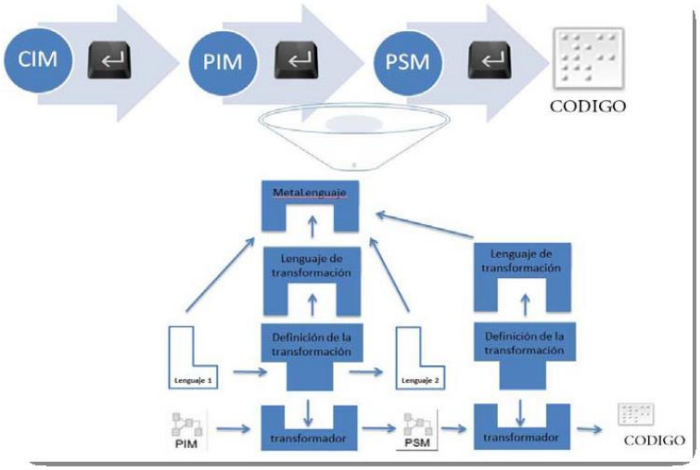


Ilustración 12 Arquitectura MDA

La transformación de modelos (Model To Model M2M) se considera el proceso central de MDA. La transformación de modelos es el proceso que, basado en una serie de reglas, define los mecanismos para el paso de un modelo origen a un modelo destino. EL objetivo es lograr que de un mismo PIM se pueda generar más de un PSM.

La transformación de PIM a PSM puede llevarse a cabo de varias formas:

- Construyendo manualmente el PSM a partir del PIM.
- Semiautomática, generando un esqueleto del PSM que es completado a mano.
- Automática, generando un PSM completo a partir del PIM.

Una de las principales ventajas de la arquitectura MDA es conseguir que a partir de un único modelo PIM, sea posible generar más de un modelo PSM. A partir de la estandarización promovida por QVT y atendiendo a las necesidades prácticas de la transformación de modelos, surgen diversas estrategias de transformación que van más allá del uso de elementos del metamodelo y definen mecanismos de transformación basados en tipos de elementos, patrones, información de la plataforma y otros modelos e información adicional.

El metamodelado es otra de las principales características de la arquitectura MDA y permite definir formalmente los lenguajes de modelado. Para la definición de los lenguajes de modelado a través de metamodelos, el OMG plantea una arquitectura de cuatro niveles o capas denominad MOF (Meta Object Facility). En terminología OMG estas capas se llaman M0, M1, M2, M3 y a continuación las describimos en detalle.

- **Capa M0 – Instancias:** Describe instancias de las entidades propuestas en un modelo de un sistema de información. En este nivel están las entidades "reales" de los sistemas, los objetos. Aquí no existen clases ni atributos, sino entidades físicas que existen en el sistema o aplicación. El código generado desde modelos UML estaría situado en esta capa M0.
- **Capa M1 - Modelo del sistema:** Representa el modelo de un sistema se software. Cada elemento de M0 es una instancia de un elemento de M1. Los modelos que construimos con UML estarían situados en esta capa M1.
- **Capa M2 - Metamodelo:** Especifica las entidades de un lenguaje de modelado y sus relaciones. Los elementos de la capa M1 son a su vez instancias del nivel M2. Aquí aparecerán conceptos como clase, atributo o relación. El metamodelo UML estaría situado en esta capa M2.
- **Capa M3 – Meta-metamodelo:** Corresponde a MOF [5], es una especificación que define un lenguaje abstracto para especificar, construir y manejar elementos comunes a cualquier metamodelo.

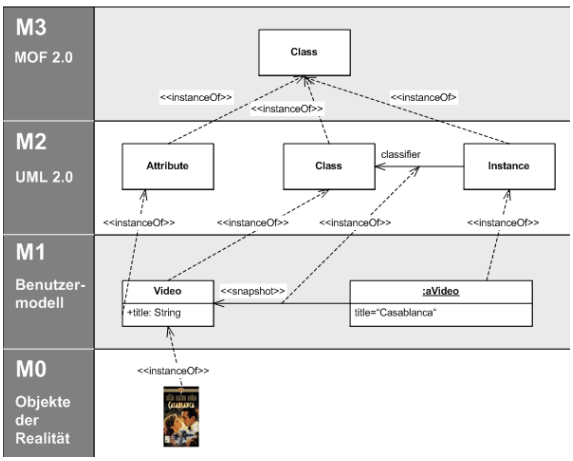


Ilustración 13 Ilustración de Meta-Object Facility

Para la descripción, gestión e intercambio de modelos, EMF utiliza el lenguaje XMI que provee un mecanismo de intercambio de modelos entre herramientas CASE utilizando XML. Su importancia se evidencia en que las herramientas CASE más conocidas utilizan XMI como especificación estándar para la funcionalidad de intercambio [6] . Una de las razones de la aceptación de XMI como lenguaje de intercambio de modelos es la tecnología ya desarrollada alrededor de XSLT, un modelo de procesamiento de documentos XML que incluye a la vez el lenguaje para describir las reglas de transformación. De esta forma se pueden construir diversas plantillas, que permiten generar documentos, recorrer su estructura y realizar transformaciones.

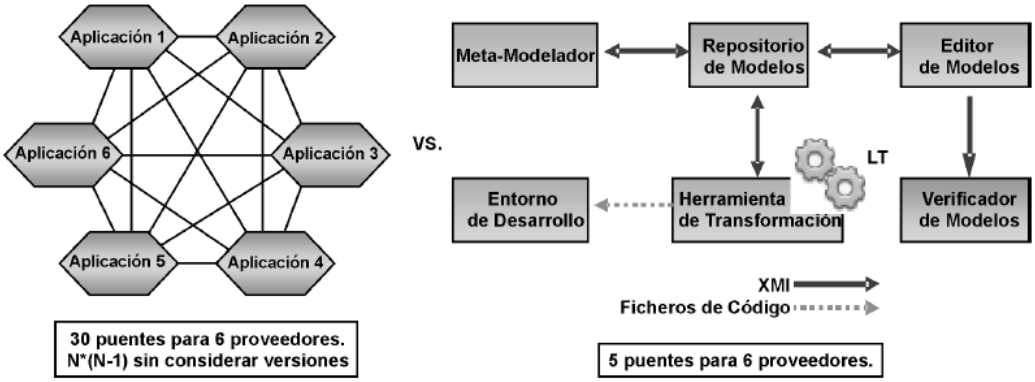


Ilustración 14 Importancia de XMI en el proceso con MDA

Comparado un ambiente desarrollado con 6 herramientas que utilizan UML contra un ambiente que usa UML y XMI en el contexto de MDA. El uso de XMI ha facilitado enormemente la distribución de las diversas responsabilidades que se exigen en un entorno de desarrollo con el enfoque MDA: las funcionalidades como edición de modelos, repositorio de modelos, verificación de modelos y transformación utilizan, producen o modifican documentos XMI que sirven para los propósitos de MDA.

Un proceso MDA tiene como fin la obtención de código a partir de un modelo. El proceso puede ser más o menos largo pudiendo así implicar más o menos transformaciones entre modelos (M2M) pero la idea fundamental es obtener un modelo definitivo al final del proceso de desarrollo que posteriormente será transformado a código mediante una transformación de modelo a texto (Model To Text o M2T) . El funcionamiento de una transformación de modelo a texto (M2T) consiste en un proceso por el que uno o varios modelos de entrada se transforman para generar ficheros de texto plano. Para llevar a cabo estas transformaciones se aplican reglas de transformación sobre cada uno de los elementos del modelo que nos interesan convertir a texto plano. Durante la transformación del modelo a texto, se lleva a cabo un proceso de inferencia por el cual se deduce la salida esperada de cada uno de los elementos del metamodelo que se analizan.

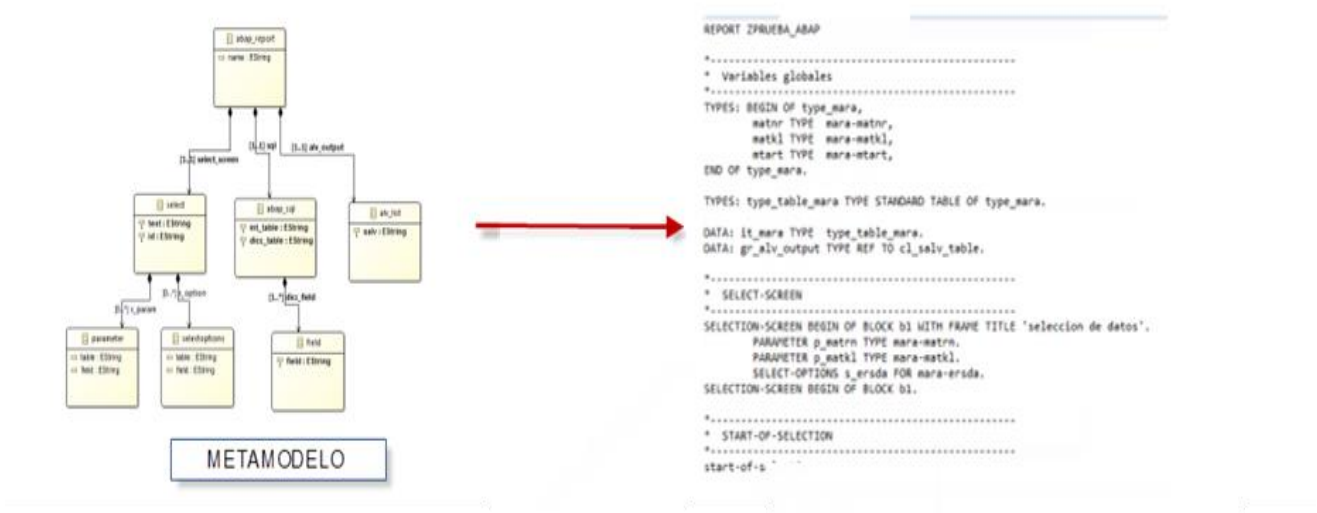


Ilustración 15 Transformación Modelo a Texto (M2T)

Para este proceso solamente necesitamos uno o varios modelos de entrada y definir las reglas de transformación que nos interesen. Estas reglas pueden corresponderse con reglas de transformación para obtención de código fuente en un lenguaje de programación conocido pero, también debemos tener en cuenta que puede crearse cualquier tipo de regla con el fin de obtener el texto en el formato que nos interese.

Así pues, por ejemplo, podríamos aplicar las reglas necesarias para transformar los modelos de entrada en sus correspondientes clases java, c++, android o como veremos más adelante en clases ABAP IV. Cualquier tipo de regla que genere código fuente que pueda resultarnos útil para el desarrollo de nuestras aplicaciones sobre una plataforma especifica.

Pero además, estos sistemas M2T, pueden enfocarse a temas totalmente contrarios a la generación de código fuente. Si aplicásemos las reglas correspondientes, podríamos por ejemplo crear ficheros ".txt" sobre cada elemento del modelo para generar documentación o información adicional al código que se está generando. En general, podrían aplicarse estas transformaciones M2T a cualquier cosa que se nos ocurra y que por supuesto nos resulte interesante para nuestro desarrollo, como la generación de ficheros XML, modelos XMI… ya que, lo que en cierto modo consiguen estas reglas de transformación, es automatizar la generación de código aplicando reglas de transformación sobre los modelos de entrada.

3.2.1 Aproximación MDA y SISTEMAS ERP

Como se ha comentado anteriormente, un sistema ERP es clase de software dirigido a la gestión y optimización de procesos de negocio de empresas y organizaciones. Uno de los aspectos donde más problemas surgen a la hora de implementar un sistema ERP, es su adaptación a las características y procesos de negocio propios de cada empresa. Dado que los procesos de negocio o la estructura organizativa de una empresa pueden cambiar muy rápidamente en el tiempo, los sistemas ERP deben poder adaptarse y dar respuesta a los cambios y nuevos procesos en el menor tiempo posible.

El proceso típico de un desarrollo de software para sistemas ERP incluye las siguientes fases:

- Toma de requisitos y análisis
- Diseño
- Codificación
- Pruebas
- Implantación
- Optimización

Las primeras fases son realizadas por analistas de negocio, consisten en realizar el análisis de los requerimientos y la documentación de los procesos de negocio de la empresa u organización. En las siguientes fases, los desarrolladores, se basaran en la documentación generada para diseñar, implementar, testear y desplegar el software en el sistema ERP.

Esta forma de desarrollo de software, denominada comúnmente desarrollo tradicional de software, es muy sensible a los cambios y modificaciones. Si durante la fase de implantación se produce algún cambio en el proceso de negocio o en los requerimientos, puede ocasionar retroceder hasta la fase de diseño para incluir los nuevos cambios o incluso que el analista deba volver a generar la documentación para poder incluir los cambios en el análisis.

Otro problema muy común en el desarrollo de software para sistemas ERP es, el gran hueco existente entre un analista de procesos de negocio y el desarrollador. Este hueco es debido que no tiene herramientas comunes para poder trabajar conjuntamente. Todos estos problemas acaban consumiendo gran parte del tiempo disponible de la fase de diseño e implementación.

Para solucionar ambos problemas, varios investigadores han propuesto utilizar arquitectura MDA para mejorar la comunicación entre las diferentes fases del desarrollo del software para sistemas ERP. Se propone la generación de modelos que representen procesos de negocio y que estos modelos sean utilizados como modelos de entrada para las siguientes fase del desarrollo [7] [8] [9]. Para esta propuesta, la arquitectura MDA se presenta como una candidata perfecta, porque utiliza los modelos formales para describir y automatizar el desarrollo y la integración del software. En los sistemas ERP los modelos MDA pueden utilizarse para representar formalmente procesos de negocio de las empresas u organizaciones. Estos modelos representarían de una forma lo más independientemente posible los procesos, sin tener en cuenta los sistemas ERP sobre los que serán implementados. Adicionalmente, estos modelos independientes del sistema ERP donde serán implementados, permitirían a los analistas de negocio modificar los procesos sin tener que reprogramar el sistema o los sistemas que lo llevan a cabo [10].

Siguiendo la arquitectura MDA, el modelo CIM sería la descripción de los procesos de negocio de la organización. Estos modelos deberían utilizar un lenguaje de modelado comprensible por los analistas de negocio. Como modelo se podría instanciar un modelo del metamodelo MOF apropiado y en el caso de que dicho modelo no fuera instancia de un metamodelo MOF, debería ser transformado al metamodelo MOF para representar procesos de negocio y poder realizar las transformaciones a los posteriores modelos de la arquitectura MDA. A partir de

transformaciones pertinentes, este modelo CIM se transformaría en un modelo PIM, independiente de la plataforma que podría estar expresado en un lenguaje de modelado de sistemas de información como UML. Una vez conseguido el modelo PIM, el proceso de desarrollo MDA seguiría su curso normal, utilizando herramientas automatizadas para la generación de modelos dependientes de plataforma (PSM) y generación del código que implementa dichos procesos sobre un sistema ERP especifico utilizando transformaciones de modelo a texto M2T [7] [8].

Una aproximación al anterior escenario la encontramos en el trabajo de Philippe Dugerdil y Gil Gaillard "Model-Driven ERP Implementation" [8] donde proponen el uso de los mecanismos extensibles existente en el metamodelo UML que incluye estereotipos, valores etiquetados y restricciones para crear los elementos de los modelos CIM y PIM. En concreto, los elementos de estos modelos serán entidades de negocio representadas por el elemento "recurso" que será una extensión de la metaclase "CLASS" en UML. Un recurso representara una entidad que es consumida o transformada por un proceso y tendrá un valor booleano asociado que representara el estado de la entidad ("Used" "Unused") esta entidad podrá representar por ejemplo ordenes, facturas, entradas de mercancías, albaranes, etc…

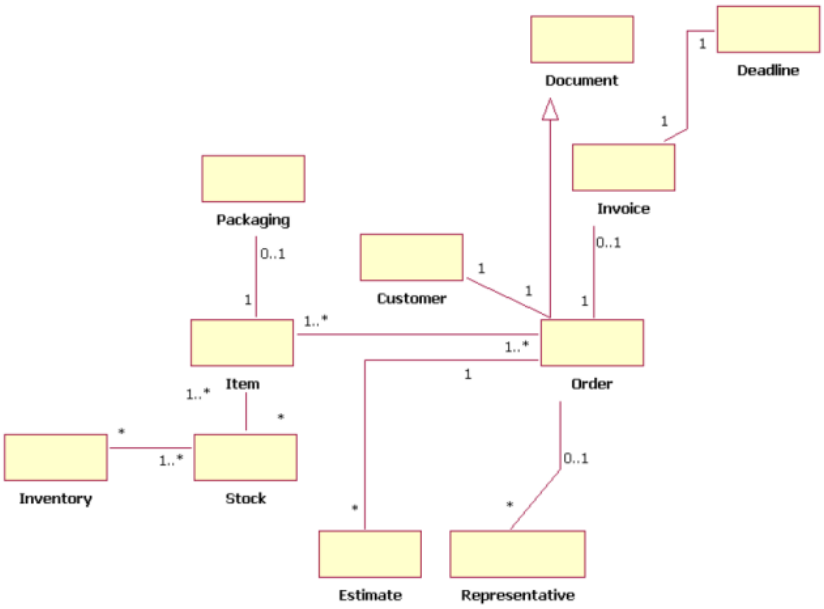


Ilustración 16 El modelo CIM

En cambio, para los modelos PSM, Philippe Dugerdil y Gil Gaillard proponen que los elementos del modelo representen entidades específicas del sistema ERP en el que se va a implementar el modelo tales como tablas de la base de datos, formularios, elementos de la pantalla, reports, objetos, etc…

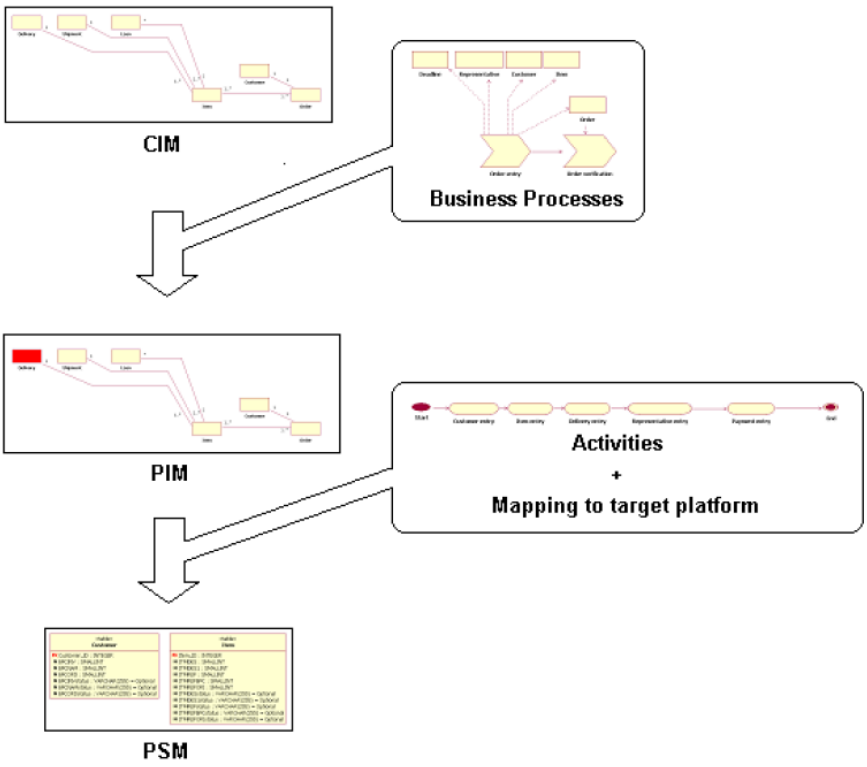


Ilustración 17 Resumen de transformaciones del modelo

Aplicando este procedimiento, podría conseguirse que los cambios constantes de los procesos de negocio de una empresa u organización puedan afrontarse de una manera mucho más eficiente y rápida, todo mediante la aplicación de una arquitectura MDA. Así, simplemente, el analista de negocio tendría que modificar los modelos CIMs que representan los procesos de la empresa y, mediante la aplicación de las transformaciones MDA se llegaría a la producción de código fuente para que sea ejecutado sobre un sistema ERP especifico.

Un ejemplo del interés que despierta la arquitectura MDA para sistemas ERP y de la aplicación del proceso anteriormente descrito, lo podemos observar en la herramienta SAP Composite Application Framework (CAF) para el entorno de desarrollo SAP NetWeaver basada en el IDE ECLIPSE. Este entorno de desarrollo de SAP es un conjunto de herramientas para integrar componentes de manera que formen una sola aplicación, enfocándose al diseño y configuración con un mínimo esfuerzo de programación. A través del CAF, podemos desarrollar servicios web a partir de cualquier componente definido en SAP NetWeaver, o componentes externos a través de estándares y protocolos como SOAP, XMLA, servicios de SSO, RMI, IIOP, entre otros que pueden ser usados para modelar procesos de negocio y generar el código JAVA JEE (edición Enterprise Java) de los servicios web que serán ejecutas sobre el servidor SAP Web Application Server Java [11] [12].

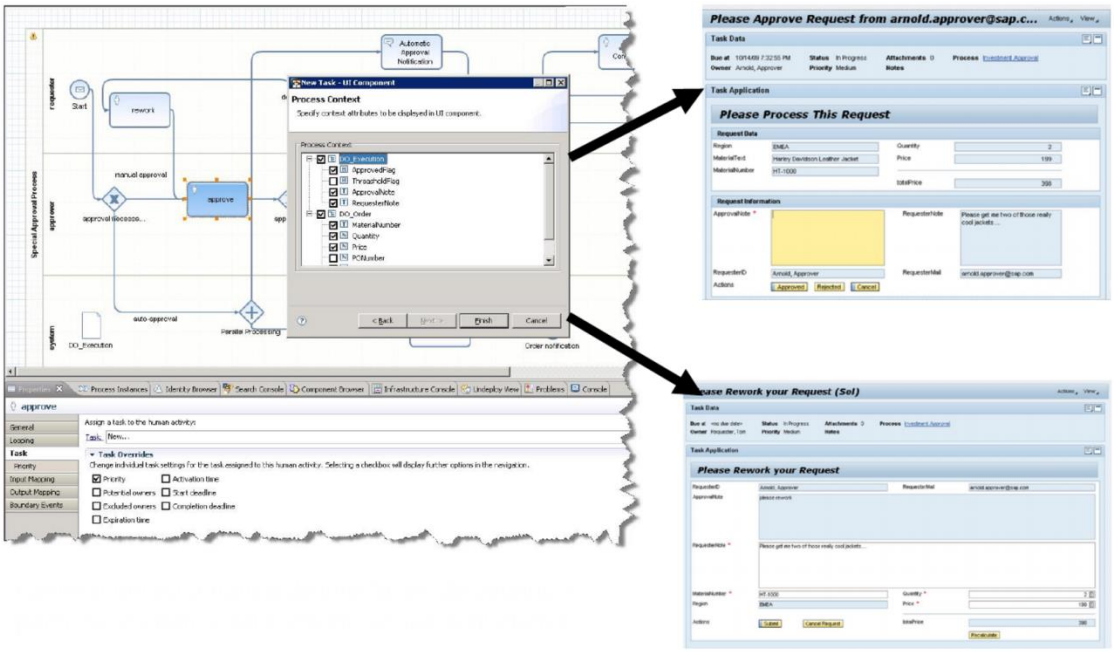


Ilustración 18 Generación automática de los servicios web basados en modelos

SAP Composite Application Framework (CAF) tiene un fuerte enfoque MDA para el desarrollo de aplicaciones web, lo que resulta en un rápido desarrollo de la capa de lógica de negocio para sistemas SAP.

3.3 Descripción del problema

EL desarrollo de aplicaciones a medida para sistemas SAP R/3, sigue generalmente el ciclo de vida de software tradicional. Los desarrolladores deben diseñar e implementar las aplicaciones, en base a una documentación generada por un consultor SAP, que recoge los requisitos y el análisis de las aplicaciones a medida solicitadas por el cliente.

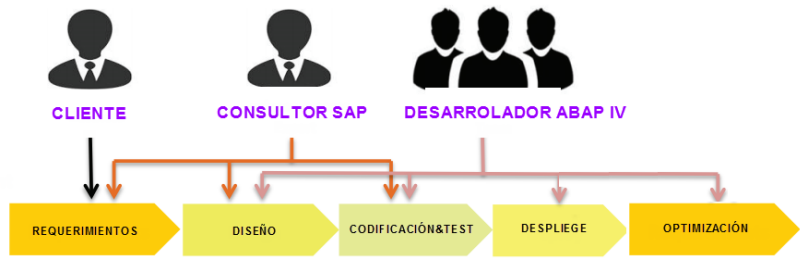


Ilustración 19 Proceso tradicional de desarrollo de aplicaciones

Dentro de un sistema SAP, como hemos visto, tenemos todo el entorno para el desarrollo de las aplicaciones SAP durante la fase de implementación. Pero para la fase de diseño, no existe ninguna herramienta que ayude en el proceso de diseño. Es posible utilizar herramientas de diseño externas para diseñar la aplicación SAP como DIA o Rational Rose. Pero no es posible la generación de código automáticamente desde los modelos generados desde las herramientas externas el entorno de desarrollo de SAP R/2. Queda bajo la responsabilidad del programador, en base a sus conocimientos, la transformación de los modelos diseñados durante la fase de diseño en código ABAP IV. Esta situación acaba normalmente derivando en los siguientes problemas:

1. Todo el diseño recae sobre el desarrollador ABAP
2. Los diseños se implementa de diferente manera dependiendo del nivel del programador asignado a la tarea perjudicando la reusabilidad del código.
3. Requisitos incompletos o que cambian durante el desarrollo.
4. No se puede validar el resultado hasta que el desarrollador ABAP ha terminado.
5. Cualquier cambio en el desarrollo conlleva un coste muy alto de tiempo.
6. Generación de documentación incompleta para el mantenimiento del código.

Estos problemas causan que en los proyectos, generalmente, la fase de diseño sea inexistente o depende por completo del programador. El programador, por motivos de tiempo y presupuesto, acaba saltando directamente a la fase de implantación de la aplicación ABAP IV en base a los requerimientos del cliente. Esta falta de diseño repercute enormemente en el código generado así como en su documentación, repercutiendo en el mantenimiento del código. La falta también de una fase de diseño acaba desplazando al consultor SAP a un mero papel de observador y validador del resultado del trabajo realizado por el desarrollador ABAP.
En el peor de los casos, acaba generando un bucle continuo entre las fases de implantación y pruebas sin control hasta el punto que tanto los desarrolladores como los analistas son incapaces de entender ellos mismo el desarrollo aumentando el tiempo y coste para la implantación del sistema SAP R/3 por parte de la empresa.

3.4 Solución propuesta

La solución propuesta en el presente trabajo de fin de máster es, basándonos en la arquitectura dirigida por modelos MDA, definir un lenguaje de modelado que permita crear modelos abstractos por parte de los desarrolladores ABAP específicos de plataforma (PSM) para sistemas SAP de las aplicaciones SAP y que sirvan como modelos entrada para un generador automático de código ABAP que realizara la transformación modelo a texto (M2T) y generar el código ABAP IV.

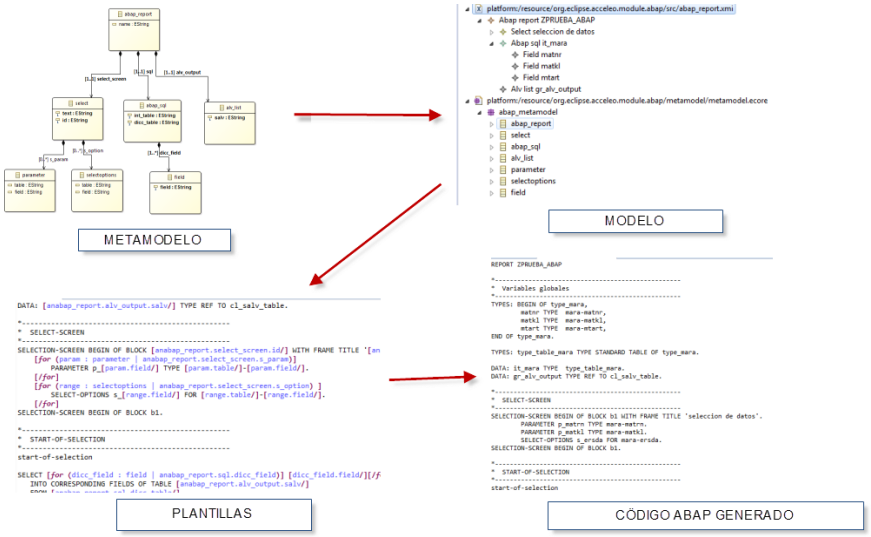


Ilustración 20 Transformaciones del modelo a código

Como se describirá posteriormente, siguiendo la arquitectura de 4 capas de OMG, utilizaremos el meta-metamodelo Ecore, un subconjunto estandarizado de MOF, de la herramienta Eclipse Modeling Framework (EMF) para definir un metamodelo con el que poder modelar aplicaciones SAP y cuyos elementos represente componentes o de desarrollos ABAP con los que poder diseñar modelos que representen aplicaciones SAP y que sirvan como modelos de entrada a un generador automático de código que transforme los modelos a código ABAP IV para ejecutar sobre un sistema SAP R3.

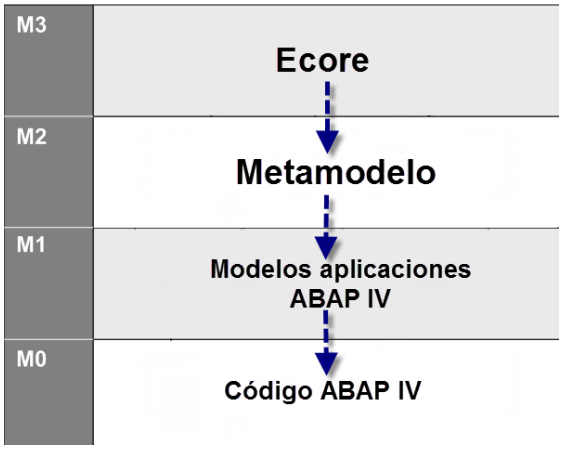


Ilustración 21 Solución propuesta, 4 niveles MOF

La transformación de modelo a texto (MT2) se realizara utilizando la herramienta Acceleo, desarrollado por la empresa Obeo Networks para Eclipse Modeling Tool. Esta herramienta es un sistema de generación de código basado en el estándar MOFM2T de la OMG perfectamente integrado en la plataforma Eclipse y que soporta cualquier tipo de modelo EMF como pueden ser UML, Ecore, DSLs, etc…

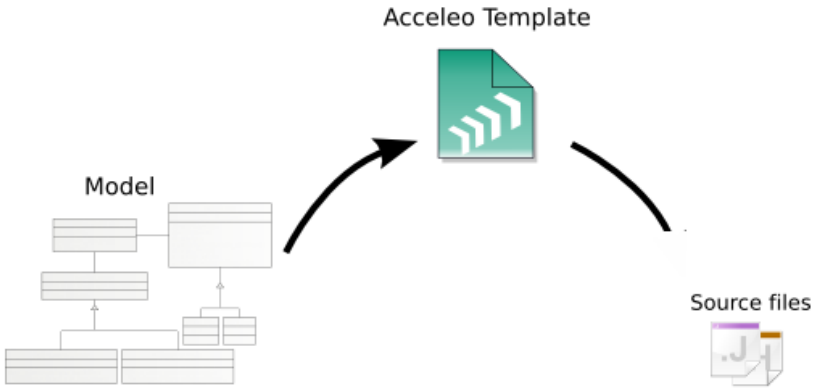


Ilustración 22 Transformación modelo a texto con Acceleo

Acceleo utiliza plantillas o templates donde se definen las reglas de generación que Acceleo utilizara para generar el código. En la solución propuesta en el presente trabajo diseñaremos las plantillas que definirán la transformación de los modelos de aplicaciones SAP a código ABAP IV ejecutable para sistemas SAP R3.

3.5 Restricciones introducidas al problema genérico

Existen 4 tipos de programas ABAP que pueden desarrollarse:

- ABAP Report
- Module Pool
- BAPIS
- ABAP Object

El presente trabajo de fin de máster se centrara en la generación de programas en ABAP de tipo Report o reporte. Los Report o reportes, son los desarrollos más solicitados por los clientes porque con ellos pueden extraer y analizar información de los procesos de negocio que realizan que les ayuden en la toma de decisiones. También demostraremos que es posible ampliar el metamodelo para crear ABAP Object y poder generar el esqueleto de clases a partir de de un modelo.

Dejaremos para futuras líneas de investigación la generación de programas de tipo Module Pool ya que requiere el diseño de las dynpro y que actualmente solamente es posible dentro del sistema SAP. Lo mismo ocurre con la Bapis que deben ser registradas manualmente en el repositorio ABAP del sistema SAP y tiene su propio interface para poder ejecutar el protocolo RFC.

3.6 Soluciones existentes al problema

No existe actualmente ninguna herramienta comercial disponible para tratar este problema en la fase de diseño de las aplicaciones SAP. Queda a elección del desarrollador, el uso de editores externos de diagramas u otras herramientas externas pero ninguna incluye la generación de código automático ABAP IV a partir del modelo correspondiente.

Si que existen iniciativas y proyectos de código abierto que están avanzando en esta dirección. Todas estas iniciativas, utilizan el nuevo entorno de programación ABAP para Eclipse y las herramientas de generación automática de código disponibles en esta plataforma a través del proyecto Eclipse EMF. Estas iniciativas utilizan el metamodelo UML para la generación de los modelos, a diferencia de la solución propuesta en el presente trabajo que utiliza Ecore para definir su propio metamodelo, centrándose en la generación de código orientado a objetos.

A continuación describimos dos de estas iniciativas que han conseguido la transformación de modelos UML, en concreto los diagramas de clases, a código ABAP IV.

UMAP - The UML to ABAP Code Generator

El propósito del Proyecto UMAP [13] es proporcionar a los desarrolladores ABAP los medios y herramientas para un mejor flujo de trabajo. Su objetivo es generar el código fuente de un diagrama de clases UML para ABAP IV mejorando la comunicación entre las fases de diseño e implementación en un proyecto.

Este proyecto está enfocado en la creación de clases e interfaces globales directamente en el repositorio de SAP. No incluye un editor propio para el diseño de los diagramas UML sino que es capaz de importar los diagramas UML creados con otros editores UML y generar en el sistema SAP clases correspondientes. Para poder importar los diagramas UML y generar el código ABAP, requiere la creación en el sistema SAP de una estructura de tablas, elementos de datos y dominios en la base de datos además de implementar un programa ABAP que será el encargado de generar las clases. También es necesario instalar la herramienta SAPLINK [14] en el sistema SAP para que el código generado suba automáticamente al repositorio.
Indicar que no ha sido posible probar este proyecto al no conseguir el autor de este trabajo permisos para crear las estructuras, tablas y código necesario para su ejecución en un sistemas SAP real, como explicaremos posteriormente nos basaremos en la documentación existente para evaluar este proyecto frente a la solución aportada en este trabajo.

UML2ABAP - UML to ABAP Code Generation

Creado Frédéric Madiot el proyecto UML2ABAP [15]se basa en la generación de código ABAP IV a partir de diagramas de clases UML. . Este proyecto también está enfocado en la generación de clases para la programación orientada a objetos pero A diferencia del anterior proyecto no es necesario crear ninguna estructura adicional en la base de datos de los sistemas SAP sino que genera un fichero con el código de las clases e interfaces del modelo que puede ser incluido como una clase local en los programas ABAP IV o para crear una clase o interface global en el repositorio SAP.

Como editor para los diagramas UML utiliza el plugin Papyrus de Eclipse [16]y para generar el código Abap utiliza ACCELEO [17]y es capaz de generar tano de generar código ABAP (clases) desde un diagrama UML como de generar el diagrama UML a partir de las clases existentes en el desarrollo.

El enfoque de UML2ABAP [15] según su autor es:

- Aprovechar los modelos UML que describen visualmente su aplicación en un formato independiente del lenguaje.
- Facilitar la comunicación con otras personas que no están familiarizadas con ABAP mejorar su productividad mediante la automatización de la producción de código de una definición más concisa y de alto nivel.
- Evitar errores de codificación en las partes que se generan de forma automática.

El problema es que el prototipo, como comenta su autor Frédéric Madiot, no está disponible para pruebas aunque aporta videos sobre su funcionamiento. También indicar que el último artículo publicado data del año 2013. No existen noticias posteriores ni documentación oficial sobre este proyecto a partir del año 2013.

Dada la imposibilidad de conseguir el prototipo para realizar las pruebas y evaluar la solución propuesta frente a UML2ABAP, como explicaremos posteriormente nos basaremos en la documentación existente para evaluar este proyecto frente a la solución aportada en este trabajo.
En el siguiente enlace UML2ABAP - Code generation for ABAP puede verse un video subido por Frédéric Madiot a youtube donde muestra sus primeros resultados generado código ABAP a partir de modelos UML.

3.7 Parámetros de evaluación

A final de este trabajo compararemos el resultado obtenido con nuestro metamodelo con los resultados de las iniciativas UML2ABAP y UMAP. Para esta evaluación compararemos los resultados según los siguientes parámetros:

1. Capacidad de generar código para programas de tipo ABAP Report
2. Capacidad de generar código para programas de tipo ABAP Object (clases)
3. Capacidad de generar código para programas de tipo MODULE POOL
4. Capacidad de generar los modelos a partir del código ABAP
5. Clases de Meta-modelos utilizados (UML, SYML, ECORE…)

4 RECURSOS PARA DESARROLLO MDA EN ECLIPSE

Eclipse es una plataforma de desarrollo *open source* basada en Java. Aunque es ampliamente utilizada entre la comunidad de desarrolladores del lenguaje Java, su principal objetivo es proporcionar un entorno de desarrollo integrado (Integrated Development Environment o IDE en ingles) extensible de forma indefinida a través de plugins sobre el que se pueden montar herramientas de desarrollo para cualquier lenguaje, mediante la implementación de los plugins adecuados. Existen plugins para el desarrollo de Java (JDT Java Development Tools) así como para el desarrollo en otros lenguajes de programación como HTML, XML, C/C++, COBOL, PHP, Python, ABAP IV, etc….

El IDE Eclipse se engloba dentro del Proyecto Eclipse [18]. El objetivo del proyecto eclipse es aunar tanto el desarrollo del IDE Eclipse como de algunos de los plugins más importantes como son el JDT, plugin para el lenguaje Java, o el CDT, plugin para el lenguaje C/C++. Este proyecto también alcanza a las librerías que sirven como base para la construcción del IDE Eclipse (pero pueden ser utilizadas de forma completamente independiente), como por ejemplo, la librería de widgets SWT.

Inicialmente, el proyecto Eclipse fue creado y mantenido por IBM en 1999 como el sucesor de su familia de herramientas para VisualAge. Bajo la dirección de IBM, se fundó el consorcio Eclipse, al cual se unieron empresas importantes como Rational, HP o Borlan. Desde el año 2004, el consorcio Eclipse, creó la Fundación Eclipse, una entidad legal y sin ánimo de lucro, que es independiente de IBM y está formado por un gran número de empresas importantes como **SAP**, HP, QNX, IBM, Intel, SAP, Fujitsu, Hitachi, Novell, Oracle, Palm, Ericsson y RedHat, además de algunas universidades e institutos tecnológicos para dirigir el desarrollo y evolución del proyecto Eclipse.

Ilustración 23 SAP es miembro de la Fundación Eclipse

Entre las principales características de Eclipse se encuentran:

- **Perspectivas, editores y vistas:** El entorno de trabajo de Eclipse está basado en perspectivas, que son pre configuraciones de ventanas y editores, relacionadas entre sí y que nos permiten trabajar en un entorno de forma optima.
- **Gestión de proyectos:** El desarrollo sobre eclipse se basa en proyectos, que agrupan los recursos relacionados entre sí como el código fuente, documentación, ficheros de configuración, etc.
- **Depurador de código:** Eclipse incluye un depurador de código fácil e intuitivo. Solo tenemos que pulsar un botón y nuestro código se ejecutar modo de depuración. La perspectiva del depurador está diseñada para mostrar, e forma ordenada, toda la información en ventanas y editores para facilitar la depuración del código.
- **Plugins para extender su funcionalidad:** Es una de las grandes ventajas de Eclipse. Es posible extender la funcionalidad del con la instalación de plugins tanto desarrollados y mantenidos por el proyecto Eclipse como por terceros. La colección disponible es muy grande. Los hay gratuitos, de pago, bajo distintas licencias, pero casi para cualquier cosa que nos imaginemos tenemos el plugin adecuado.
- **Lenguaje neutral:** Como hemos comentado, el uso del IDE Eclipse no está restringido a un único lenguaje de programación, con los plugins adecuados puede ser utilizado para desarrollar aplicaciones en un gran número de lenguajes de programación como HTML, Java, C, JSP, EJB, XML, GIF, ABAP IV,...
- **Multiplataforma:** El IDE Eclipse actualmente se puede ejecutar en:
    - Microsoft Windows 10,8,7,XP, 2000, Me, 98, NT
    - Linux (Red Hat, Suse, Ubuntu, Debian).
    - Sun Solaris 8 SPARC.
    - Apple Mac OS.
    - QNX Neutrino

Para el desarrollo del presente trabajo utilizaremos el paquete Eclipse Modeling Tools. Este paquete proporciona un entorno eclipse completo más un conjunto de librerías que tiene como finalidad proporcionar todas las herramientas que necesites para crear los diagramas de los modelos de sistemas de software. Además contiene un extenso catalogo de plugins adicionales desde el que se puede instalar una amplia gama de herramientas para el desarrollo de software basado en modelos como son Eclipse Modeling Framework (EMT) y Acceleo, dos herramientas que utilizaremos para el diseño del generador automático de código ABAP IV.

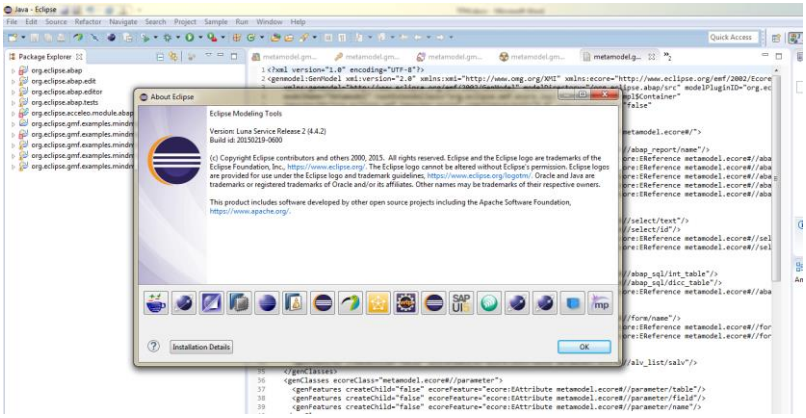


Ilustración 24 Paquete de distribución Eclipse Modeling Tools

4.1 ABAP Development Tools for Eclipse (ADT)

Hasta hace unos años, independientemente de la versión del componente SAP BASIS que este instalado en un sistema SAP, un desarrollador ABAP utiliza el entorno de desarrollo ABAP Workbench (transacción SE80) que se encuentra integrado dentro del sistema SAP y al que se accede a través de cliente SAPGUI.

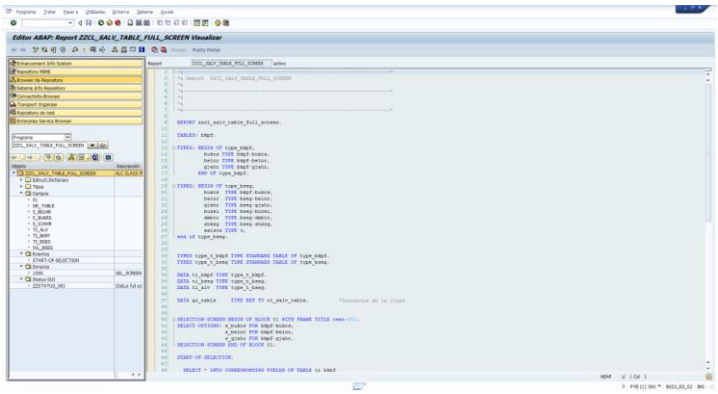


Ilustración 25 Entorno de desarrollo ABAP Workbench.

Desde el año 2012, SAP, como miembro de la Fundación Eclipse, empezó a migrar el desarrollo de aplicaciones ABAP IV para sistemas SAP R/3 a la plataforma abierta Eclipse. El objetivo de SAP es convertir a Eclipse en el entorno de desarrollo estándar para toda clase de desarrollos ya sean JAVA, SAPUI5, elementos de SAP HANA y ABAP IV. Es decir, la estrategia actual de SAP es proveer una única herramienta que contenga todas las tecnologías de desarrollo en el mismo IDE Eclipse.

ABAP en Eclipse (AiE) o formalmente SAP Application Development Tools for Eclipse (ADT), es un "plugin", un "add-on" de Eclipse que agrega la perspectiva ABAP a la plataforma Eclipse, proveyendo funciones para conectarse al repositorio ABAP (objetos DDIC) desde esta plataforma.

ADT, permite básicamente a los programadores ABAP desarrollar código fuente ABAP usando las capacidades del servidor de aplicaciones ABAP (AS ABAP) pero desde un entorno de desarrollo integrado (IDE) basado en Eclipse en vez del tradicional IDE, ABAP Workbench (Transacción SE80).

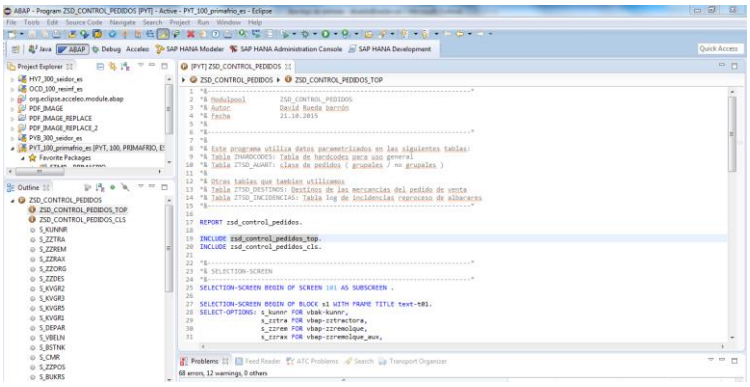


Ilustración 26 Eclipse + ABAP Development Tools for Eclipse

La idea general de ABAP en Eclipse es proveer un cliente Eclipse instalado en el equipo o PC del desarrollador y pueda conectarse a distintos sistemas SAP con versiones diferentes.

La conexión a los distintos sistemas SAP se establece mediante un protocolo basado en RFC/REST. En un lado, tenemos el cliente o front-end que provee herramientas de desarrollo estándar propios de la plataforma Eclipse, con la posibilidad de editar cada objeto de desarrollo (clases, reportes, módulos de función, etc.) y que solicitara servicios como actualizar código, compilar, revisar versiones a un sistema SAP que actuara como Backend.

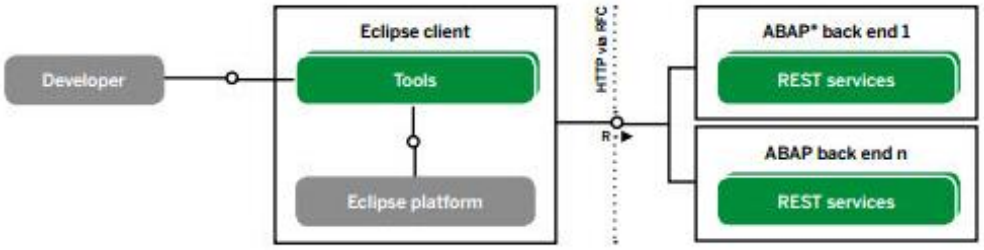


Ilustración 27 Conexión RFC/REST entre Eclipse y el servidor SAP

Aunque diseñemos nuestras aplicaciones en Eclipse, como hemos descrito anteriormente, el paradigma continúa siendo basado en el servidor, por lo tanto los objetos de desarrollo son almacenados únicamente en el sistema backend, y todos los servicios (chequeo de sintaxis, búsqueda, etc.) se ejecutan en el sistema backend al cual se está conectado, es decir, se necesita una plataforma o ambiente donde la aplicación pueda correr.

El plugin ABAP en Eclipse (AiE) posee las siguientes ventajas sobre ABAP Workbench como entorno para el desarrollo de aplicaciones SAP [19]:

- **Editor de código avanzado**: El editor de código del IDE Eclipse permite escribir nuestro código de una forma mucho más fluida y fácil que el editor de código del ABAP Workbench. Por ejemplo a la hora de escribir las clases en programación orientan a objetos, podemos escribir directamente la definición e implementación con sus métodos y atributos en vez de navegar entre las pantallas y pestañas del editor de clases del ABAP Workbench.
- **Eclipse permite trabajar offline:** Podemos trabajar sin conexión al sistema SAP y sincronizar las versiones de nuestras aplicaciones cuando nos conectemos al sistema SAP. Esta característica es imposible con el ABAP Workbench.
- **Trabajar en paralelo con varias conexiones:** En eclipse con ABAP en Eclipse es posible trabajar con varias conexiones a la vez de una forma clara y ordenada en pestañas en la misma ventana.
- **Mejor soporte para la refactorización:** tenemos a nuestra disposición todas las herramientas de la plataforma Eclipse para la refactorización del código.
- **Integración con otras herramientas de Eclipse:** Otra de las ventajas que ha traído la integración del IDE de ABAP en ECLIPSE ha sido la posibilidad de trabajar con otras herramientas diseñadas para el IDE Eclipse como el framework de modelado Eclipse Modeling Framework (EMF) [20] para la generación de código ABAP IV de forma automática y que utilizaremos como herramienta para el desarrollo del prototipo propuesto en este trabajo de fin de máster.

Como algunas de las características son dependientes de la funcionalidad del sistema backend, es claro que NO todas las características están disponibles con todas las versiones de los sistemas backend. Por lo tanto, aprovechar esas características, dependerá de la versión de SAP Netweaver que se tenga instalada en el sistema servidor ABAP backend.
Así, si bien se puede operar con las ABAP Development Tools (ADT) desde la versión de SAP Netweaver 7.31 SP04, para poder explotar las nuevas funcionalidades ofrecidas en la última versión del plugin ADT liberada por SAP (por ejemplo para la 2.31, es requerido contar con las funcionalidades del backend de SAP NetWeaver Release 7.40 SP08 o SAP NetWeaver Release 7.31 SP13.

4.3 Eclipse Modeling Framework (EMF)

Eclipse Modeling Framework, en adelante EMF, constituye el núcleo de la plataforma Eclipse para el desarrollo dirigido por modelos. Es un proyecto de eclipse para el desarrollo de un framework de modelado que facilite la generación de código a partir de la definición de un modelo de datos estructurado. Las librerías y herramientas de Eclipse Modeling Framework están incluidas en el paquete de distribución Eclipse Modeling Tools.

EMF empezó como una implementación de la especificación MOF y que a su vez, dado su amplio uso, EMF ha influido en el proceso de estandarización de MOF 2.0. Como consecuencia de dicha influencia, OMG identificó un subconjunto de MOF llamado essential MOF (EMOF), durante la estandarización de MOF 2.0 en 2003, como subconjunto suficiente, ya que MOF era muy complicado, para la mayoría de las realizaciones del estándar. Por último, el modelo de metadatos del tipo MOF en EMF se denomina Ecore y es conforme con el estándar EMOF del OMG [21] [22].

EMF utiliza XMI (intercambio de metadatos XML) como formato canónico de una definición de modelo. Su importancia se evidencia en que las herramientas CASE más conocidas utilizan XMI como especificación estándar para la funcionalidad de intercambio [6] . Una de las razones de la aceptación de XMI como lenguaje de intercambio de modelos es la tecnología ya desarrollada alrededor de XSLT, un modelo de procesamiento de documentos XML que incluye a la vez el lenguaje para describir las reglas de transformación. De esta forma se pueden construir diversas plantillas, que permiten generar documentos, recorrer su estructura y realizar transformaciones y proporciona herramientas y soporte de ejecución para producir un conjunto de clases Java, otro conjunto de clases Adapter que facilitan la tarea de crear, editar y mantener los modelos generados.

EMF agrupa otros plugins que permiten:

- Realizar consultas sobre los elementos de un modelo EMF y su contenido(Query)
- Verificar si el modelo es conforme a su meta-modelo (Validation)
- Generar implementación en Java del modelo
- Generar un editor de modelos

El metamodelo Ecore, como hemos comentado, es un subconjunto de MOF, que define un lenguaje común y abstracto para definir lenguajes de modelado, y cómo acceder e intercambiar modelos expresados en dichos lenguajes. Ecore usa 6 construcciones básicas para definir un lenguaje de modelado:

- EClass: Conceptos o entidades del meta-modelo
- EAttribute: Propiedades de una EClass en forma de tipos primitivos. Por ejemplo los tipos primitivos int, float, String o tipos enumerados,
- EReference: Relación entre conceptos o EClass Multiplicidad Rol, Contención, Navegabilidad.
- EEnum: Definición de tipos de datos con valores numéricos
- EPackage: Definición de paquetes para organizar los elementos anteriores
- EOperation: Definición de métodos en los conceptos o EClass

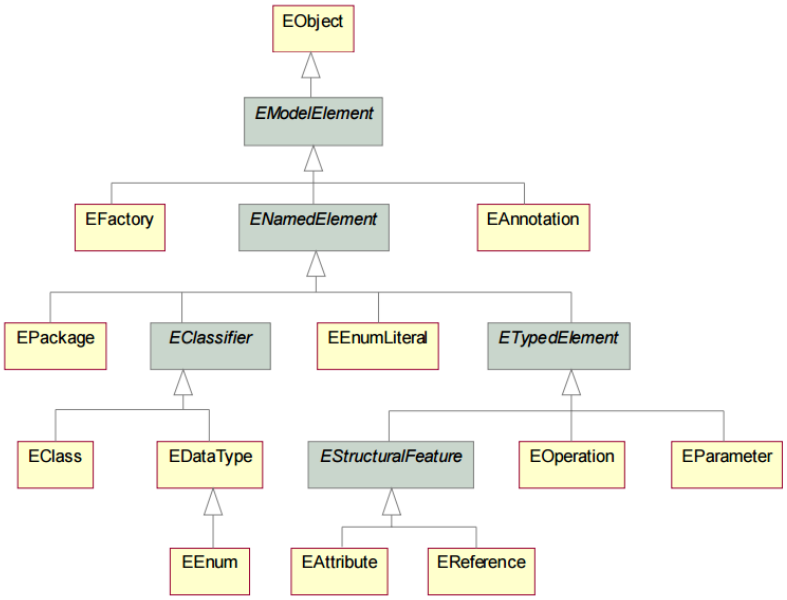


Ilustración 28 El meta-modelo Ecore

Como explicaremos en detalle en la sección 5 del presente trabajo, siguiendo la arquitectura de cuatro capas de modelado del OMG, definiremos un lenguaje de modelado para la representación de desarrollos SAP utilizado el Meta-metamodelo ECore.

Existen otros entornos similares al EMF donde desarrollar MDE, como por ejemplo las Microsoft DSL Tools para Visual Studio. Este entorno no se ha considerado por ser menos utilizado y no tiene la misma integración con sistemas SAP que posee el IDE de Eclipse

4.4 Acceleo

Por lo general un proceso MDA tiene como fin la obtención de código a partir de un modelo. El proceso puede ser más o menos largo pudiendo así implicar más o menos transformaciones entre modelos (M2M) pero la idea fundamental es obtener un modelo definitivo al final del proceso de desarrollo que posteriormente será transformado a código mediante una transformación de modelo a texto.

Acceleo es un generador de código basado en las especificaciones de las transformaciones de modelo de texto de las OMG desarrollado por la empresa Obeo Networks en 2005. Ha sido desarrollado en Java como un plugin de Eclipse con licencia EPL. Acceleo implementa la especificación del lenguaje estándar para las transformaciones de modelo a texto desarrollado por la OMG y que se denomina MOFM2T [17] [4].

Sus principales características son:

- Generación de código a través de cualquier tipo de meta-modelo compatible con EMF como UML 1, UML 2 e incluso meta-modelos adaptados a nuestro dominio específico.
- Un editor con autocompletado, detección y refactorización de errores en tiempo real
- Permite la generación del código de un modo modular a través de platilla MOFM2T.
- Permite implementar llamadas al sistema y librerías externas a través de Java.

En cuanto al funcionamiento de la herramienta es bastante sencillo. Se trata de una herramienta compatible con muchos de los modelos que actualmente podemos utilizar para representar realidades, entre los que se encuentran modelos UML, UML2, EMF, ECORE, XML, XMI, modelos generados por MoDisco (JAVAXMI)…

Utiliza un enfoque basado en plantillas que hacen uso de la API de Acceleo y se asocia a un metamodelo concreto cuando son diseñadas. Una plantilla contiene texto estático que se generara igualmente en todas las generaciones y texto dinámico, este ultimo son instrucciones y expresiones para recorrer, seleccionar y extraer información de los modelos. Estas expresiones están basadas en la implementación de Eclipse del lenguaje OCL [23].

Ilustración 29 Transformación modelo a código con plantilla Acceleo

Por ejemplo, la anterior imagen muestra una sección de una plantilla de Acceleo para generar los métodos públicos de una clase. La plantilla genera el mismo texto estático pero el texto dinámico se extrae del modelo.

5 DISEÑO DEL GENERADOR DE CÓDIGO ABAP IV

5.1 Diseño del metamodelo

Para el diseño del metamodelo se han utilizado los siguientes componentes de Ecore:

- **EPackage:** Componente que permite organizar clases y tipos de datos.
- **EClass:** Conceptos en el meta-modelo, representa una entidad propia del ámbito que se quiere modelar. En este caso representaran componentes de un programa ABAP IV.
- **EReference:** Asociación entre conceptos y las entidades como por ejemplo su cardinalidad o multiplicad.
- **EAttribute:** Define una propiedad de una Eclass como por ejemplo nombre del campo, tabla, nombre del programa, etc. y pueden ser de tipo numérico, carácter o booleano.

El diseño de metamodelos con Ecore es muy parecido a l diseño de un diagrama de clases de UML. Cada metclase EClass representara unívocamente a un elemento o componente de un programa tipo report o informe en ABAP IV (pantalla de selección, subrutina, listado ALV para mostrar los datos en pantalla…) y para cada Eclass definiremos sus atributos utilizando el componente EReference (por ejemplo para un listado ALV el nombre de la tabla que contiene los datos, el nº de columnas…). Por último definiremos como se relacionan las diferentes EClass entre ellas, estas relaciones se caracterizan principalmente por su tipo de asociación y la cardinalidad/multiplicidad de la misma.

Con estos componentes se ha diseñado el metamodelo, que a continuación pasamos a describir en detalle. En el siguiente apartado describiremos el proceso de transformación a código ABAP utilizando la herramienta de transformación de modelo a texto Acceleo [17].

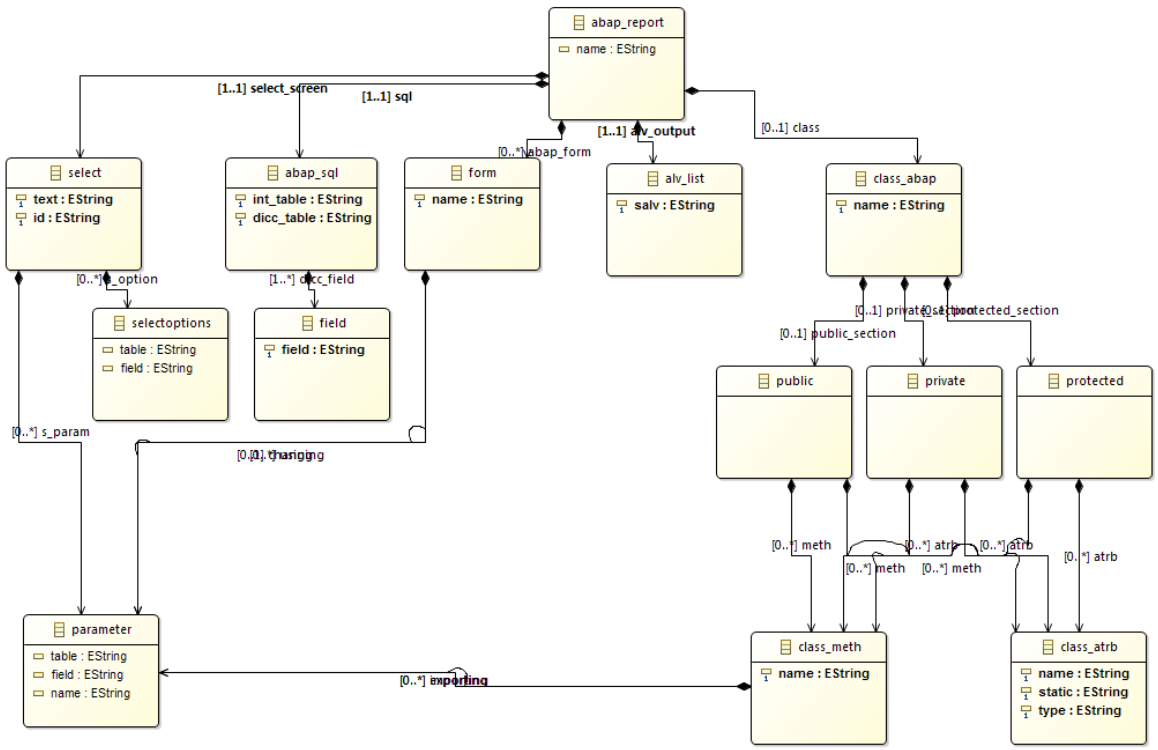


Ilustración 30 Metamodelo para modelar aplicaciones SAP

El metamodelo está compuesto por las siguientes EClass:

EClass abap_report: Es la Eclass principal o padre en el metamodelo. Representa un programa ABAP que puede ser un ABAP Repor o Abap Object para un sistema SAP R/3. Todo metamodelo debe tener una metaclase que actúe de contenedor raíz y todas las metaclases deben estar conectadas directa o indirectamente a la metaclase raíz.

- **Atributos**:
  - **EAtribute name:** Nombre del programa ABAP. Es un atributo de tipo cadena de caracteres (string).
- **Relaciones**:
  - **EReference select_screen:** Referencia a una EClass select, representa la pantalla de selección del programa ABAP Report. Una Eclass Abap_report podrá tener de 0 a 1 Eclass select_screen.
  - **EReference sql:** Referencia a una Eclass Abap _sql, representa una selección de datos de la BD. Una Eclass Abap_report podrá tener de 0 a N Eclass select_screen.
  - **EReference alv_output:** Referencia a una EClass alv_list que representa la salida por pantalla de los datos del programa ABAP Report. Una Eclass Abap_report podrá tener de 0 a 1 EClass alv_list.
  - **EReference abap_form:** Referencia a una EClass form, que representa una función o subrutina local para un programa ABAP Report. Una Eclass Abap_report podrá tener de 0 a N EClass form.
  - **EReference class:** Referencia a una Eclass class_abap, representa una clase ABAP Object.

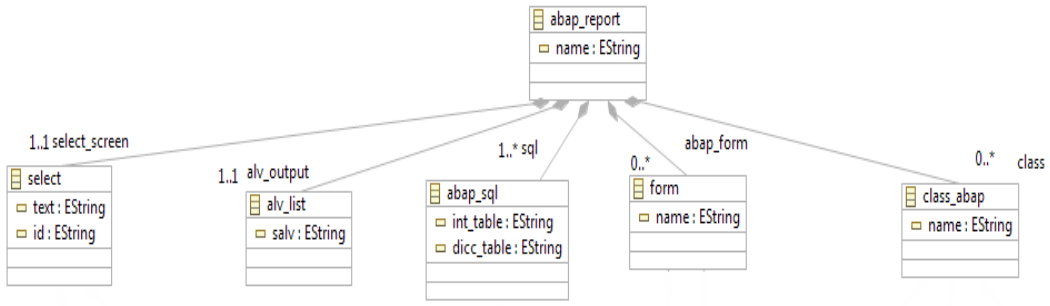


Ilustración 31 Metamodelo Eclass ABAP_REPORT y sus relaciones

EClass abap_select: Representa la pantalla de selección de un programa ABAP Report. Las pantallas de selección de datos son las pantallas de inicio de los programa ABAP Report, donde el usuario introduce los datos para seleccionar y filtrar los datos de la BD. El desarrollador debe programar que campos se mostraran si el usuario podrá introducir un único valor o un rango de valores.

- **Atributos**:
  - **EAtribute id:** Código identificado de la pantalla de selección Es un atributo de tipo cadena de caracteres (string).
  - **EAtribute text:** Titulo de la pantalla de selección. Es un atributo de tipo cadena de caracteres (string).

- **Relaciones:**
  - **EReference s_param:** Referencia a una EClass parameter, representa un parámetro de entrada simple de un único valor para la pantalla de selección. Una Eclas Abap_selet podrá tener de 0 a N EClass parameter.
  - **EReference s_option:** Referencia a una Eclass select_options, representa un parámetro de entrada de tipo rango que admite múltiples valores para la pantalla de selección. Una Eclas Abap_selet podrá tener de 0 a N EClass select_options.

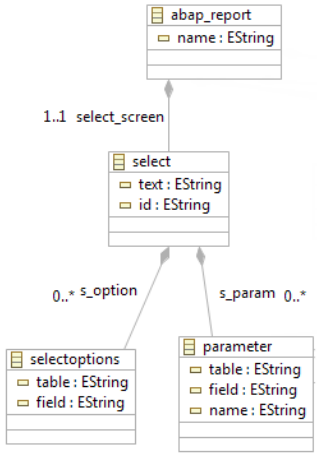


Ilustración 32 Metamodelo Eclass ABAP_SELECTT y sus relaciones

EClass abap_sql: Representa una sentencia SQL para recuperar datos de la base de datos. Normalmente los parámetros de la pantalla de selección se pasan como parámetros de la clausula WHERE de la sentencia SQL. El resultado se almacena en una tabla interna o variable del programa ABAP IV.

- **Atributos**:
  - **EAtribute in_table:** Nombre de la tabla interna donde se almacenara el resultado
  - **EAtribute dicc._table:** Nombre de la tabla en la BD a la que accedemos
- **Relaciones:**
  - **EReference dicc_field:** Referencia a una EClass field, representa un campo de la tabla de la BD a la que accedemos, abap_sql puede referenciar a mas de una EClass field para acceder a múltiples campos de la tabla en la DB.

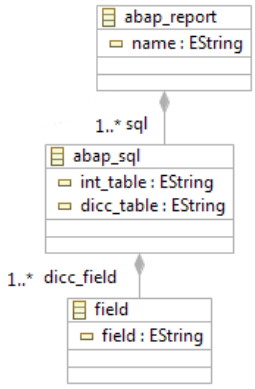


Ilustración 33 Metamodelo Eclass ABAP_SQL y sus relaciones

EClass form: Representa una subrutina o función ABAP. Puede tener parámetros de entrada y/o de salida, sirve para encapsular operaciones y reutilización de código.

- **Atributos**:
  - **EAtribute name:** Nombre de la subrutina o función
- **Relaciones**
  - **EReference using:** Referencia a una Eclass parameter. Representa los parámetros de entrada de la subrutina o función
  - **EReference changing:** Referencia a una EClass parameter, Representa los parámetros de salida de la subrutina o función

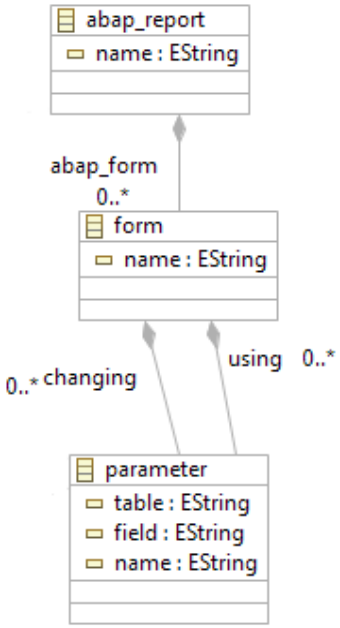


Ilustración 34 Metamodelo Eclass ABAP_FORM y sus relaciones

EClass alv_list: Representa la salida por pantalla en forma de listado de los datos. El listado por pantalla de tipo ALV se implementa instanciando la clase global CL_SALV_TABLE de ABAP y pasándole como parámetro de entrada el nombre de la tabla interna que contiene los datos.

- **Atributos**:
  - **EAtribute salv:** Nombre del listado por pantalla

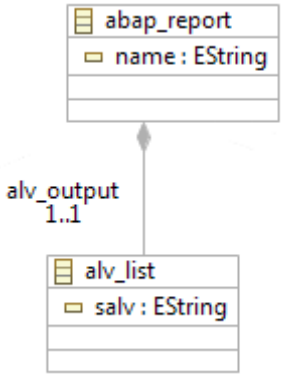


Ilustración 35 Metamodelo Eclass ALV_LIST y sus relaciones

EClass class_abap: Representa una clase ABAP para la programación orientada a objetos. Las clases en ABAP se componen de dos secciones DEFINCION e IMPLEMETNACION.

- **Atributos**:
  - **EAtribute name:** Nombre de la clase ABAP.
- **Relaciones**
  - **EReference public_section:** Referencia a una Eclass public. Representa la sección pública de la clase donde se definirán los atributos y métodos públicos de la clase.
  - **EReference private_section:** Referencia a una Eclass private. Representa la sección privados de la clase donde se definirán los atributos y métodos privados de la clase.
  - **EReference protected_section:** Referencia a una Eclass protected. Representa la sección protegida de la clase donde se definirán los atributos y métodos protegidos de la clase ABAP.

EClass public: Representa la sección PUBLIC de una clase ABAP. En esta sección se definen los atributos y métodos accesibles de la clase.

- **Relaciones:**
  - **EReference atrb:** Referencia a una Eclass class_atrb. Representa un atributo de la clase, la Eclass public puede contener de 0 a N referencias a la Eclas class_atrb.
  - **EReference meth:** Referencia a una Eclass class_meth. Representa un método de la clase, la Eclass public puede contener de 0 a N referencias a la Eclas class_meth.
  -

EClass private: Representa la sección PRIVATE de una clase ABAP.

- **Relaciones:**
  - **EReference atrb:** Referencia a una Eclass class_atrb. Representa un atributo de la clase, la Eclass private puede contener de 0 a N referencias a la Eclas class_atrb
  - **EReference meth:** Referencia a una Eclass class_meth. Representa un método de la clase, la Eclass private puede contener de 0 a N referencias a la Eclas class_meth

EClass protected: Representa la sección PROTECTED de una clase ABAP.

- **Relaciones:**
  - **EReference atrb:** Referencia a una Eclass class_atrb. Representa un atributo de la clase, la Eclass protected puede contener de 0 a N referencias a la Eclas class_atrb
  - **EReference meth:** Referencia a una Eclass class_meth. Representa un método de la clase, la Eclass protected puede contener de 0 a N referencias a la Eclas class_meth

EClass class_atrb: Representa un atributo de una clase ABAP

- **Atributos**:
  - **EAtribute name:** nombre del atributo.
  - **EAtribute static:** indica si el atributo es de tipo estático o no.
  - **EAtribute type:** indica el tipo de datos del atributo.

EClass class_meth: Representa un método de una clase ABAP

- **Atributos**:
  - **EAtribute name:** nombre del método.
- **Relaciones:**
  - **EReference importing:** Referencia a una EClass parameter. Representa un parámetro de entrada del método de la clase. La EClass clas_meth puede tener de 0 a N parámetros de entrada.
  - **EReference exporting:** Referencia a una EClass parameter. Representa un parámetro de salida del método de la clase. La EClass clas_meth puede tener de 0 a N parámetros de salida.

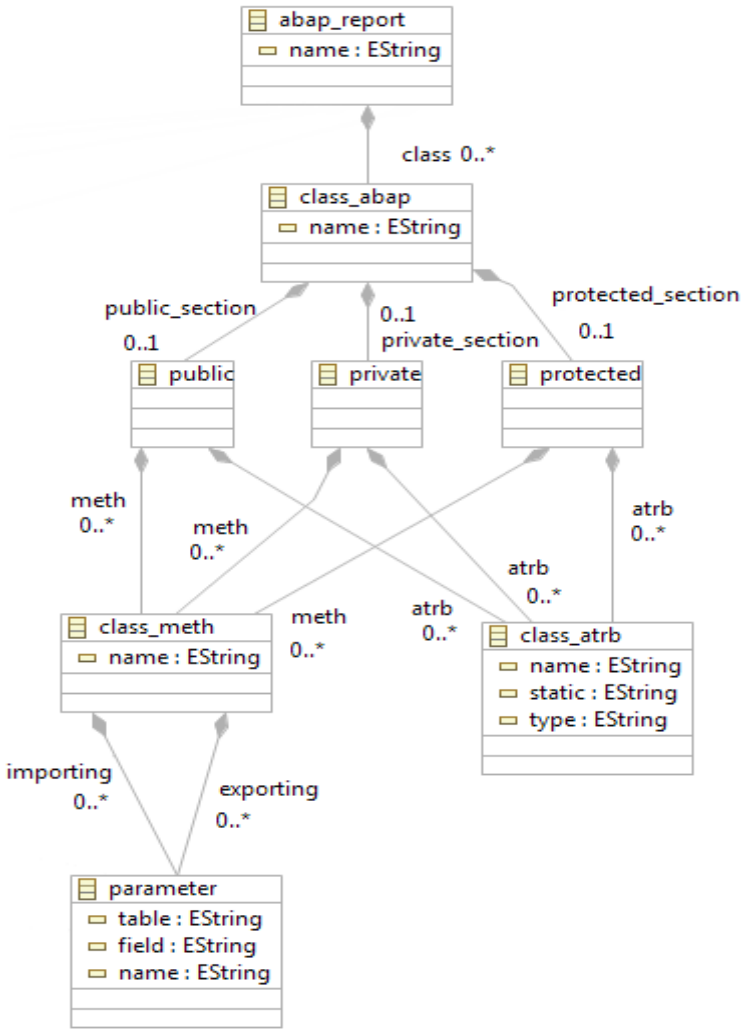


Ilustración 36 Metamodelo Eclass CLASS_ABAP y sus relaciones

EClass parameter: Representa un parámetro o variable del programa ABAP.

- **Atributos**:
  - **EAtribute name:** nombre del parámetro o variable
  - **EAtribute field:** tipo del parámetro
  - **EAtribute tabla:** tabla y campo de la base de datos en caso de que utilice el tipo de un campo existente en la base de datos

EClass selectoption: Representa un tipo de variable que acepta valores de rangos

- **Atributos**:
  - **EAtribute name:** nombre de la variable
  - **EAtribute field:** nombre del campo de la tabla para el rango
  - **EAtribute type:** tabla de la base de datos para el rango

EClass field: Representa un campo, variable básico de un ABAP Report

- **Atributos**:
  - **EAtribute field:** nombre del campo de la tabla para el rango

5.2 Diseño de la transformación modelo a código

En este capítulo describiremos las transformaciones modelo a texto (M2T) para la generación de código ABAP IV a partir de modelos generados con el metamodelo propuesto. Para la generación del código fuente a partir de los modelos desarrollados con el metamodelo propuesto se utiliza la herramienta Acceleo que permite diseñar plantillas que recurren a la API de Acceleo y permiten la navegación e identificación de los elementos del metamodelo al que estén asociados y generar tanto texto estático escrito directamente, que se generará de igual forma en todas las generaciones, y texto dinámico que variará en función del modelo de entrada en cada generación

Para la solución propuesta en este trabajo de investigación, se han diseñados dos plantillas:

- **Generate_abap_report:** Recibe como modelo de entrada un modelo que representa un programa tipo Abap Report y genera el código ABAP correspondiente ABAP IV para ejecutar sobre un sistema SAP R/3.
- **Generate_abap_class:** Recibe como modelo de entrada un modelo que representa una clase ABAP y genera código ABAP IV (definición e implementación de la clase) correspondiente para ejecutar sobre un sistema SAP R/3.

A continuación describiremos el diseño y funcionamiento de cada una de las plantillas.

5.2.1 Plantilla para ABAP reports

Esta plantilla de Acceleo recibe como modelo de entrada un modelo que representa un programa tipo Abap Report y genera el código ABAP correspondiente ABAP IV para ejecutar sobre un sistema SAP R/3.

La cabecera de la plantilla indica a qué tipo de metamodelo está asociada, en este caso se asocia la pantalla con el metamodelo desarrollado en el anterior apartado.

```
[comment encoding = UTF-8 /]
[comment Asociamos el metamodelo a la plantilla de Acceleo/]
[module generate_abap_report('http://www.example.org/abap_metamodel')]
```

Ilustración 37 Asociación del metamodelo a la plantilla ABAP REPORT

A continuación la plantilla recorre el modelo de entrada generado las diferentes secciones o eventos de un programa Abap report:

Atributos del programa: Se analiza la clase raíz ABAP_REPORT, se leen sus atributos para determinar el nombre del programa, se añade un texto estático que representa comentarios en ABAP para generar una cabecera con información del programa

```
[template public generateElement(anabap_report : abap_report)]
[comment @main/]
[file (anabap_report.name + '.txt', false, 'UTF-8')]
*---------------------------------------------------
* Programa: [anabap_report.name/]
* Autor:
* Fecha:
* Orden de transporte:
*---------------------------------------------------
* Descripción:
*
*---------------------------------------------------
REPORT [anabap_report.name/].
TABLES: [anabap_report.sql.dicc_table/].
```

Ilustración 38 Transformación para generar los atributos del programa

Variables Globales: La plantilla recorre las clases ABAP_SQL para leer sus atributos y sus clases dicc._filed asociadas que representan campos de una tabla de la base de datos, con esta información genera el tipo de tabla interna y la tabla interna donde se almacenara el resultado de la consulta sql a la base de datos. Por último, crea la variable que instanciara a la clase CL_SALV_TABLE que es representada en el modelo con la Eclass ALV_LIST

```
*---------------------------------------------------
*  Variables globalesa
*---------------------------------------------------
TYPES: BEGIN OF type_[anabap_report.sql.dicc_table/],
    [for ( atributo : field | anabap_report.sql.dicc_field )]
        [atributo.field/] TYPE  [anabap_report.sql.dicc_table/]-[atributo.field/],
    [/for]
END OF type_[anabap_report.sql.dicc_table/].

TYPES: type_table_[anabap_report.sql.dicc_table/] TYPE STANDARD TABLE OF type_[anabap_report.sql.dicc_table/].

DATA: [anabap_report.sql.int_table/] TYPE  type_table_[anabap_report.sql.dicc_table/].
DATA: [anabap_report.alv_output.salv/] TYPE REF TO cl_salv_table.
```

Ilustración 39 Transformación para generar las variables globales

Pantalla de selección: Recorre el modelo hasta encontrar la Eclass ABAP_SELECT y analiza las asociaciones existentes con las EClass SELECTOPTION y PARAMETER y que genera el código para el evento SELECT-SCREEN que muestra una pantalla inicial donde el usuario introducirá los datos de selección.

```
*-----------------------------------------------------
*  SELECT-SCREEN
*-----------------------------------------------------
SELECTION-SCREEN BEGIN OF BLOCK [anabap_report.select_screen.id/] WITH FRAME TITLE TEXT-001.
    [for (param : parameter | anabap_report.select_screen.s_param)]
        PARAMETER p_[param.field/] TYPE [param.table/]-[param.field/].
    [/for]
    [for (range : selectoptions | anabap_report.select_screen.s_option) ]
        SELECT-OPTIONS s_[range.field/] FOR [range.table/]-[range.field/].
    [/for]
SELECTION-SCREEN BEGIN OF BLOCK b1.
```

Ilustración 40 Transformación para generar el evento SELECT-SCREEN

Sentencias SQL: Antes hemos generado el código para crear una tabla interna donde guardar los datos de la consulta a la base de datos, ahora volvemos a recorrer la Eclass ABAP_SQL y generamos el código del evento START-OF-SELECTION que se ejecuta siempre después del evento SELECT-SCREEN y escribimos el código de la sentencia AQL representada en el modelo.

```
*-----------------------------------------------------
*  START-OF-SELECTION
*-----------------------------------------------------
start-of-selection.

SELECT [for (dicc_field : field | anabap_report.sql.dicc_field)] [dicc_field.field/][/for]
   INTO CORRESPONDING FIELDS OF TABLE [anabap_report.sql.int_table/]
   FROM [anabap_report.sql.dicc_table/]
   WHERE [for (param : parameter | anabap_report.select_screen.s_param) separator('\n AND ' )][param.field/] EQ p_[param.field/][/for]
    [if (anabap_report.select_screen.s_option <> null)]
      [if (anabap_report.select_screen.s_param <> null)]
        [for (range : selectoptions | anabap_report.select_screen.s_option) before('AND ') separator(' AND\n' ) ] [range.field/] IN s_[range.field/][/for]
      [else]
        [for (range : selectoptions | anabap_report.select_screen.s_option) separator(' AND\n' ) ] [range.field/] IN s_[range.field/][/for]
      [/if]
    [/if].
```

Ilustración 41 Transformación para generar el evento STRAT-OF-SELECTION

Llamadas a subrutinas: Recorremos las Eclass FORM del modelo, por cada una de ellas , recuperamos sus atributos y sus asociaciones a Eclass PARAMETER y generamos el código de la llamada a la subrutinas.

```
*-----------------------------------------------------
*  Llamadas a subrutidas para procesar os datos u operaciones
*  Solo hace falta rellenar el nombre de los parametros de entrada/salida
*-----------------------------------------------------
[if ( anabap_report.abap_form <> null )]
[for (abap_form: form | anabap_report.abap_form) separator('\n' )]
    PERFORM [abap_form.name/]
    [if ( abap_form.using <> null )]
       [for ( using : parameter | abap_form.using ) before('USING' ) separator('\n')] <<parametro de entrada [using.name/]>> [/for]
    [/if]
    [if ( abap_form.changing <> null )]
       [for ( changing : parameter | abap_form.changing ) before('CHANGING' ) separator('\n')] <<parametro de salida [changing.name/]>>  [/for]
    [/if]
        .
[/for]
[/if]
```

Ilustración 42 Transformación para generar las llamadas a las subrutinas

Listado por pantalla: Generamos el código para instanciar la clase CL_SALV_TABLE y llamar al método FACTORY que nos retorna una instancia de la clase asociada a nuestra tabla interna con los datos a mostrar y la llamada posterior al método DISPLAY para mostrar por pantalla el contenido de la tabla interna.

```
*-----------------------------------------------------
*  Generación del ALV -> listado por pantalla
*-----------------------------------------------------
TRY.
    CALL METHOD cl_salv_table=>factory
        EXPORTING
         list_display = if_salv_c_bool_sap=>false
        IMPORTING
          r_salv_table = [anabap_report.alv_output.salv/]
        CHANGING
          t_table      = [anabap_report.sql.int_table/] .
  CATCH cx_salv_msg .
  ENDTRY.

  CALL METHOD  [anabap_report.alv_output.salv/]->display.
```

Ilustración 43 Transformación para generar el listado por pantalla

Subrutinas: El código de las subrutinas, en un programa ABAP se implementa al final del programa. Recorremos las Eclass FORM del modelo e implementamos el código de las subrutinas, están se crean vacías para el programado pueda implementar dentro su código.

```
*-----------------------------------------------------
*  SUBRUTINAS / FUNCIONES
*-----------------------------------------------------
[if ( anabap_report.abap_form <> null )]
[for (abap_form: form | anabap_report.abap_form) separator('\n' )]
    [generate_form(abap_form)/]
[/for]
[/if]

[/file]
[/template]

[template public generate_form(abap_form : form )]
FORM [abap_form.name/]
    [for ( using : parameter | abap_form.using ) before('USING' ) separator('\n')] [using.name/] TYPE [using.field/] [/for]
    [for ( changing : parameter | abap_form.changing ) before('CHANGING' ) separator('\n')] [changing.name/] TYPE [changing.field/] [/for]
    .

ENDFORM.
[/template]
```

Ilustración 44 Asociación del metamodelo a la plantilla ABAP REPORT

5.2.2 Plantilla para clases ABAP

Esta plantilla de Acceleo recibe como modelo de entrada un modelo que representa una clase ABAP y genera el código ABAP correspondiente para implementar la definición e implementación de los métodos y atributos tanto públicos, privados o protected.

La cabecera de la plantilla indica a qué tipo de metamodelo está asociada, en este caso se asocia la plantilla con el metamodelo desarrollado en el anterior apartado.

```
[comment encoding = UTF-8 /]
[comment Asociamos el metamodelo a la plantilla de Acceleo/]
[module generate_abap_report('http://www.example.org/abap_metamodel')]
```

Ilustración 45 Asociación del metamodelo a la plantilla ABAP REPORT

La siguiente parte de la plantilla analiza la clase raíz ABAP_REPORT, determinando el nombre del programa y generando una cabecera de comentarios ABAP para que el programador la rellene con información descriptiva de la clase.

```
[template public generateElement(anabap_report : abap_report)]
[comment @main/]
[file (anabap_report.name + '.txt', false, 'UTF-8')]
*---------------------------------------------------
* CLASE: [anabap_report.name/]
* Autor:
* Fecha:
* Orden de transporte:
*---------------------------------------------------
* Descripción de la clase:
*  Atributos:
*
*  Metodos:
*
*---------------------------------------------------
```

Ilustración 46 Asociación del metamodelo a la plantilla ABAP REPORT

A continuación se analiza la Eclas CLASS_ABAP del modelo y se genera la definición de la clase ABAP. En la definición de una clase ABAP se implementa el código que define sus atributos y métodos. La plantilla analiza las relaciones existentes entre la Eclass CLASS_ABAP y las Eclass PUBLIC, PRIVATE y PROTECTEC para generar el código pertinente.

```
CLASS [anabap_report.class.name/] DEFINITION.

 [if ( anabap_report.class.public_section <> null )]
  PUBLIC SECTION.
  [if  ( anabap_report.class.public_section.atrb <> null )]
    [for ( cls_atrb : class_atrb | anabap_report.class.public_section.atrb )]
        DATA: [cls_atrb.name/] TYPE [cls_atrb.type/].
    [/for]
  [/if]
  [if  ( anabap_report.class.public_section.meth <> null )]
    [for ( cls_meth : class_meth | anabap_report.class.public_section.meth )]
        METHODS [cls_meth.name/]
         [if  (cls_meth.importing <> null )]
          [for ( p_import : parameter | cls_meth.importing ) before('IMPORTING ')] [p_import.name/] TYPE [p_import.field/] [/for]
         [/if]
         [if  (cls_meth.exporting <> null )]
          [for ( p_export : parameter | cls_meth.exporting ) before('EXPORTING ')] [p_export.name/] TYPE [p_export.field/] [/for].
         [/if]
    [/for]
  [/if]
 [/if]

 [if ( anabap_report.class.private_section <> null )]
  PRIVATE SECTION.
  [if  ( anabap_report.class.private_section.atrb <> null )]
    [for ( cls_atrb : class_atrb | anabap_report.class.private_section.atrb )]
        DATA: [cls_atrb.name/] TYPE [cls_atrb.type/].
    [/for]
  [/if]
  [if  ( anabap_report.class.private_section.meth <> null )]
    [for ( cls_meth : class_meth | anabap_report.class.private_section.meth )]
        METHODS [cls_meth.name/]
         [if  (cls_meth.importing <> null )]
          [for ( p_import : parameter | cls_meth.importing ) before('IMPORTING ')] [p_import.name/] TYPE [p_import.field/] [/for]
         [/if]
         [if  (cls_meth.exporting <> null )]
          [for ( p_export : parameter | cls_meth.exporting ) before('EXPORTING ')] [p_export.name/] TYPE [p_export.field/] [/for].
         [/if]
    [/for]
  [/if]
 [/if]


 [if ( anabap_report.class.protected_section <> null )]
  PROTECTED SECTION.
  [if  ( anabap_report.class.protected_section.atrb <> null )]
    [for ( cls_atrb : class_atrb | anabap_report.class.protected_section.atrb )]
        DATA: [cls_atrb.name/] TYPE [cls_atrb.type/].
    [/for]
  [/if]
  [if  ( anabap_report.class.protected_section.meth <> null )]
    [for ( cls_meth : class_meth | anabap_report.class.protected_section.meth )]
        METHODS [cls_meth.name/]
         [if  (cls_meth.importing <> null )]
          [for ( p_import : parameter | cls_meth.importing ) before('IMPORTING ')] [p_import.name/] TYPE [p_import.field/] [/for]
         [/if]
         [if  (cls_meth.exporting <> null )]
          [for ( p_export : parameter | cls_meth.exporting ) before('EXPORTING ')] [p_export.name/] TYPE [p_export.field/] [/for].
         [/if]
    [/for]
  [/if]
 [/if]

ENDCLASS.
```

Ilustración 47 Asociación del metamodelo a la plantilla ABAP REPORT

Por último, se genera el código para implementar los métodos de la clase. La plantilla recorre las Eclass PUBLIC, PRIVATE y PROTECTED recorriendo sus Eclass CLASS_METH para crear los correspondientes métodos de la clase.

```
CLASS [anabap_report.class.name/] IMPLEMENTATION.


    [if (anabap_report.class.public_section <> null) ]
      [if (anabap_report.class.public_section.meth <> null) ]
       [for ( cls_meth : class_meth | anabap_report.class.public_section.meth)]
        METHOD  [cls_meth.name/].

        ENDMETHOD.
       [/for]
      [/if]
    [/if]

    [if (anabap_report.class.private_section <> null) ]
      [if (anabap_report.class.private_section.meth <> null) ]
       [for ( cls_meth : class_meth | anabap_report.class.private_section.meth)]
        METHOD  [cls_meth.name/].

        ENDMETHOD.
       [/for]
      [/if]
    [/if]

    [if (anabap_report.class.protected_section <> null) ]
      [if (anabap_report.class.protected_section.meth <> null) ]
       [for ( cls_meth : class_meth | anabap_report.class.protected_section.meth)]
        METHOD  [cls_meth.name/].

        ENDMETHOD.
       [/for]
      [/if]
    [/if]

ENDCLASS.

[/file]
[/template]
```

Ilustración 48 Asociación del metamodelo a la plantilla ABAP REPORT

6 EVALUACIÓN Y PRUEBAS

6.1 Descripción del proceso para la creación de un modelo de pruebas

A continuación se describen los pasos necesarios para generar un modelo de una aplicación.

1. En el explorador de proyectos, seleccionamos el metamodelo -> metamodel.ecore. En el editor, aparecerá nuestro metamodelo estructurado en forma de árbol.

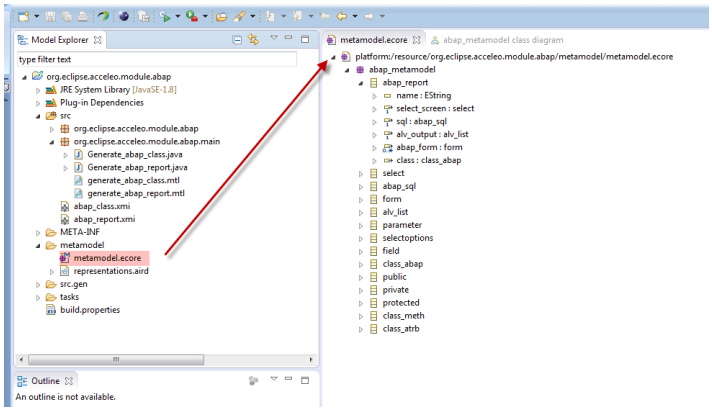


Ilustración 49 Seleccionar el metamodelo metamodelo para modelar aplicaciones SAP

2. Seleccionamos la clase *abap_report* y pulsamos el botón derecho para desplegar el menú contextual, seleccionamos la opción *Create Dynamic Instance…*

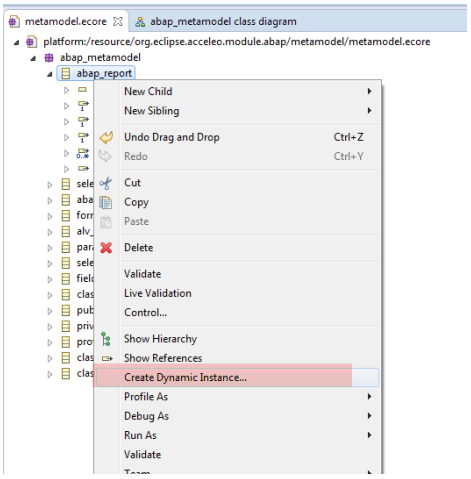


Ilustración 50 Crear una instancia del metamodelo para diseñar el modelo

3. Seleccionamos la carpeta donde se creara la instancia e introducimos un nombre. Los modelos se guardan como archivos XMI (XML Metadata Interchange) como lenguaje de intercambio de modelos utilizando XML.

4. Doble click sobre la instancia creada en el explorador de proyectos
   En el editor se abrirá la instancia en forma de una estructura en árbol

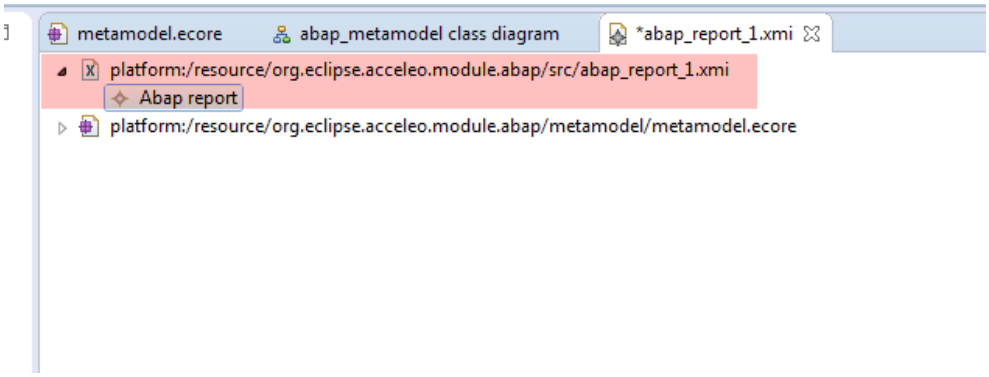


Ilustración 51 Instancia para el modelo en formato XMI

5. El modelo se crea en el primer árbol y podemos ir añadiendo elementos del metamodelo para diseñar nuestro modelo que luego será traducido a código ABAP.
   Con el botón derecho del ratón se abre un menú contextual donde podemos elegir el elemento a añadir en el modelo.

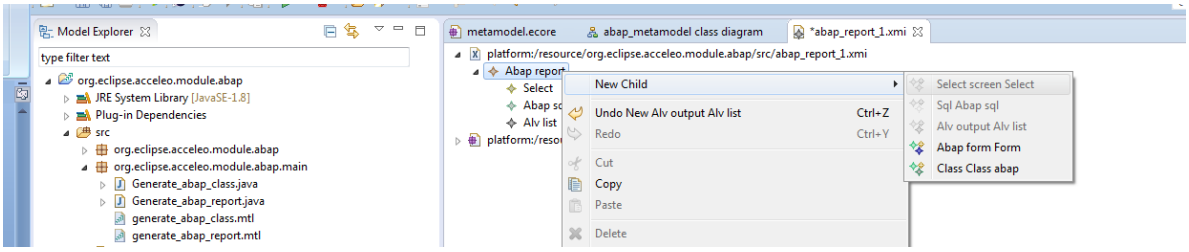


Ilustración 52 Añadir nuevos elementos al modelo

6. En la ventana de propiedades, damos valores a los atributos de cada elemento.

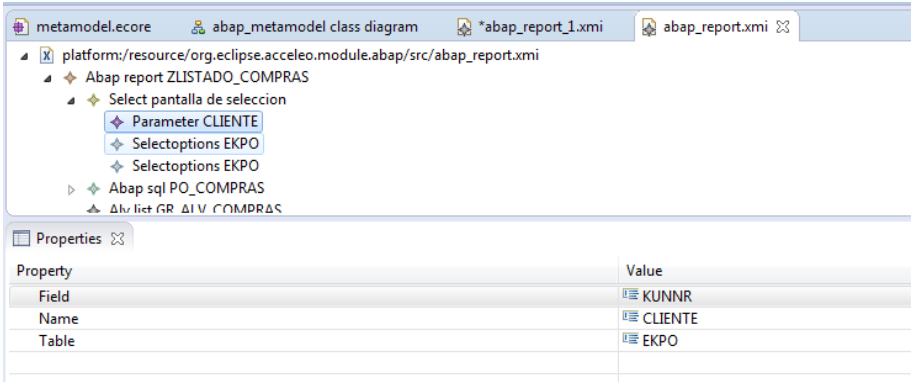


Ilustración 53 Valor de los atributos del elemento del modelo

6.2 Descripción prueba del proceso transformación a código

El modelo creado en el anterior apartado sirve como modelo de entrada para el generador construido con la herramienta ACCELEO. Como se ha descrito en el capitulo anterior, se han creados dos plantillas diferentes para generar el código ABAP IV a partir de un modelo:

- **Generate_abap_report:** Para genera el código ABAP correspondiente a un modelo que represente un programa de tipo ABAP Report
- **Generate_abap_class:** Para generar el código ABAP correspondiente a un modelo que represente una clase ABAP Object.

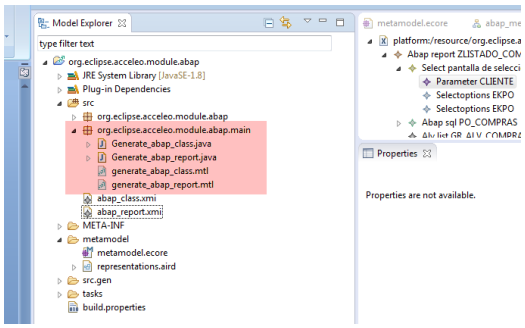


Ilustración 54 Plantillas y generador de código ABAP IV

Para ejecutar las plantillas, desde la perspectiva Acceleo en Eclipse:

1. Ir a *Run -> Run Configurations…*
2. Seleccionamos *Acceleo Aplication*
3. Introducimos las localizaciones de:
4. la clase MAIN de la plantilla (tiene el mismo nombre que la plantilla).
5. El modelo que queremos transformar a código ABAP
6. La localización del fichero de salida que contendrá el código ABAP generado
7. Una vez completada la información pulsar el botón *Run*.
   En la localización seleccionada, se genera aun archivo con el código.

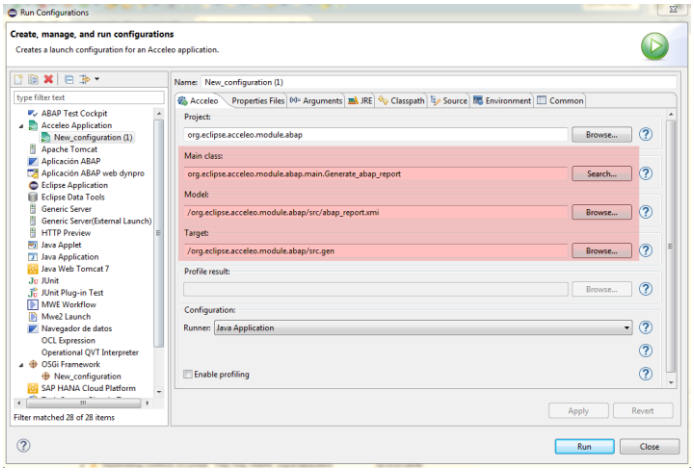
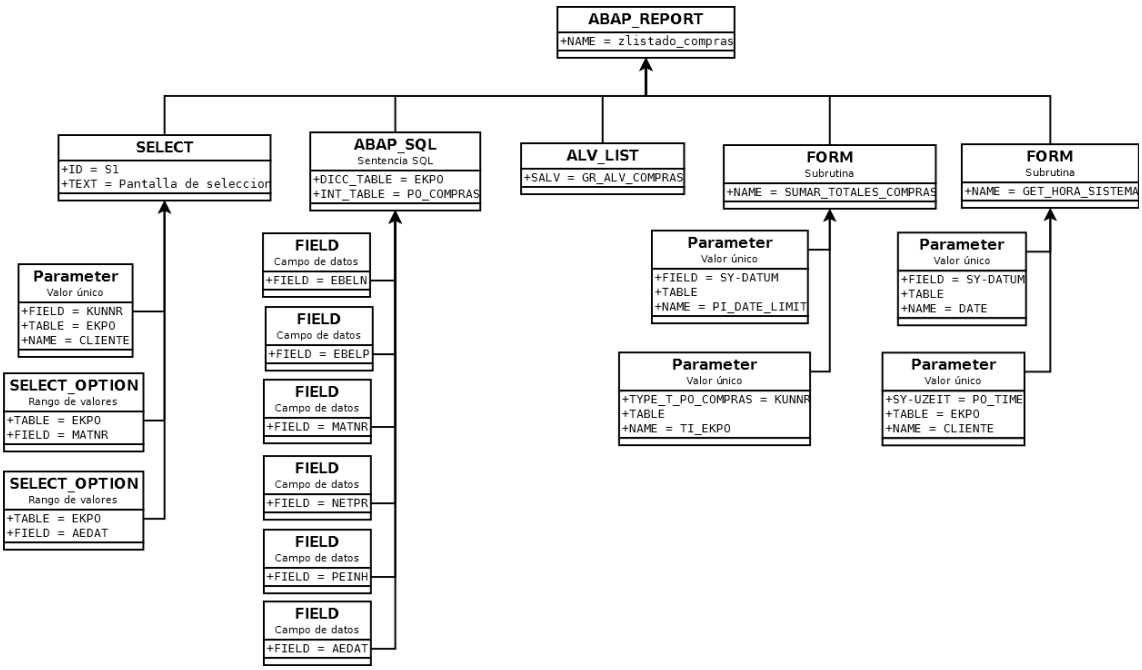


Ilustración 55 Parámetros de entrada del generador de código ABAP IV

6.3 Generación de un reporte ABAP

En la primera prueba realizada, intentaremos generar el código para un reporte ABAP que muestre un listado de pedidos de compra realizados por la empresa dependiendo del periodo y material introducido por el usuario. En la siguiente imagen se muestra el modelo generado con nuestro prototipo de meta-modelo en ECore.



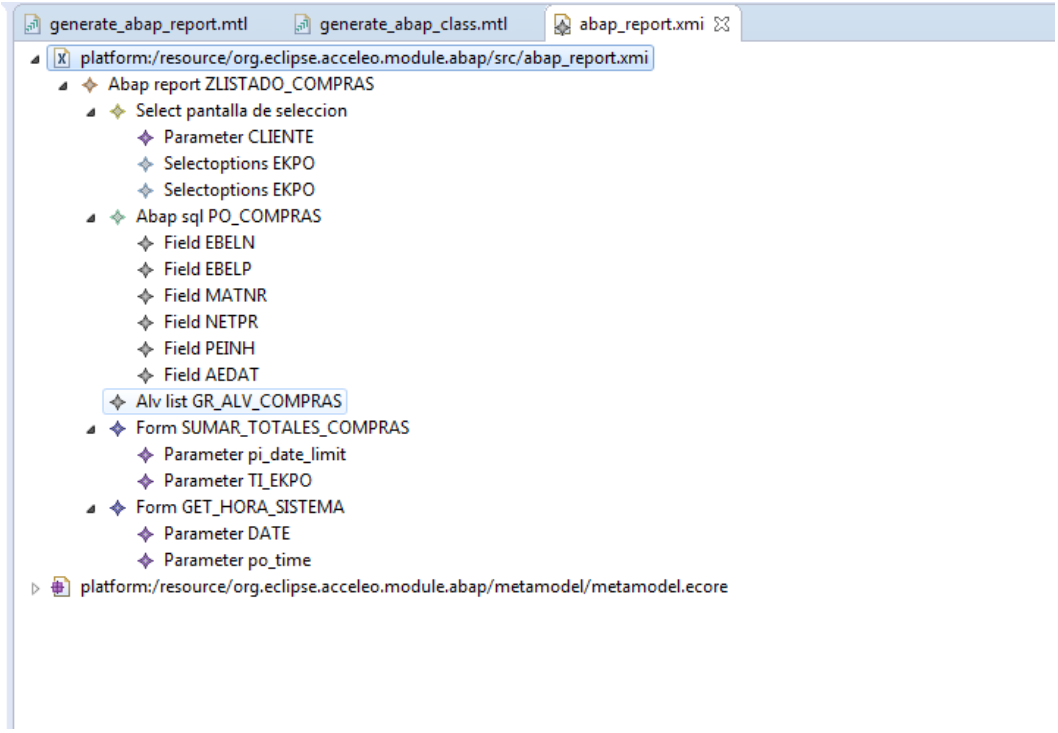


Ilustración 56 Modelo generado para la prueba de reports

Para genera el código, ejecutamos la plantilla generate_abap_report, pasamos como parámetros de entrada nuestro modelo en xmi y la carpeta donde generara el fichero con el código ABAP del reporte.

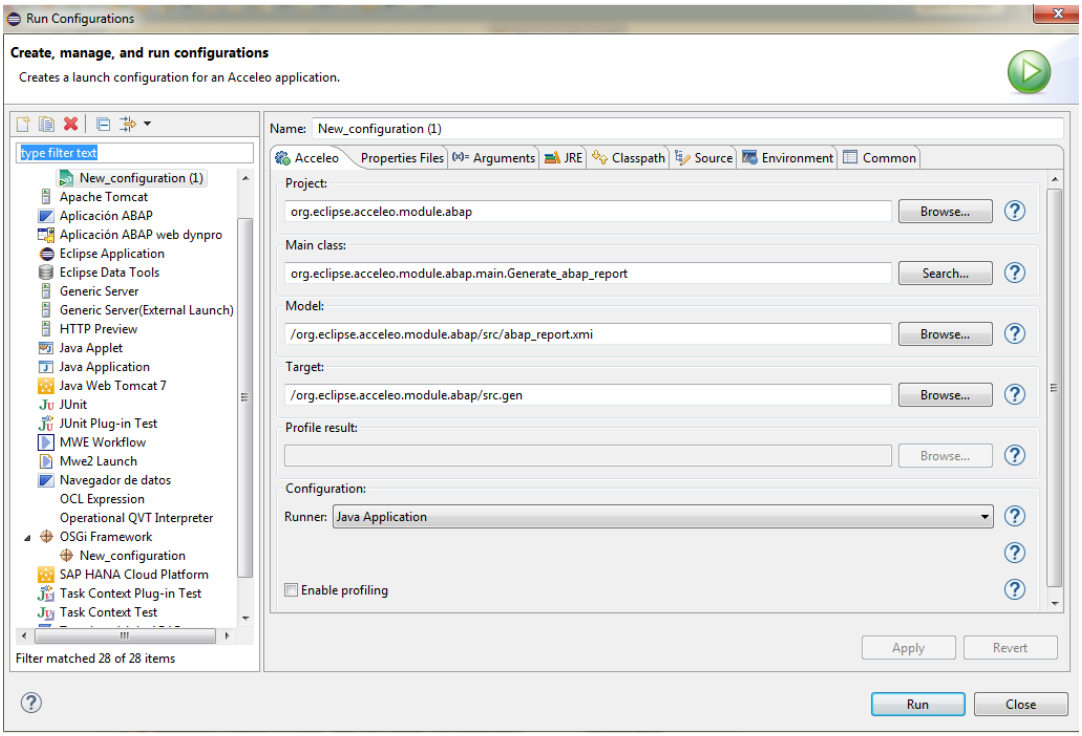


Ilustración 57 Parámetros de entrada para generar el código ABAP IV

Comprobamos el código generado en la carpeta */org.eclipse.acceleo.module.abap/src.gen*

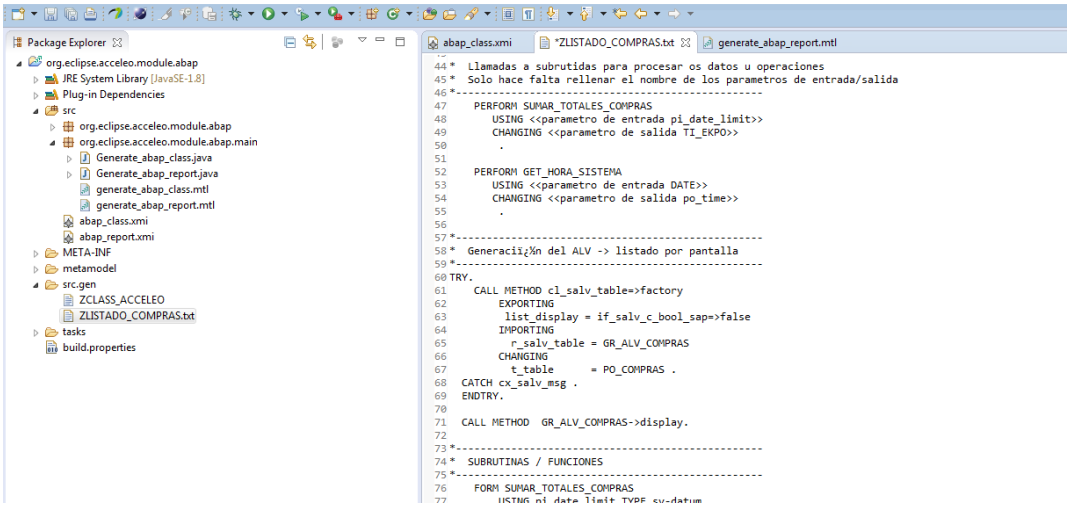


Ilustración 58 Código ABAP IV para el report generado a partir del modelo

Para verificar el código, debemos acceder a un sistema SAP e importar el código desde el editor ABAP del sistema. SAP permite importar código ABAP desde ficheros de texto plano.

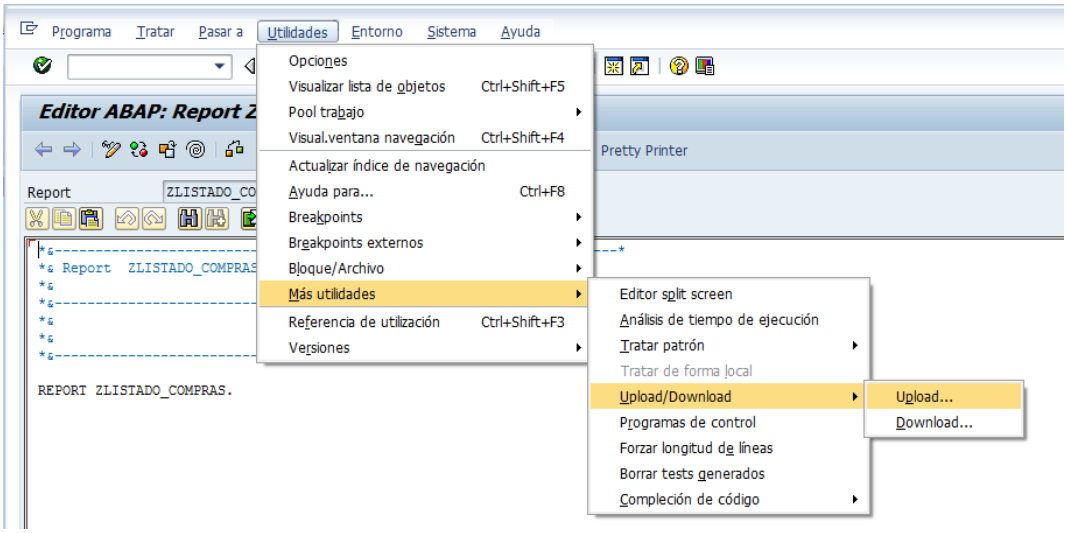


Ilustración 59 Upload del código ABAP IV al sistema SAP R3

Comentamos las subrutinas generadas puesto que no vamos a utilizarlas en la ejecución, solo las hemos añadido para comprobar que es posible generar la llamada y el código de la subrutina.

Activamos el código ABAP para comprobar que no hay ningún error.

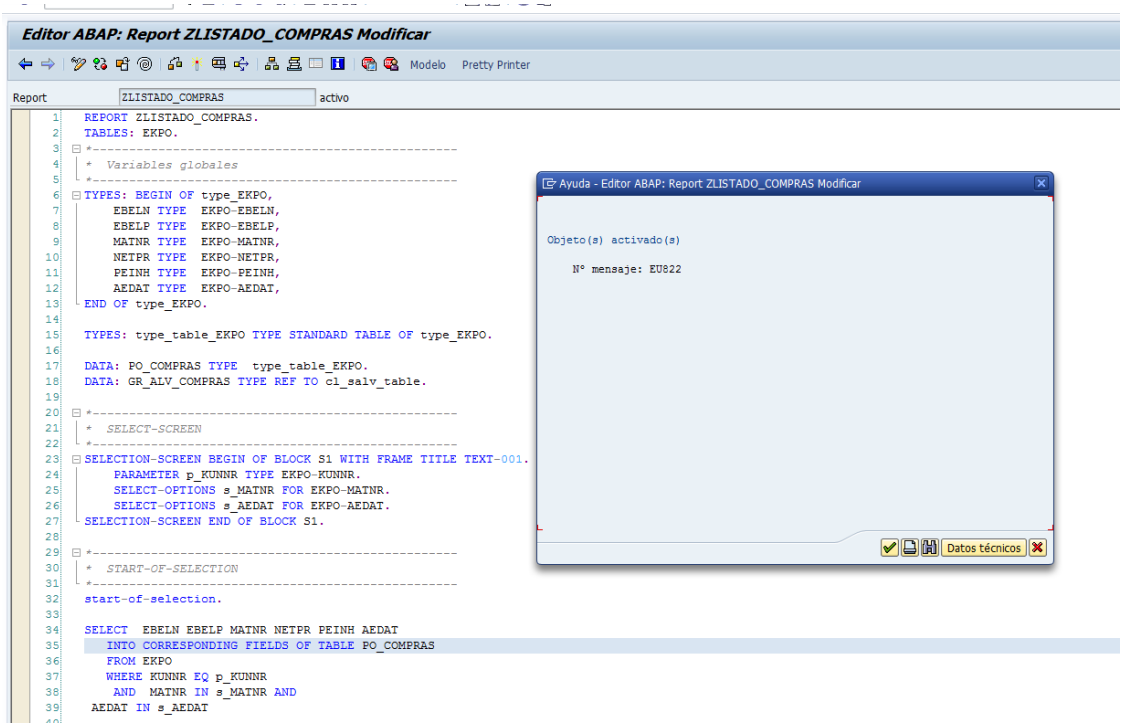
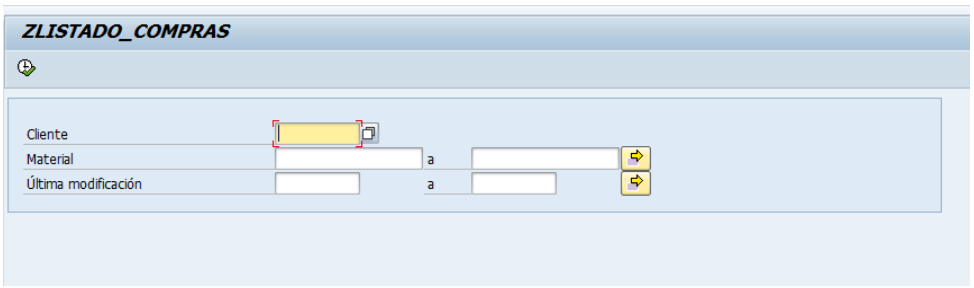


Ilustración 60 Compilar y activar el código ABAP IV en el sistema SAP R3

Ejecutamos el programa y comprobamos que nos permite seleccionar pedidos de compra por material y fecha, mostrando el resultado como una lista por pantalla

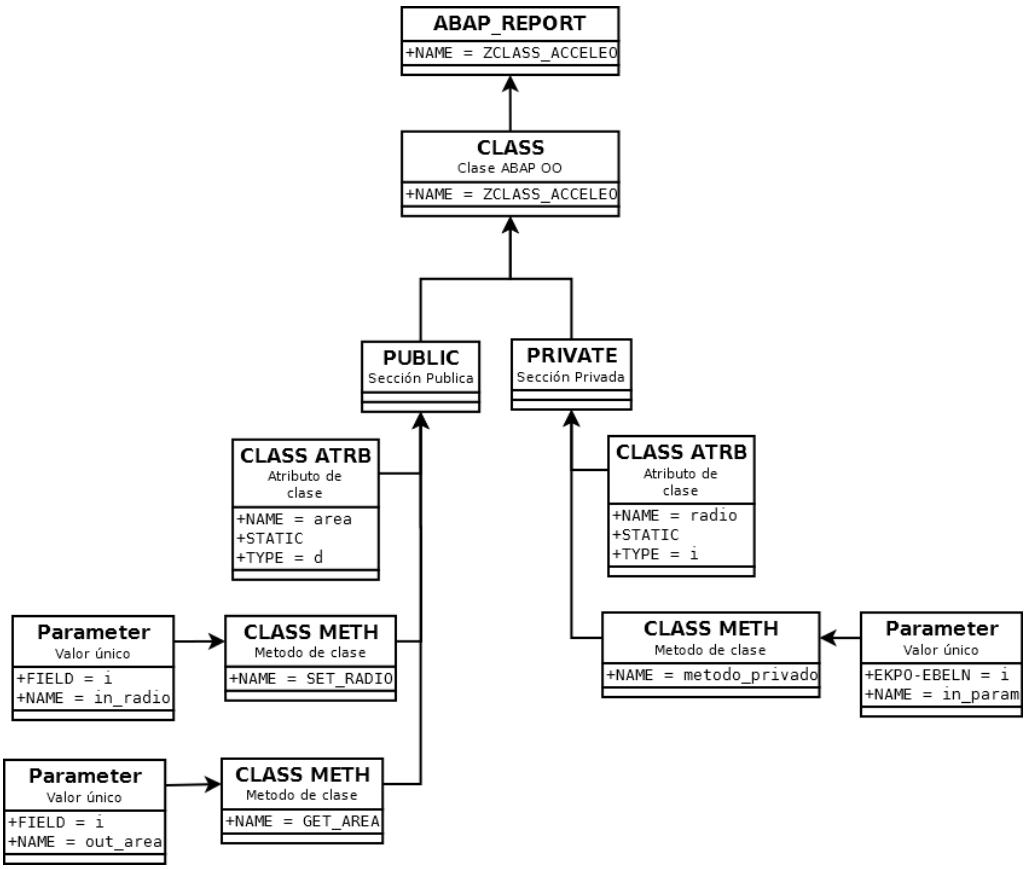


Doc.compras	Posi...	Material	Precio neto	Por	Fecha modif.
4500000...	10	PY-MER01	1,50	1	31.08.2006
45000000...	20	PY-MER02	1,90	1	31.08.2006
45000000...	10	PY-MER01	1,20	1	31.08.2006
45000000...	20	PY-MER02	1,70	1	31.08.2006
45000000...	30	PY-MER03	2,20	1	31.08.2006
45000000...	40	PY-MER04	3,10	1	31.08.2006
45000000...	10	PY-MER01	1,50	1	31.08.2006
45000000...	20	PY-MER02	1,40	1	31.08.2006
45000000...	10	PY-MER01	2,00	1	31.08.2006
45000000...	20	PY-MER02	0,70	1	31.08.2006
45000000...	10	PY-MNIN02	43,00	1	31.08.2006
45000000...	20	PY-MNIN03	30,00	1	31.08.2006
45000000...	10	PY-SERV04	0,50	1	31.08.2006
45000000...	20	PY-SERV04	1,00	1	31.08.2006
45000000...	10	PY-MER05	4,50	1	01.09.2006
45000000...	10	PY-MER01	0,00	0	15.09.2006
45000000...	10	PY-MER02	0,00	0	15.09.2006
45000000...	20	PY-MER01	0,00	0	15.09.2006
45000000...	10	PY-MER01	3,50	1	15.09.2006
45000000...	20	PY-MER02	2,00	1	15.09.2006
45000000...	10	PY-MPR01	11,00	1	15.09.2006
45000000...	20	PY-MPR02	9,50	1	15.09.2006
45000000...	30	PY-MPR03	2,90	1	15.09.2006
45000000...	40	PY-MPR03	2,90	1	15.09.2006
45000000...	50	PY-MPR04	30,00	1	15.09.2006
45000000...	60	PY-MPR04	30,00	1	15.09.2006
45000000...	70	PY-MPR05	1,20	1	15.09.2006
45000000...	80	PY-MPR05	1,20	1	15.09.2006
45000000...	90	PY-MPR06	12,00	1	15.09.2006
45000000...	100	PY-MPR06	12,00	1	15.09.2006
45000000...	10	PY-SEM01	8,00	1	15.09.2006
45000000...	20	PY-SEM02	16,00	1	15.09.2006
45000000...	10	PY-MPR07	22,40	1	17.09.2006
45000000...	20	PY-MPR08	0,22	1	17.09.2006

Ilustración 61 Ejecución y resultado del report

6.4 Generación de una clase local ABAP Objects

Para esta prueba, vamos a generar un modelo que represente una clase en ABAP Objects con sus métodos y atributos,. Esta clase representara un circulo con sus atributos radio y aérea, contendrá dos métodos uno publico que permitirá asignar valor al radio del circulo y otro método privado para calcular su área.



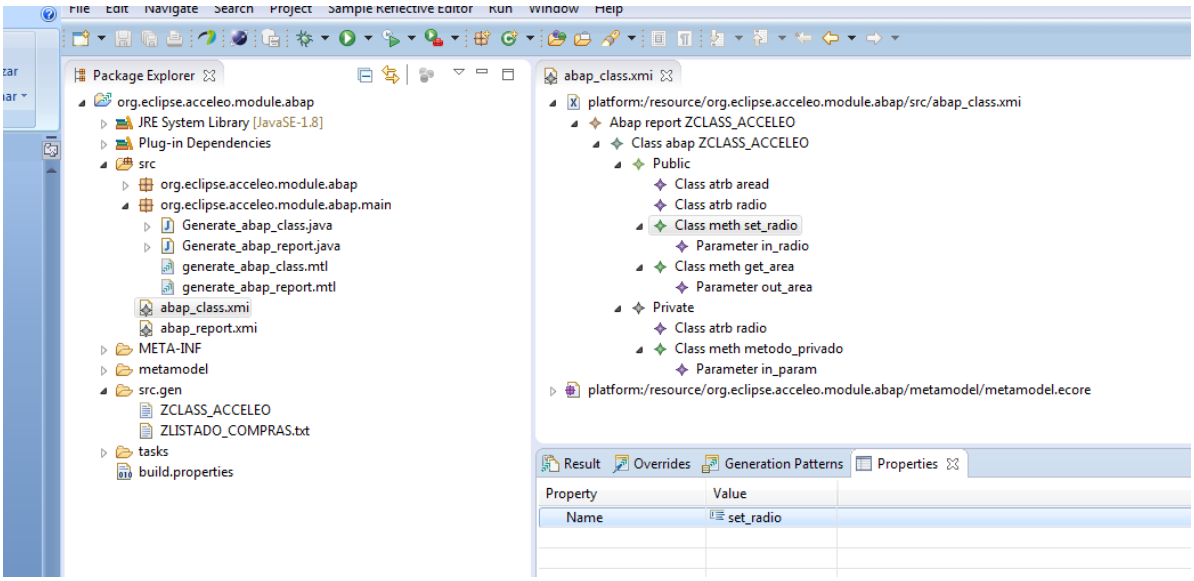


Ilustración 62 Modelo para clase ABAP IV generado con el metamodelo

Para generar el código ABAP IV, seleccionamos y ejecutamos la plantilla *generate_abap_class.*

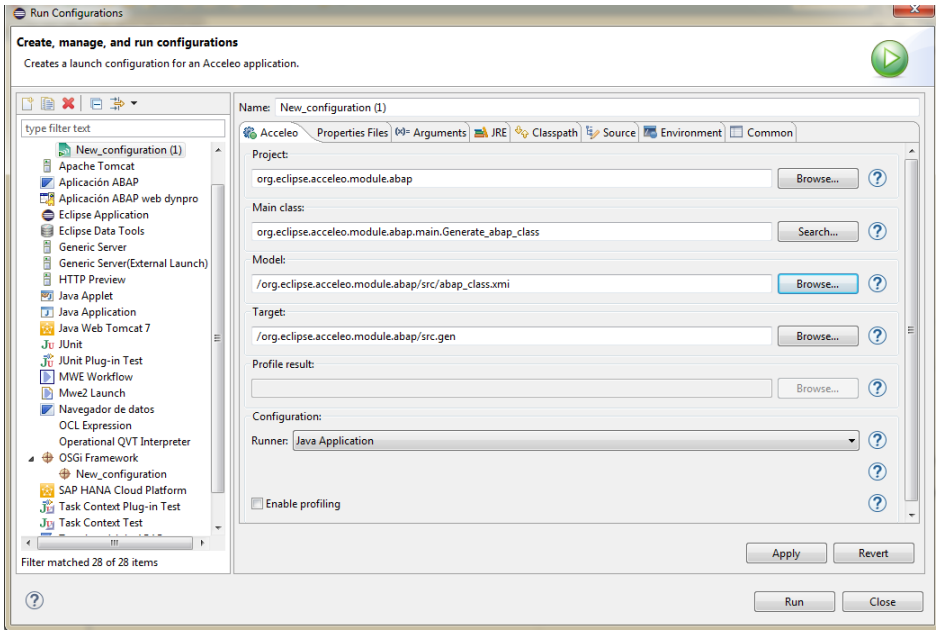


Ilustración 63 Parámetros de entrada para generar el código de la clase ABAP IV

Se genera el ZCLASS_ACCELEO.TXT en la carpeta */org.eclipse.acceleo.module.abap/src.gen*.

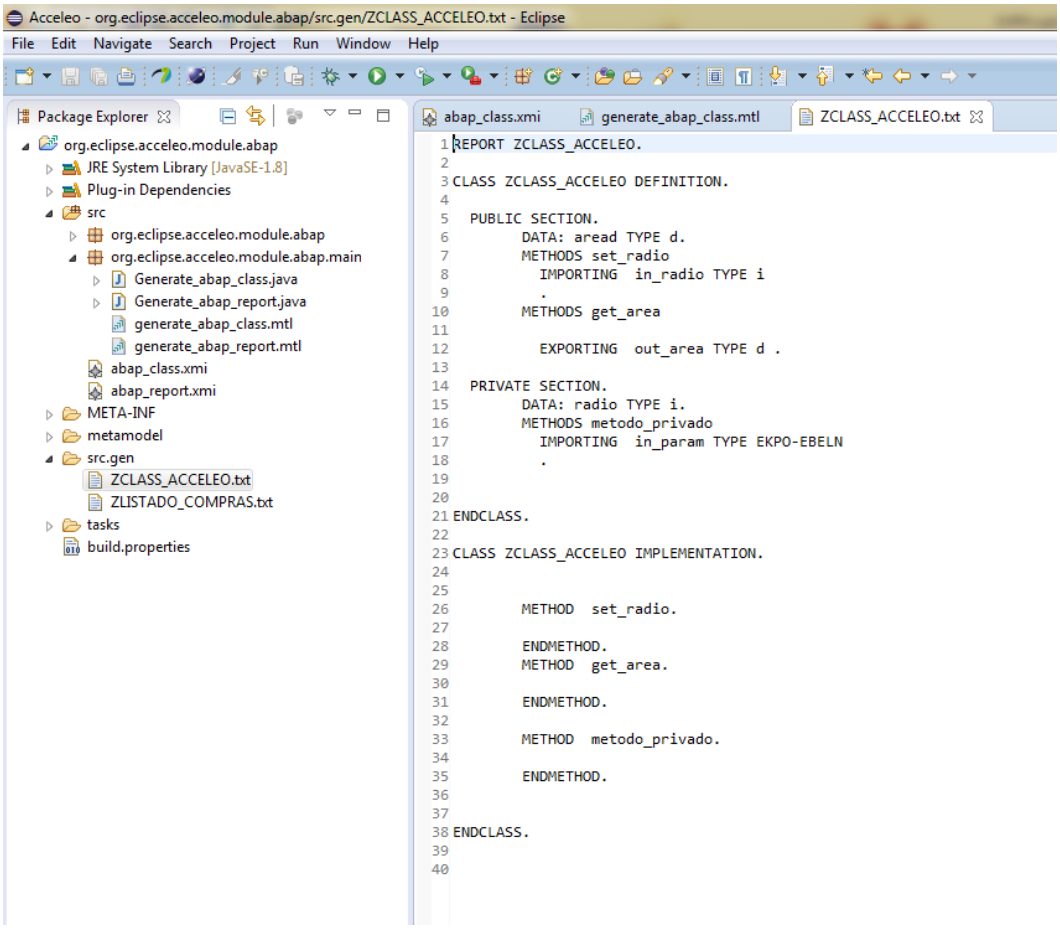


Ilustración 64 Clase ABAP IV generada a partir del modelo

Para verificar el código debemos acceder a un sistema SAP e importar el código desde el editor ABAP del sistema. SAP permite importar código ABAP desde ficheros de texto plano.

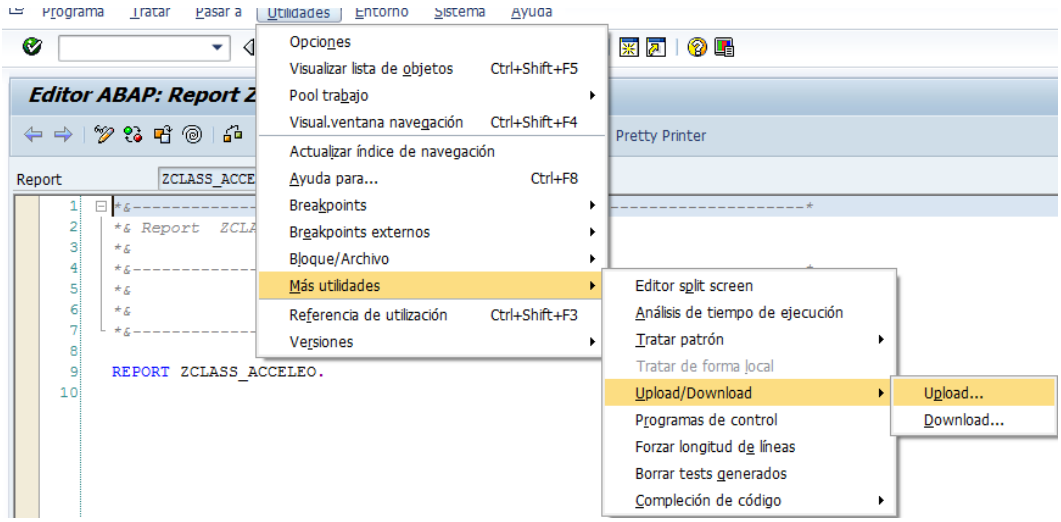


Ilustración 65 Upload de la clase ABAP al sistema SAP R3

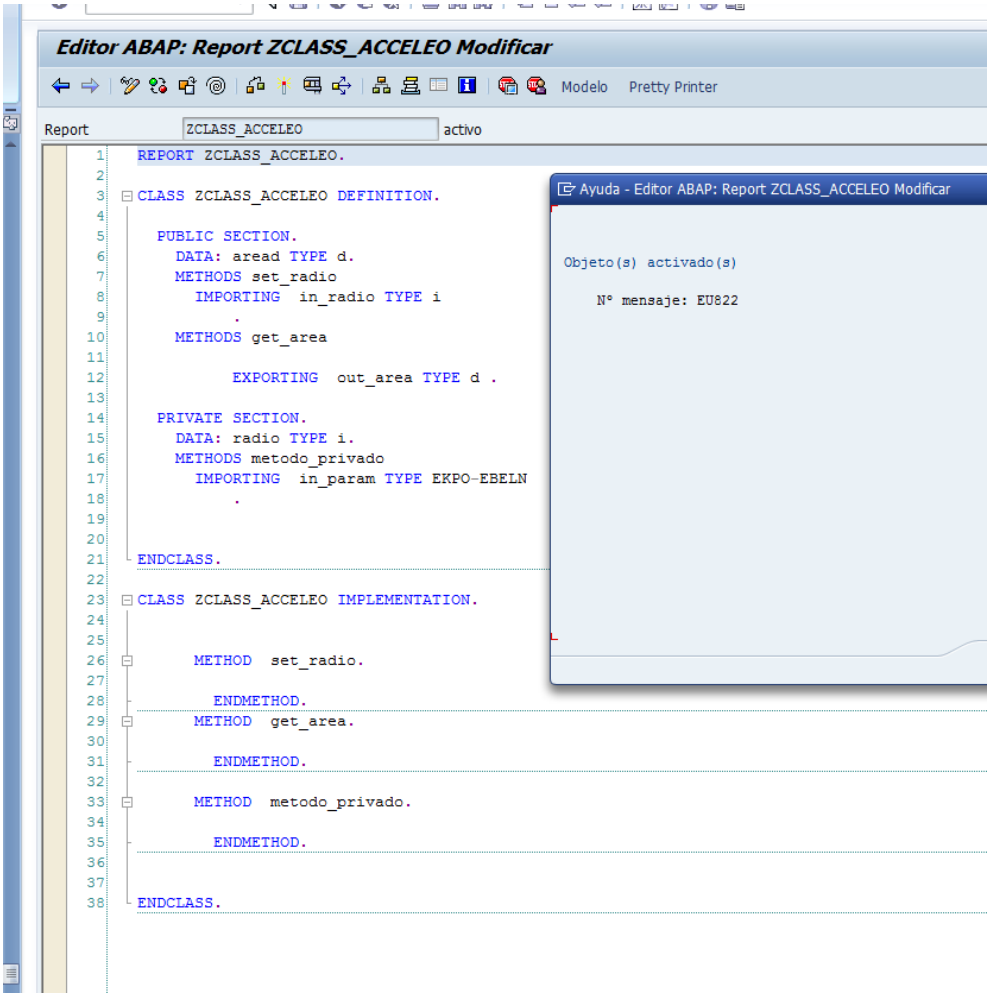


Ilustración 66 Compilación y activación de la clase ABAP IV generada

6.5 Evaluación de la solución desarrollada

A continuación evaluaremos los resultados obtenidos por la solución propuesta en este trabajo de investigación frente a los 2 proyectos actuales descritos anteriormente:

- **UMAP - the UML to ABAP Code Generator**
- **UML2ABAP - UML to ABAP Code Generation**

UMAP - the UML to ABAP Code Generator	
Parámetro de evaluación	Resultado
Capacidad de generar código para programas de tipo ABAP Report	NO
Capacidad de generar código para programas de tipo MODULE POOL	NO
Capacidad de generar clases ABAP	Sí, con herencia e interfaces
Capacidad de generar los modelos a partir del código ABAP	NO
Clases de Metamodelos utilizados	UML

UML2ABAP - UML to ABAP Code Generation	
Parámetro de evaluación	Resultado
Capacidad de generar código para programas de tipo ABAP Report	NO
Capacidad de generar código para programas de tipo MODULE POOL	NO
Capacidad de generar clases ABAP	Sí, con herencia e interfaces
Capacidad de generar los modelos a partir del código ABAP	SI
Clases de meta-modelos utilizados	UML

Solución propuesta en el presente trabajo	
Parámetro de evaluación	Resultado
Capacidad de generar código para programas de tipo ABAP Report	Sí, con las restricciones expuestas
Capacidad de generar código para programas de tipo MODULE POOL	NO
Capacidad de generar clases ABAP	Sí, con las restricciones expuestas
Capacidad de generar los modelos a partir del código ABAP	NO
Clases de meta-modelos utilizados	Ecore

Podemos afirmar que nuestra solución es inferior a la hora de generar clases ABAP a partir de modelos Ecore. Pero solamente porque nuestro meta-modelo no incluye interfaces y herencia y es un punto que puede incluirse en futuras líneas de trabajo al basarse en un meta-modelo Ecore ampliable en comparación a los otros proyectos, que se basa en modelos UML y en concreto únicamente en diagramas de clase UML. No es posible representar un ABAP Report únicamente con programación orientada a objetos ya que, como se ha explicado en el capitulo 3.1.3 Tipos de desarrollos ABAP, las clases no incluyen pantallas de selección, eventos ni la posibilidad de utilizar las instrucciones SUBMIT … AND RETURN y CALL TRANSACTION ya que requieren la existencia de una pantalla de selección en el programa que se invoca para ejecutarlos y pasarle los parámetros de la llamada. La solución es crear clases locales dentro de un programa ABAP report que no puede representarse en diagramas de clase UML [24].

Pero por el contrario nuestra solución si es capaz de generar programas de tipo ABAP Report, aun con las restricciones expuestas en este trabajo, es una característica que no poseen los proyectos actuales y debemos tener en cuenta son programas muy comúnmente demandados en las implantaciones de los sistemas ERP SAP por parte del cliente.

7 CONCLUSIONES Y TRABAJOS FUTUROS

Las Arquitecturas Dirigida por Modelos (MDA) definen la inclusión de nuevas capas de abstracción y la utilización de los modelos como artefacto software que guie los procesos de desarrollo software y el empleo de transformaciones entre dichos modelos que automaticen el proceso.

En este contexto, este trabajo de fin de máster se proponía como objetivo un acercamiento de las arquitecturas dirigidas por modelos a los sistemas ERP, en concreto a sistemas SAP R3. Se ha diseñado un lenguaje de meta-modelo basada en Ecore que permite modelar ABAP Reports y clases ABAP y un generador de código que trasforma estos modelos a código ejecutable sobre un sistema SAP R3. El trabajo realizado y descrito en esta memoria debe tomarse como una prueba de concepto en la que se muestra que es posible la generación automática de código ABAP a partir de los modelos creados con un metamodelo definido que represente procesos de negocio de las empresas u organizaciones.

Por último, se ha evaluado nuestro prototipo frente a otras iniciativas actuales que buscan acercar la arquitectura MDA a los sistemas SAP R3. El resultado nos muestra que un metamodelo basado en Ecore puede llegar a generar un rango mayor de tipos de desarrollos ABAP IV que los generadores de código basado en modelos UML

Aunque se ha alcanzado el objetivo principal de este trabajo de investigación, se abren nuevas líneas de trabajo para su mejora, que lo completen y permitan su uso en el desarrollo de aplicaciones ABAP IV para proyectos de implantación de sistemas SAP R/3:

1. **ABAP Objects:** Ampliar y mejorar la generación de clases ABAP Como se ha visto en la evaluación de la solución aportada, solo podemos generar clases simple de ABAP, una posible línea de investigación seria como ampliar el metamodelo para poder generar clases con herencia, interfaz, etc...

2. **Interfaz grafica:** El hecho de no haber incluido una interfaz grafica para la generación de los modelos provoca que el manejo de la herramienta sea algo complejo, pero el objetivo era demostrar la posibilidad del generar código con un metamodelo Ecore. Para facilitar su uso, una vez que el meta-modelo este completo y sea viable, sería conveniente el desarrollo de una interfaz grafica, posiblemente don la herramienta Graphical Editing Framewok o GEF que es un framework desarrollado para la plataforma Eclipse que se utiliza para crear editores gráficos a partir de un meta-modelo dado [25].

3. **Integración con eclipse IDE:** Partiendo de la línea de trabajo anterior, una vez desarrollado el entorno grafico, para facilitar su adopción por parte de los desarrolladores sería conveniente crear un plugin que permita integrarlo de manera sencilla dentro del entorno de desarrollo Eclipse

4. **Integración de más funcionalidades:** Aunque el trabajo se ha centrado en la generación de código ABAP IV, las dynpros para los desarrollos module pool son descargables desde SAP a ficheros de texto donde se guarda la información de los atributos de la dynpro, sus elementos y posiciones y atributos propios dentro de la dynpro [26]. Podría investigarse si es posible ampliar el metamodelo para poder generar este tipo de ficheros de texto con la información de la dynpro.

Personalmente, quisiera destacar los nuevos puntos de vista que me ha aportado la realización de este proyecto. Llevar a cabo este proyecto me ha enseñado a valorar la importancia de los modelos en el ámbito de un desarrollo software porque no “son los dibujitos que hacemos al terminar de programar” sino que son el reflejo de nuestra forma de entender el problema y de la forma de entender su solución.

8 ANEXO A: Instalación y configuración

A continuación describimos la instalación y configuración del IDE eclipse Luna en su versión Eclipse Modeling Tools, la instalación de las herramientas necesarias para generar los modelos y las transformaciones modelo a texto. Por último, también describimos como importar el metamodelo y el generador de código al IDE Eclipse.

1. Instalar la última versión del java SE JDK disponible. Esto se puede realizar desde: http://www.oracle.com/technetwork/java/javase/downloads/index.html

Ilustración 67 Pagina de descarga del java SE JDK

2. Descargar e instalar el IDE Eclipse Luna en su versión en su versión Eclipse Modeling Tools, de no haber descargado dicha versión, se debe descargar e instalar el proyecto EMF en su última versión disponible.

   Eclipse Modeling Tools en su versión Luna está disponible desde: https://eclipse.org/downloads/packages/release/luna/sr2

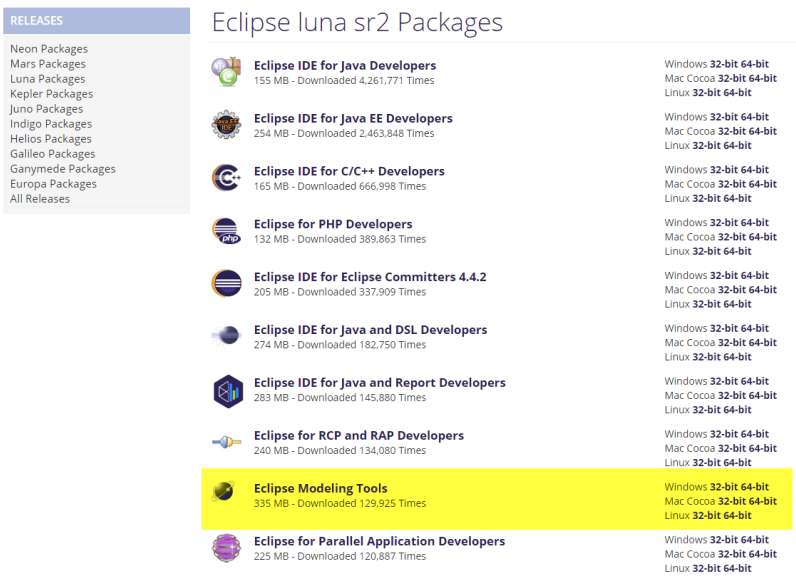


Ilustración 68 Pagina de descarga de Eclipse Luna

3. Descarga e instalar el plugin ABAP DEVELOPMENT TOOLS
   Ejecutar Eclipse EMF e ir al menú *Eclipse -> Help -> Install new Software*
   En el campo *Work with* introducir la siguiente dirección y una descripción:
   - https://tools.hana.ondemand.com/luna

   Seleccionar solo los componentes ABAP_DEVELOPMENT TOOLS FOR ABAP NetWeaver.

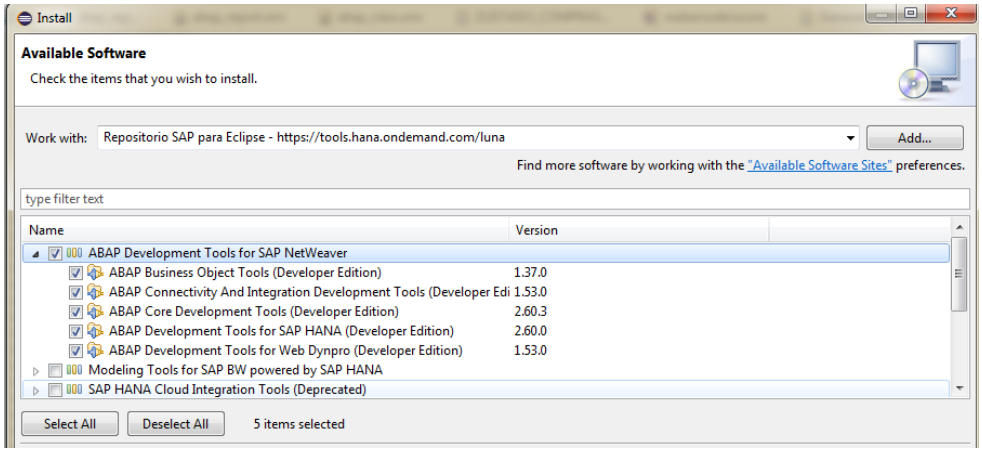


Ilustración 69 Componentes del pluging Abap Development Tools

Aceptar la licencia de uso y reiniciar Eclipse cuando termine la instalación.
Después de reiniciar, aparecerá la ventana de bienvenida del plugin.

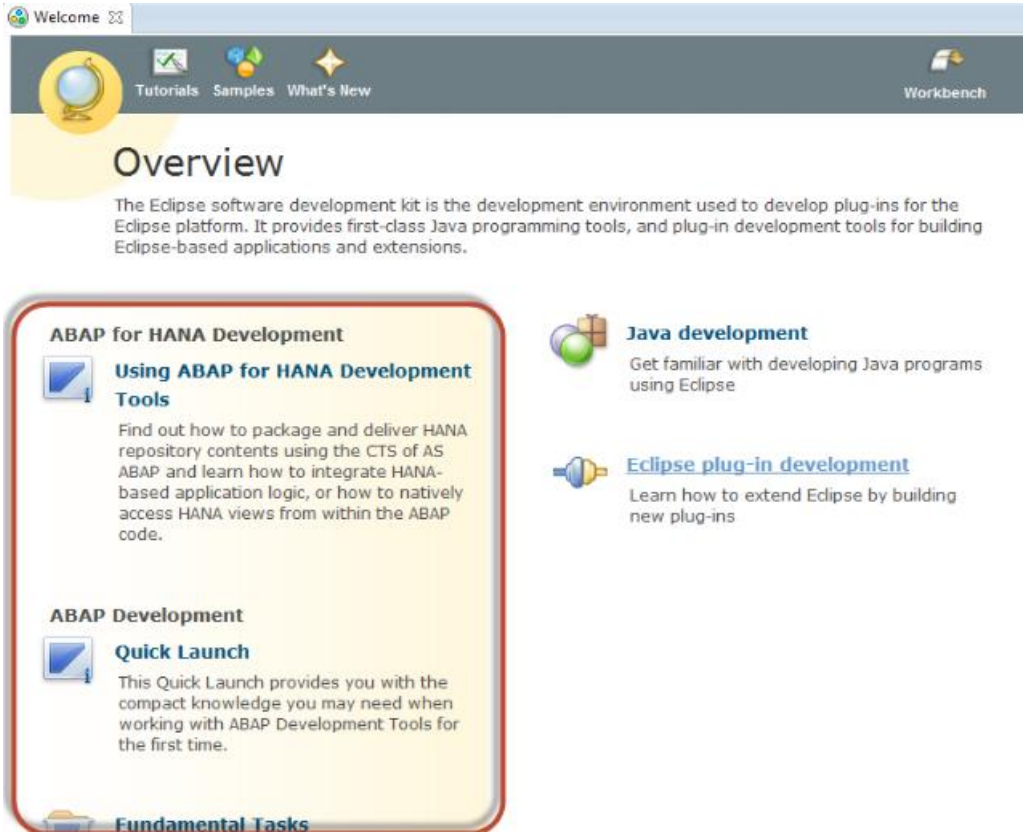


Ilustración 70 Pantalla de presentación de Abap Development Tools

4. Descarga e instalar el plugin ACCELEO
   Ejecutar Eclipse EMF e ir al menú *Eclipse -> Help -> Install Modeling Components.*
   En el apartado MODEL TO TEXT marca el pluging Acceleo
   Pulsar el Botón FINISH para iniciar la instalación.
   Una vez terminada la instalación, Eclipse se reiniciara automáticamente

   OPCIONAL: Desde este mismo menú se puede instalar Eclipse GMF, un plugin un framework de modelado y facilidad de generación de código para construir herramientas y otras aplicaciones basadas en un modelo de datos estructurado.

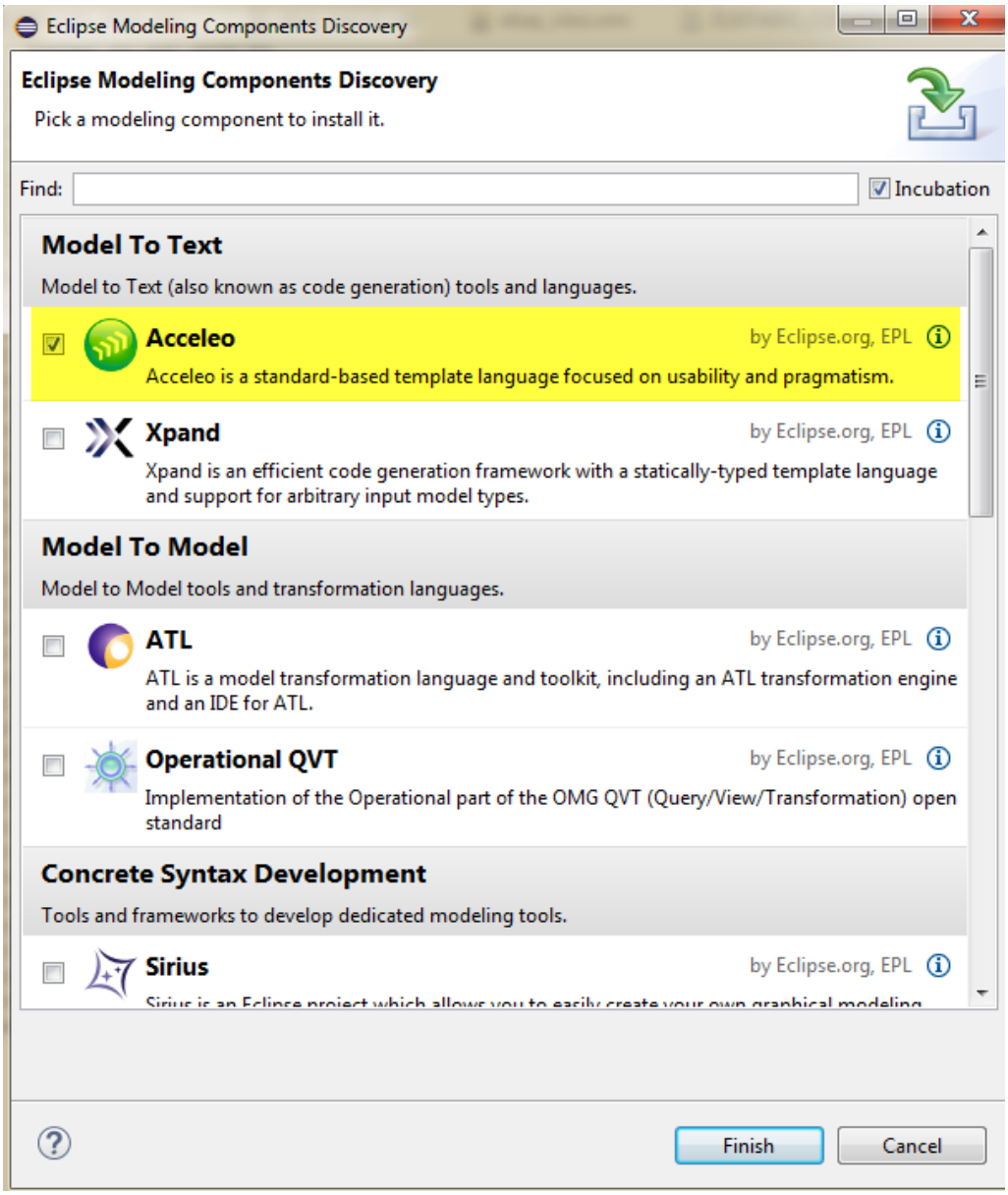


Ilustración 71 Instalación del plugin Acceleo para Eclipse EMF

5. Importar el proyecto **org.eclipse.acceleo.module.abap**
   EL proyecto de Eclipse org.eclipse.acceleo.module.abap que se adjunta con el presente trabajo

   Para importar el proyecto en Eclipse ir al menú *File -> Importç*

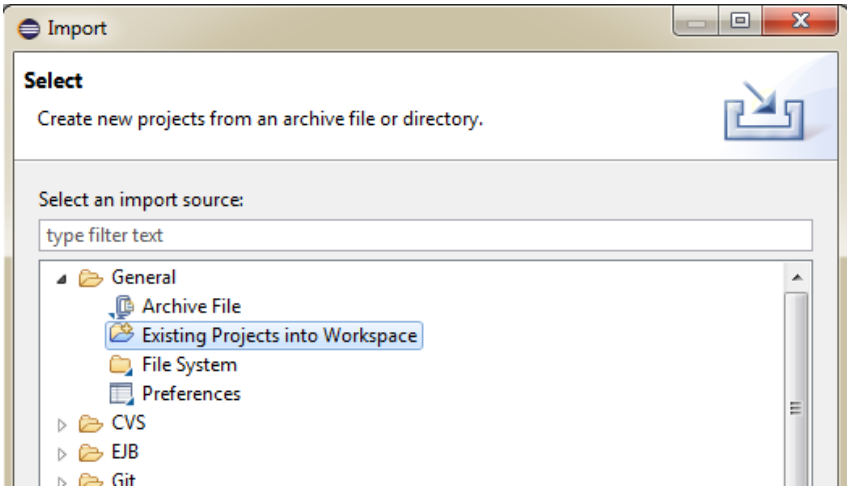


Ilustración 72 Menu para importar proyectos en Eclipse

En la carpeta General, seleccionar *Existing Proyects into Workspace*
Pulsar el botón Browse… e indicar donde se encuentra el proyecto a importar

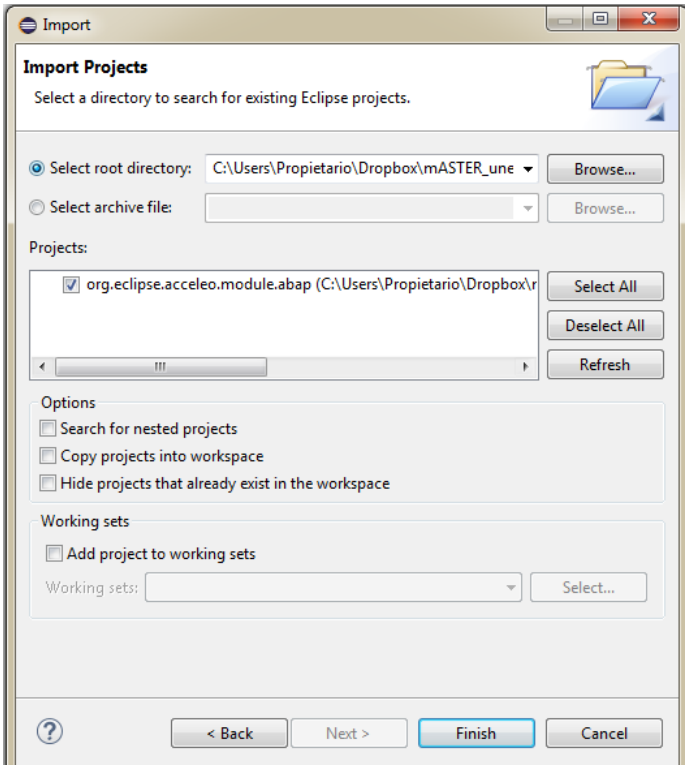


Ilustración 73 Importar el proyecto org.eclipse.acceleo.module.abap

9 BIBLIOGRAFÍA

[1] «TodoSAP - Historia de SAP,» [En línea]. Available: http://todosap.blogspot.com.es/2008/02/historia-de-sap.html.

[2] SAP, «SAP Suppor Portal - Database,» [En línea]. Available: https://support.sap.com/software/databases.html.

[3] «SAP Portal - ABAP Objects,» [En línea]. Available: https://wiki.scn.sap.com/wiki/display/ABAP/ABAP+Objects.

[4] OMG, « MOF Model to Text Transformation Language (MOFM2T), 1.0 formal/2008-01-16,» Object Management Group, 2008.

[5] M.-O. F. (MOF), «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Meta-Object_Facility.

[6] S. IYENGAR, de *XML Metadata interchange (XMI) from UML object models to XML DTDs and Documents*, Unisys Corporation, 1999.

[7] M. V. Z. M. Igor Miletić, Towards Model-Driven Approach for Rapid ERP, Volume 639 of CEUR Workshop Proceedings, page 125-130, 2010.

[8] G. G. Philippe Dugerdil, Model-Driven ERP Implementation, International. Conference on Enterprise Information Systems, 2006.

[9] D. S. Frankel, Model Driven Architecture: Applying MDA to Enterprise Computing, OMG PRESS, 2003.

[10] D. Frankel, «The Rise of Model-Driven Enterprise Systems,» *Business Process Trends,* 2003.

[11] «SAP Composite Application Framework,» [En línea]. Available: https://en.wikipedia.org/wiki/SAP_Composite_Application_Framework.

[12] «Creating Composite Applications with SAP CAF,» SAP, [En línea]. Available: http://help.sap.com/saphelp_nw73/helpdata/en/44/90509cbdc70d06e10000000a11466f/content.htm.

[13] «UMAP - the UML to ABAP Code Generator,» [En línea]. Available: http://www.uml2abap.org/.

[14] «SAPlink,» [En línea]. Available: https://wiki.scn.sap.com/wiki/display/ABAP/SAPlink.

[15] «UML2ABAP,» [En línea]. Available: http://scn.sap.com/community/abap/eclipse/blog/2013/10/17/uml-to-abap-code-generation.

[16] «Papyrus (software),» [En línea]. Available: http://www.eclipse.org/papyrus/.

[17] «ACCELEO,» [En línea]. Available: http://www.eclipse.org/acceleo/.

[18] E. p. Eclipse, «El proyecto Eclipse,» [En línea]. Available: https://eclipse.org/eclipse/.

[19] A. Krawczyk, «ABAP in eclipse vs SE80 comparison - why eclipse wins?,» Abap in Eclipse, 5 Abril 2013. [En línea]. Available: http://scn.sap.com/community/abap/eclipse/blog/2013/04/05/eclipse-vs-se80-comparison--why-eclipse-wins.

[20] T. Franz, «Model-Driven Development in ABAP,» [En línea]. Available: https://scn.sap.com/people/thorstenster/blog/2011/01/17/model-driven-development-in-abap.

[21] I. (. Object Management Group, «Meta Object Facility (MOF) Core Specification, OMG Available Specification, version 2.0,» Enero 206. [En línea]. Available: http://doc.omg.org/formal/2006-01-01.pdf.

[22] J. d. Haan, «; The Enterprise Architect Building an Agile Enterprise,» [En línea]. Available: http://www.theenterprisearchitect.eu/archive/2009/05/06/dsl-development-7-recommendations-for-domain-specificlanguage-design-based-on-domain-driven-design.

[23] J. W. a. A. Kleppe, «The Object Constraint Language: Getting,» Addison-Wesley Professional, 2003.

[24] T. Franz, «Get over ABAP Reports, create object-oriented Selection Screens,» Feb 2011. [En línea]. Available: https://scn.sap.com/people/thorstenster/blog/2011/02/11/get-over-abap-reports-create-object-oriented-selection-screens.

[25] Eclipse, «GEF (Graphical Editing Framework),» [En línea]. Available: http://www.eclipse.org/gef/.

[26] SAP, «Module Pool Notes and download,» [En línea]. Available: https://wiki.scn.sap.com/wiki/display/Snippets/Module+Pool+Notes.

[27] «Rational Rose Modeler,» [En línea]. Available: http://www-03.ibm.com/software/products/es/rosemod.

[28] «Editor des diagramas Dia,» [En línea]. Available: http://dia-installer.de/index.html.es.

[29] W. Strobl, «Análisis de código SAP ABAP,» [En línea]. Available: http://www.bspreviews.com/bspreviewsmagazine/pdf/04/bspreviews04_MAY2013_Calidad_en_los_desarrollos.pdf.

[30] L. N. d. C. d. software, «INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA,» [En línea]. Available: https://www.incibe.es/file/N85W1ZWFHifRgUc_oY8_Xg.

[31] «Eclipse Modeling Tools,» [En línea]. Available: https://www.eclipse.org/downloads/packages/eclipse-modeling-tools/lunasr2.

[32] «SAP Development Tools for Eclipse,» [En línea]. Available: https://tools.hana.ondemand.com/.

[33] OMG, «MDA - The Architecture Of Choice For A Changing World,» [En línea]. Available: http://www.omg.org/mda/.

[34] S. H. PORTAL, « Model Driven Architecture (MDA),» [En línea]. Available: http://help.sap.com/erp2005_ehp_06/helpdata/en/f4/4d7fa0aa8a4acaaf3b62504a89e83c/content.htm.

[35] Wikipedia, «Ingeniería dirigida por modelos,» [En línea]. Available: http://es.wikipedia.org/wiki/Ingenier%C3%ADa_dirigida_por_modelos.

[36] «Eclipse Modeling Framework (EMF),» [En línea]. Available: https://www.eclipse.org/modeling/emf/.

[37] «Graphical Modeling Project (GMP),» [En línea]. Available: http://www.eclipse.org/modeling/gmp/.

10 SIGLAS

ABAP: Advanced Business Application Programming, Lenguaje de programación para el desarrollo de aplicaciones de negocio en sistemas SAP.

API: Application Programming Interface, conjunto de subrutinas, funciones y procedimientos que son ofrecidos por una biblioteca para ser usados por otro software

BAPI: Business Application Programming Interface, son funciones ABAP para permitir la integración de software entre SAP y otros fabricantes de software porque soportan el protocolo Remote Function Call (RFC)

CIM: Computationally-Independent Model, modelo independiente de la computación que caracteriza el dominio del problema.

EMF: Eclipse Modeling Framework, Framework para el modelado de Eclipse para construir herramientas y otras aplicaciones basadas en un modelo de datos estructurado.

ECORE: Meta-modelo de UML para EMF para describir y soportar la ejecución de modelos

ERP: Enterprise Resource Planning**,** es un conjunto de programas que permiten a las empresas ejecutar y optimizar distintos aspectos como los sistemas de ventas, finanzas, operaciones bancarias, compras, fabricación, inventarios y relaciones con los clientes

GMF: Graphical Modeling Framewrok, Framework para el modelado grafico de Eclipse. Permite el desarrollo de entornos gráficos para el modelado de software

IDE: Integrated Development Environment, es una aplicación informática que proporciona servicios integrales para facilitar el desarrollo de software

MDA: Model Drive Architecture, arquitectura dirigida por modelos

OMG: Object Management Group es un consorcio, formado en 1989, dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA y BPMN.

PIM: Platform-Independent Model, modelo independiente de la plataforma describen una solución de software que no contiene detalles de la plataforma concreta en que la solución va a ser implementada

PSM: Platform Definition Model, modelos específicos de la plataforma que contienen los detalles de la plataforma o tecnología con que se implementará la solución

RCP: Remote Function Call, llamada de función remota es un procedimiento para intercambiar datos entre un cliente y un servidor.

SAP: Systems, Applications, Products, Compañía informática con sede en Walldorf, Alemania. sus productos incluyen SAP ERP, SAP Business Warehouse (SAP BW), SAP BusinessObjects software, y SAP HANA

UML: Unified Modeling Language es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG

XMI: XML Metadata Interchange, XML de Intercambio de Metadatos es una especificación para el Intercambio de Diagramas.

XML: eXtensible Markup Language, lenguaje de marcas Extensible. Es un metalenguaje extensible de etiquetas para estructurar documentos grandes.