

Procesadores de Lenguajes 2

# Construcción de editores de modelos con EMF

Curso 2013-2014

Iván Ruiz Rube  
Departamento de Ingeniería Informática  
Escuela Superior de Ingeniería  
Universidad de Cádiz



# Contenidos

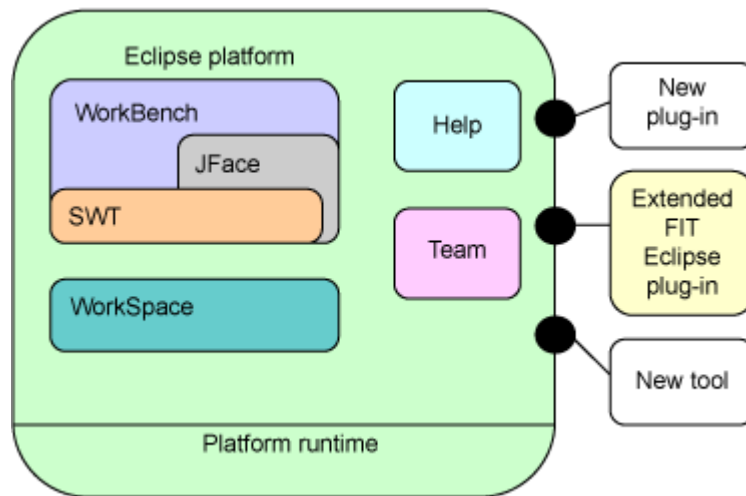
- Introducción
- Generar clases de soporte
- Despliegue de editores como plug-ins
- Despliegue de editores como productos

CONSTRUCCIÓN DE EDITORES DE MODELOS CON  
EMF



# INTRODUCCIÓN

# Desarrollo sobre Eclipse



- Eclipse se compone de una base de código (kernel) y un conjunto de extensiones adicionales (plug-ins).
- Estas extensiones de la GUI se desarrollan con JFace (un framework MVC sobre SWT)
- Las herramientas de Eclipse Modeling Project, se encargarán de hacerlo por nosotros.



# Eclipse Modeling Framework

- Ya vimos como EMF permite diseñar metamodelos utilizando el lenguaje Ecore.
- Ahora, vamos a generar sencillos editores de modelos reflexivos basados en una estructura de árbol.
- El desarrollo del editor se basará en una sucesiva transformación de modelos.

# Pasos para crear un editor de modelos

1. Elaborar y validar un metamodelo con Ecore
2. Generar clases Java de soporte al metamodelo:
  - factorías, interfaces, listeners, etc.
3. Desplegar editor
  - Plug-in de Eclipse: Eclipse PDE
  - Aplicación independiente: Eclipse RCP
  - Aplicación web: Eclipse RAP

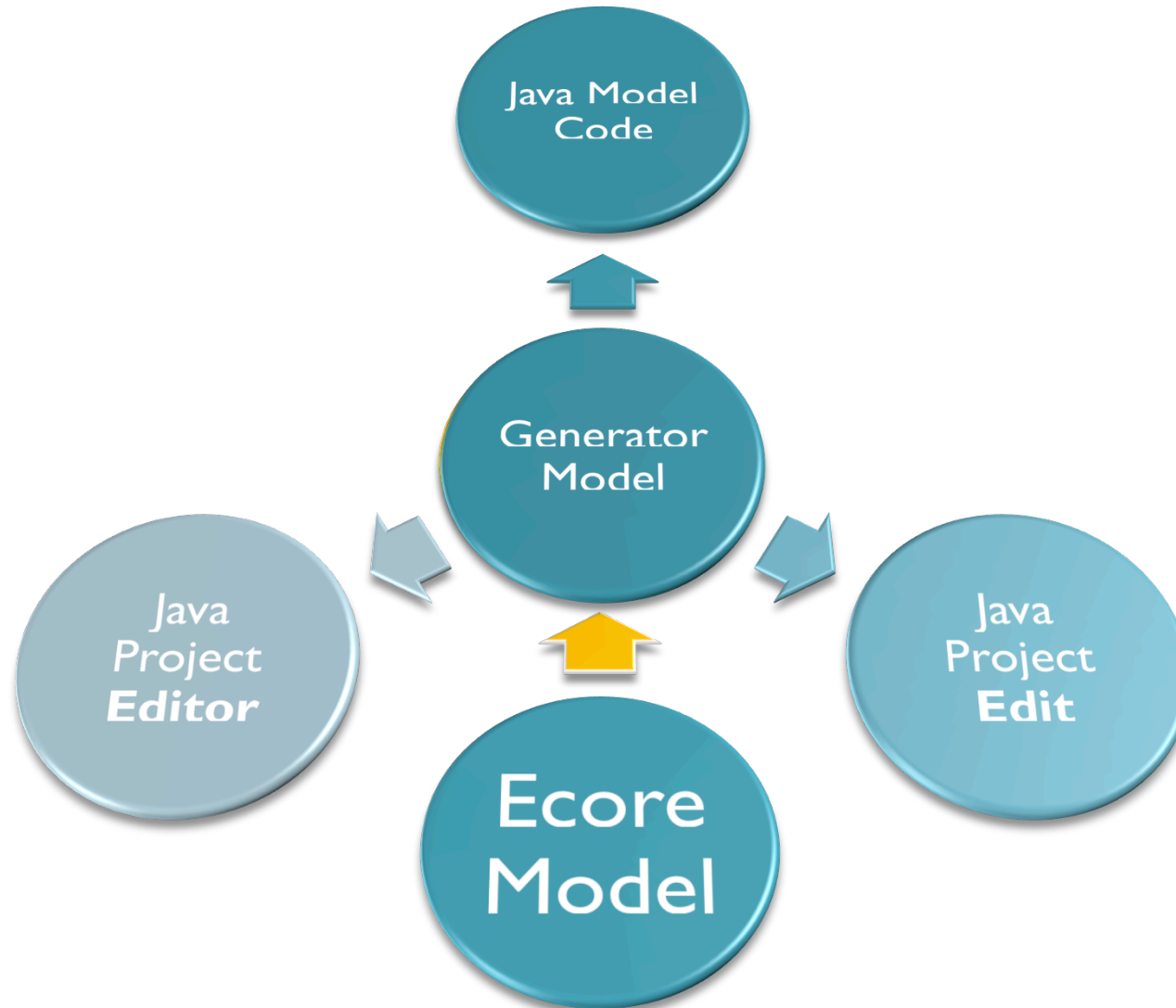


CONSTRUCCIÓN DE EDITORES DE MODELOS CON  
EMF



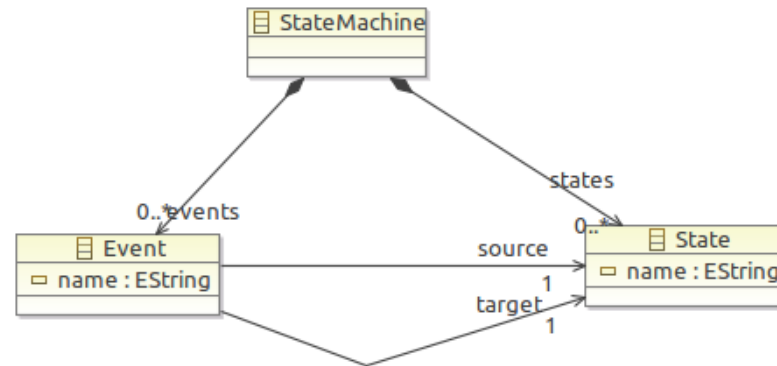
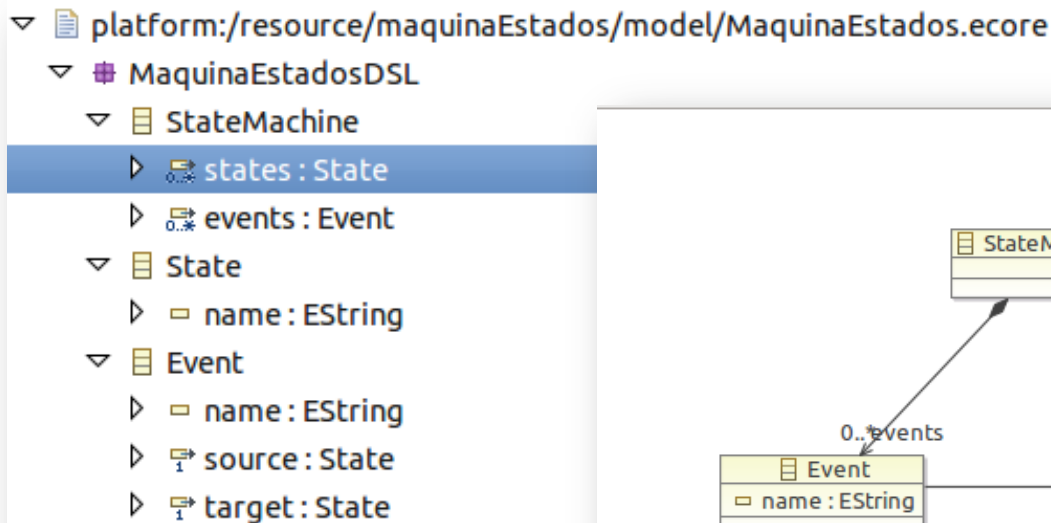
**GENERAR CLASES DE  
SOPORTE**

# Transformación de modelos



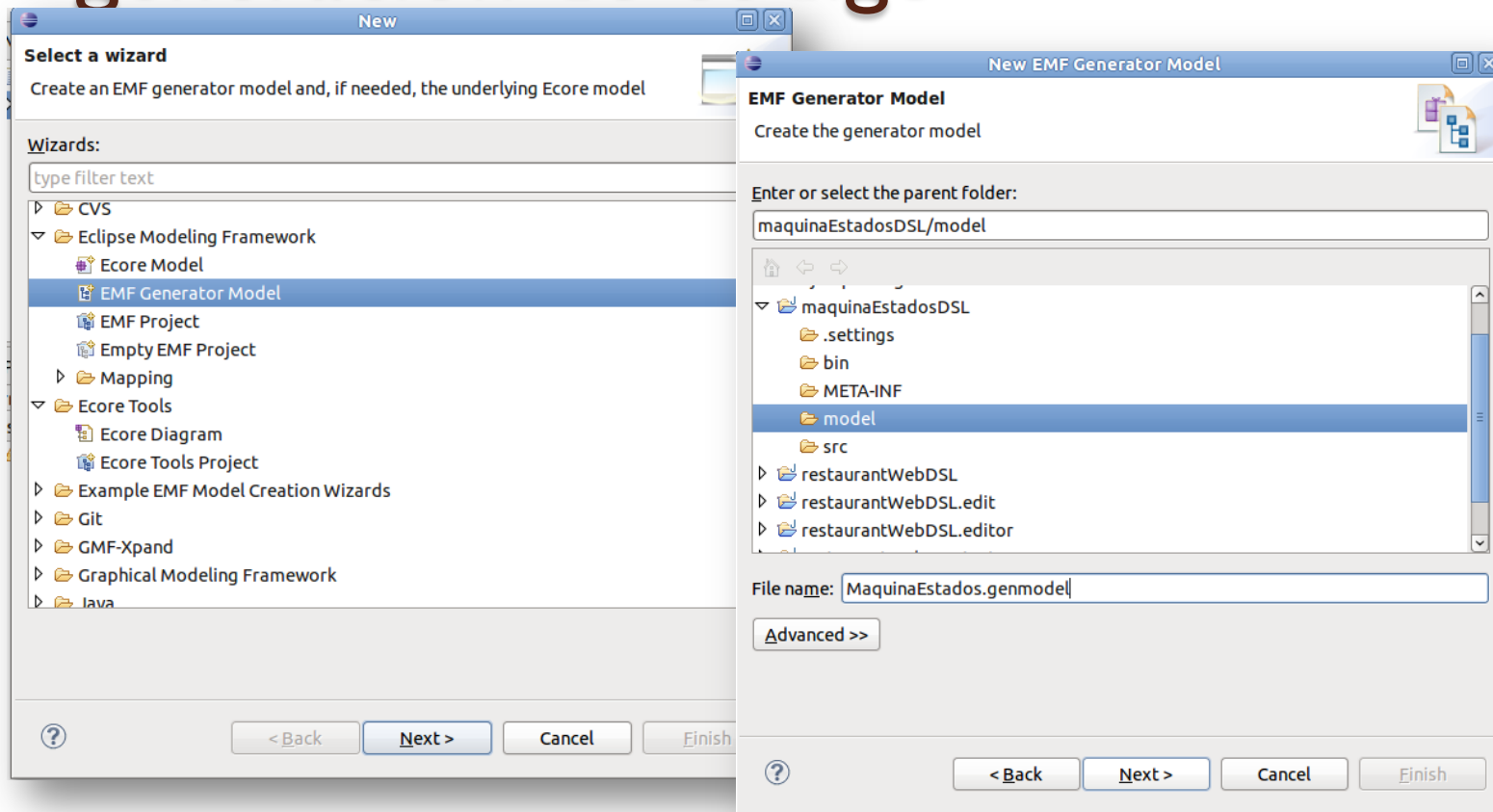


# Desarrollo del metamodelo



Diseñamos nuestro metamodelo Ecore con el editor de metamodelos de EMF

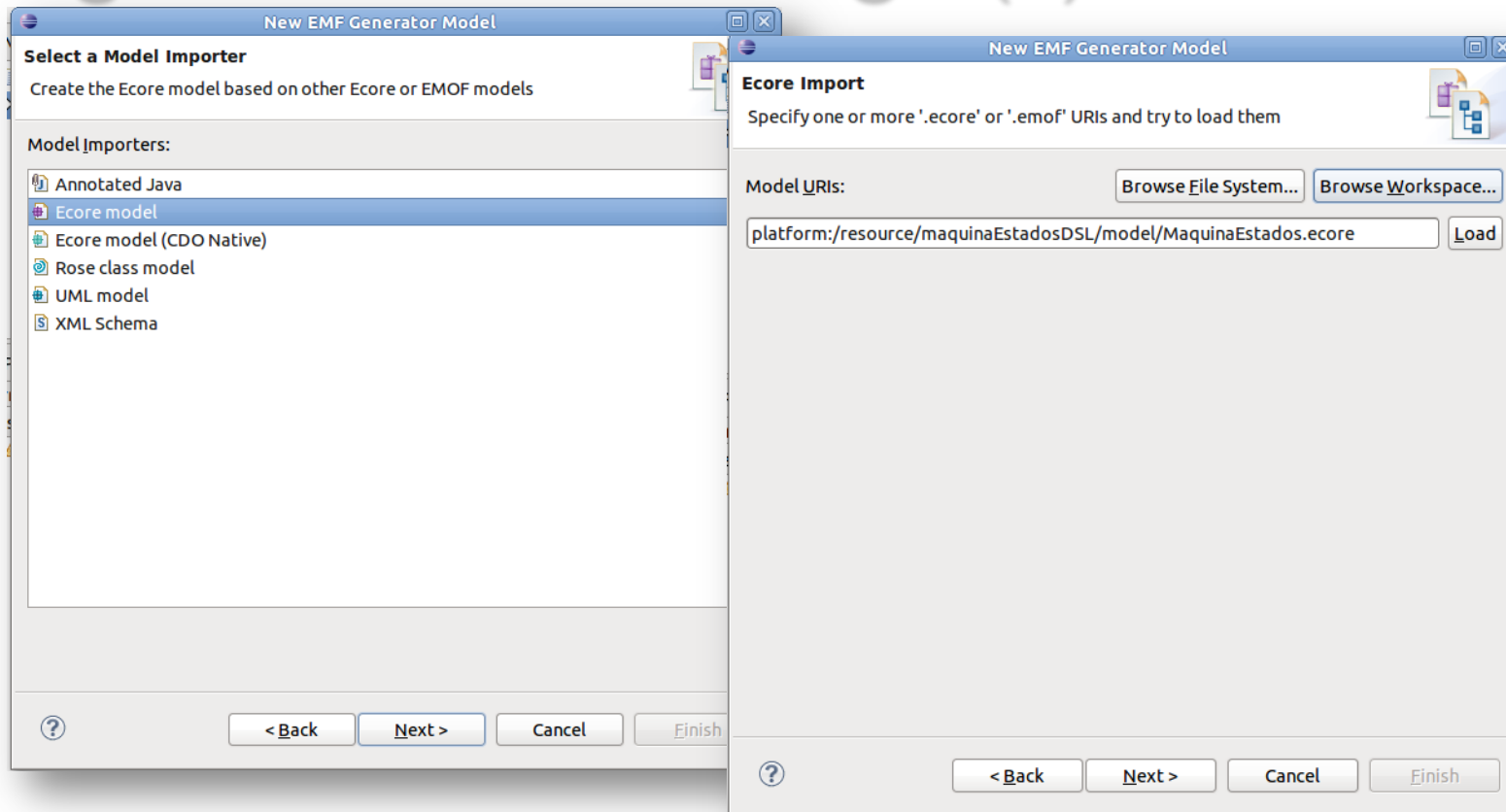
# Creación de un modelo de generación de código



File → New → EMF Generator Model

El modelo de generación de código lo llamaremos igual que el modelo Ecore, pero con extensión “.genmodel”

# Creación de un modelo de generación de código (II)



El modelo de generación de código lo crearemos a partir del modelo Ecore diseñado previamente.

# Creación de un modelo de generación de código (III)

The screenshot shows the Eclipse IDE interface. On the left, the 'New EMF Generator Model' dialog is open, showing the 'Package Selection' step. The 'Root packages' section has a table with one entry: 'maquinaestados' with the file name 'MaquinaEstados.ecore'. The 'Referenced generator models' section is empty. On the right, the 'MaquinaEstados.genmodel' project is open, showing a tree view of the model structure. The tree view shows the following structure:

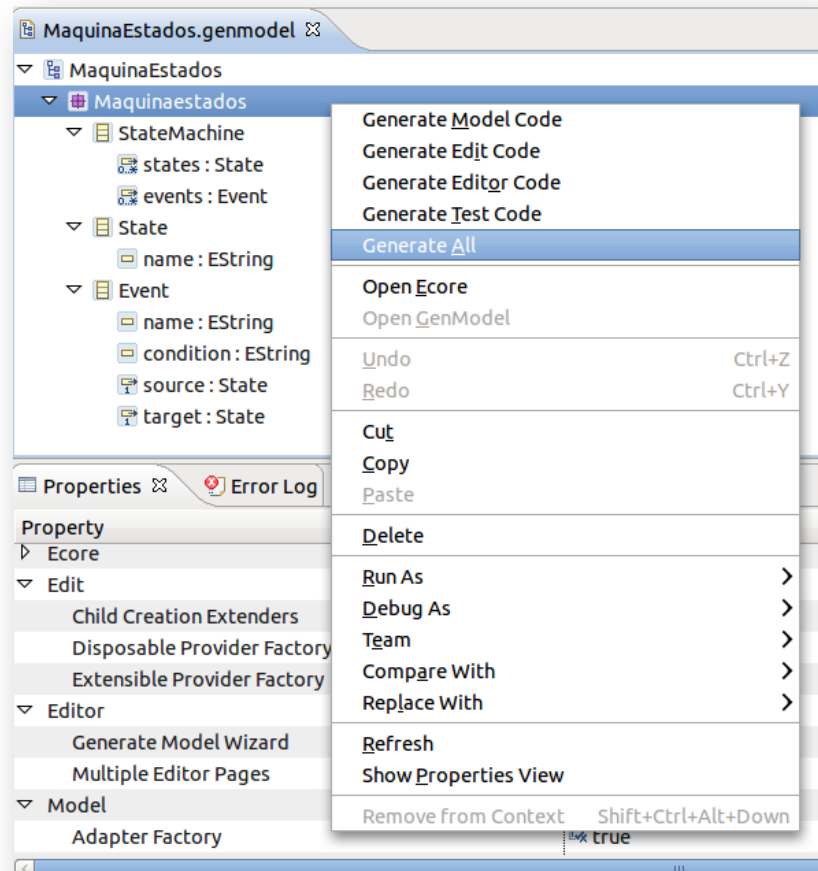
- MaquinaEstados
  - StateMachine
    - states : State
    - events : Event
  - State
    - name : EString
  - Event
    - name : EString
    - condition : EString
    - source : State
    - target : State

Below the tree view, the 'Properties' view is open, showing a table of properties and their values:

Property	Value
Ecore	
Edit	
Child Creation Extenders	false
Disposable Provider Factory	true
Extensible Provider Factory	false
Editor	
Generate Model Wizard	true
Multiple Editor Pages	true
Model	
Adapter Factory	true

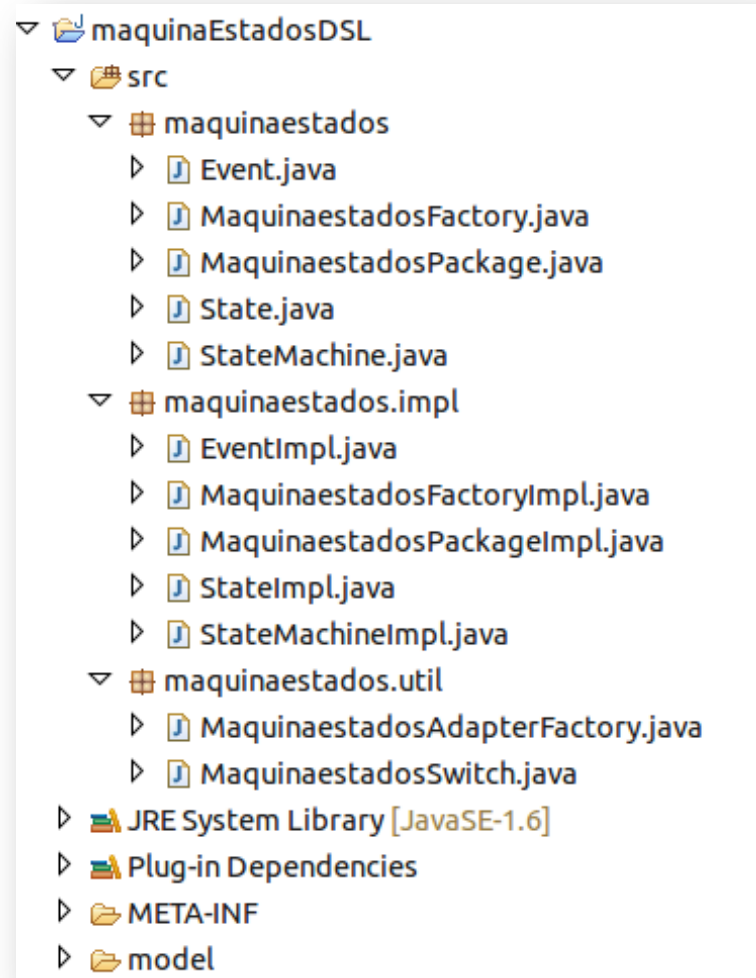
Una vez seleccionado el EPackage del modelo Ecore, automáticamente (via M2M) se construye un nuevo modelo similar al anterior, aunque con información adicional.

# Generación del tree-editor



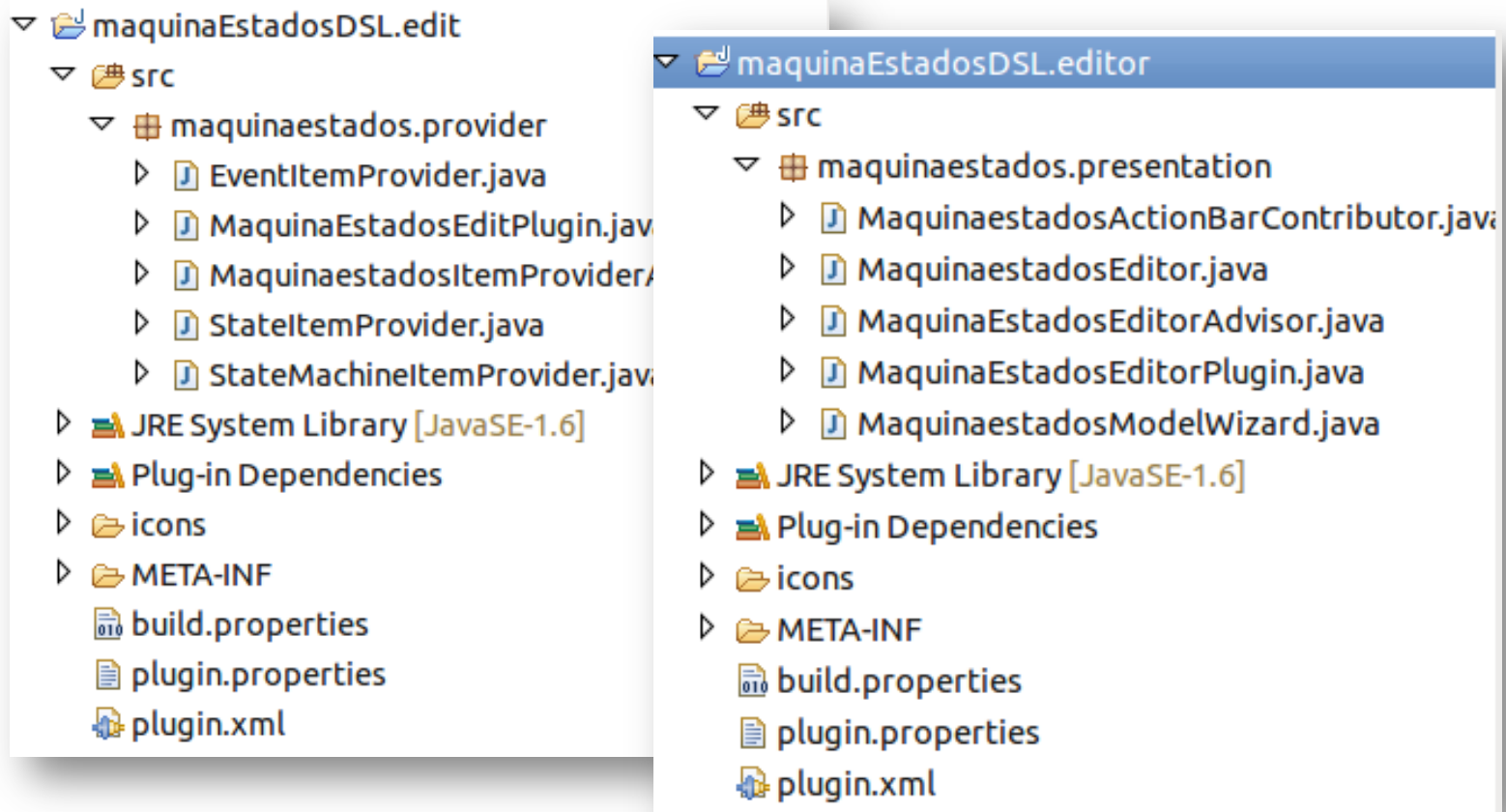
Partiendo de nuestro modelo de generación de código, generaremos el código fuente necesario para implementar nuestro editor de modelos.

# Java Model Code



- Implementación en Java de los elementos del metamodelo.
- Se generan interfaces, clases de implementación, factorías y otras clases de soporte necesarias para construir editores.
- También se genera un proyecto *Tests*, con casos de prueba en Junit.

# Edit / Editor



El proyecto *Edit* incluye las clases necesarias para la visualización y edición de los modelos. *Editor* proporciona los elementos requeridos para la interfaz de usuario de nuestro editor de modelos.

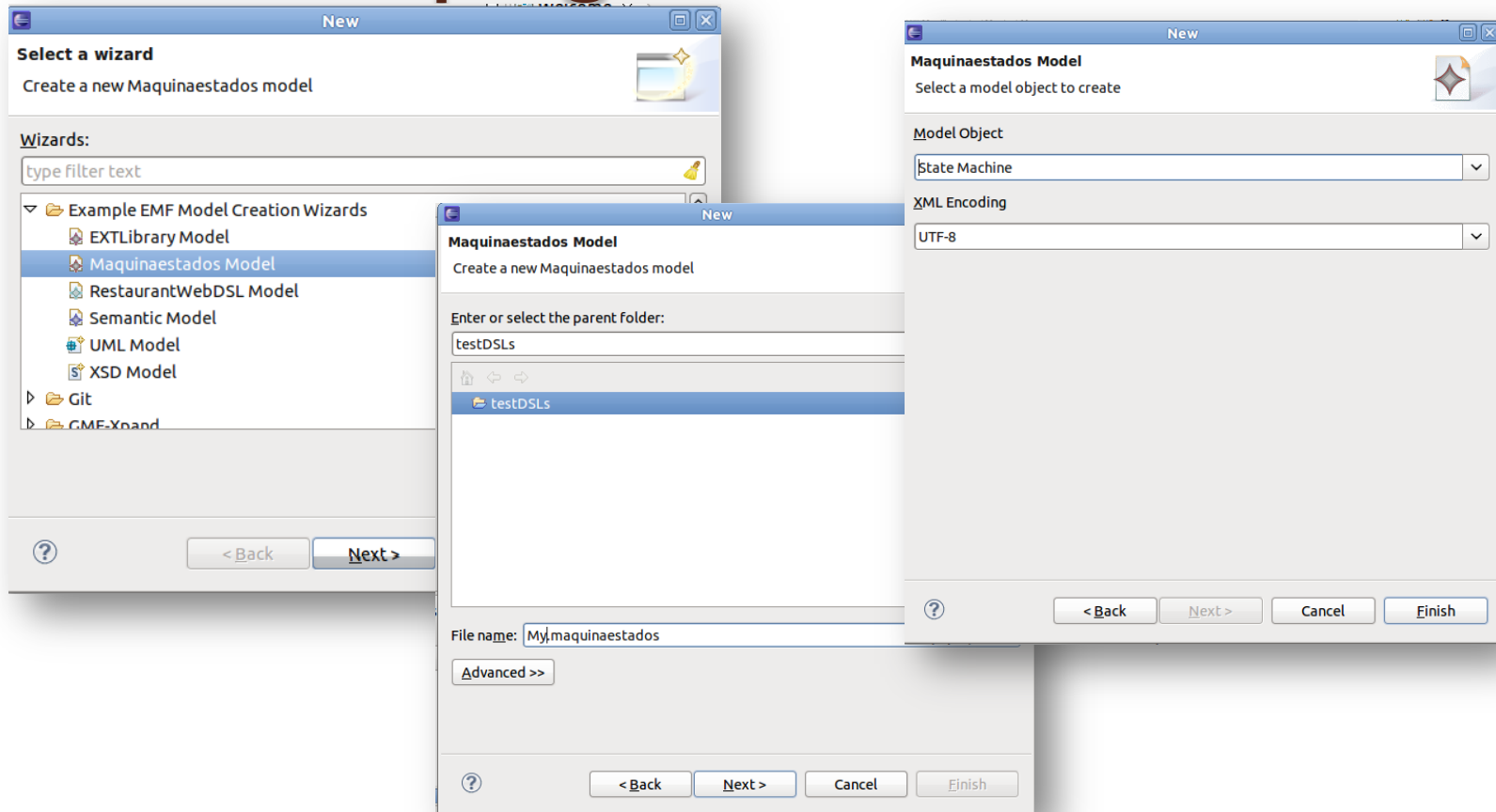
CONSTRUCCIÓN DE EDITORES DE MODELOS CON  
EMF



# DESPLIEGUE DE EDITORES COMO PLUG-INS

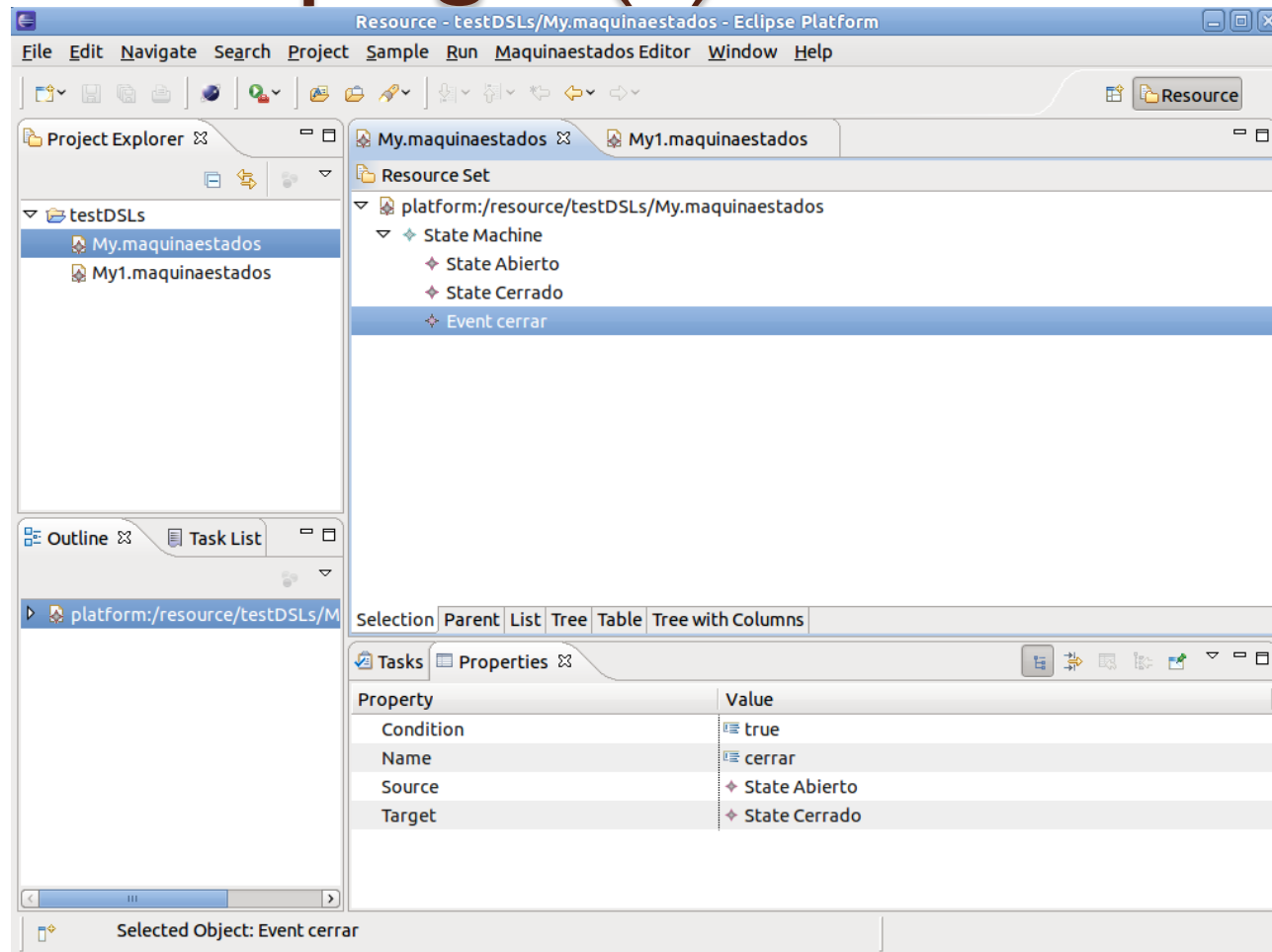


# Test del plugin



[Project Editor] Run as → Eclipse Application  
Al ejecutar el proyecto Editor, se abre una nueva instancia de Eclipse incluyendo el plugin tree-editor de modelos.

# Test del plugin (II)



Nuestro nuevo editor de modelos se integra perfectamente en el entorno Eclipse.

# Personalizar mensajes del plugin

```
pluginName = MaquinaEstados Editor
providerName = www.example.org
_UI_MaquinaestadosEditor_menu = &Maquinaestados Editor
_UI_CreateChild_menu_item = &New Child
_UI_CreateSibling_menu_item = N&ew Sibling
_UI_ShowPropertiesView_menu_item = Show &Properties View
_UI_OpenEditorError_label = Open Editor
_UI_Wizard_category = Example EMF Model Creation Wizards
_UI_CreateModelError_message = Problems encountered in file "{0}"
_UI_MaquinaestadosModelWizard_label = Maquinaestados Model
_UI_MaquinaestadosModelWizard_description = Create a new model
_UI_MaquinaestadosEditor_label = Maquinaestados Model Editor
```

[plugin.properties]

# Exportar plugin

**General Information**  
This section describes general information about this plug-in.

ID:

Version:

Name:

Provider:

Platform Filter:

Activator:

Activate this plug-in when one of its classes is loaded

This plug-in is a singleton

**Execution Environments**  
Specify the minimum execution environments required to run this plug-in.

JavaSE-1.6

[Configure JRE associations...](#)

[Update the classpath settings](#)

**Plug-in Content**  
The content of the plug-in is made up of two sections:

- [Dependencies](#): lists all the plug-ins required on this plug-in's classpath to compile and run.
- [Runtime](#): lists the libraries that make up this plug-in's runtime.

**Extension / Extension Point Content**  
This plug-in may define extensions and extension points:

- [Extensions](#): declares contributions this plug-in makes to the platform.
- [Extension Points](#): declares new function points this plug-in adds to the platform.

**Testing**  
Test this plug-in by launching a separate Eclipse application:

- [Launch a RAP Application](#)
- [Launch an Eclipse application](#)
- [Launch a RAP Application in Debug mode](#)
- [Launch an Eclipse application in Debug mode](#)

**Exporting**  
To package and export the plug-in:

- Organize the plug-in using the [Organize Manifests Wizard](#)
- Externalize the strings within the plug-in using the [Externalize Strings Wizard](#)
- Specify what needs to be packaged in the deployable plug-in on the [Build Configuration](#) page
- Export the plug-in in a format suitable for deployment using the [Export Wizard](#)

Overview | Dependencies | Runtime | Extensions | Extension Points | Build | MANIFEST.MF | plugin.xml | build.properties

[plugin.xml]

# Exportar plugin (II)

The screenshot shows the Eclipse IDE's 'Export' dialog box. The 'Deployable plug-ins and fragments' section is active, showing a list of available plug-ins and fragments. Three items are selected: 'maquinaEstadosDSL (1.0.0.qualifier)', 'maquinaEstadosDSL.edit (1.0.0.qualifier)', and 'maquinaEstadosDSL.editor (1.0.0.qualifier)'. The 'Destination' tab is selected, and the 'Directory' option is chosen with the path '/home/ivan/Packages'. A file explorer window titled 'plugins' is overlaid on the dialog, showing the contents of the directory '/home/ivan/Programas/eclipse/dropins/plugins'. The file explorer displays three files:

Nombre	Tamaño	Tipo
maquinaEstadosDSL.edit_1.0.0.201111091136.jar	12,8 KiB	archivador Java
maquinaEstadosDSL.editor_1.0.0.201111091136.jar	60,9 KiB	archivador Java
maquinaEstadosDSL_1.0.0.201111091136.jar	24,1 KiB	archivador Java

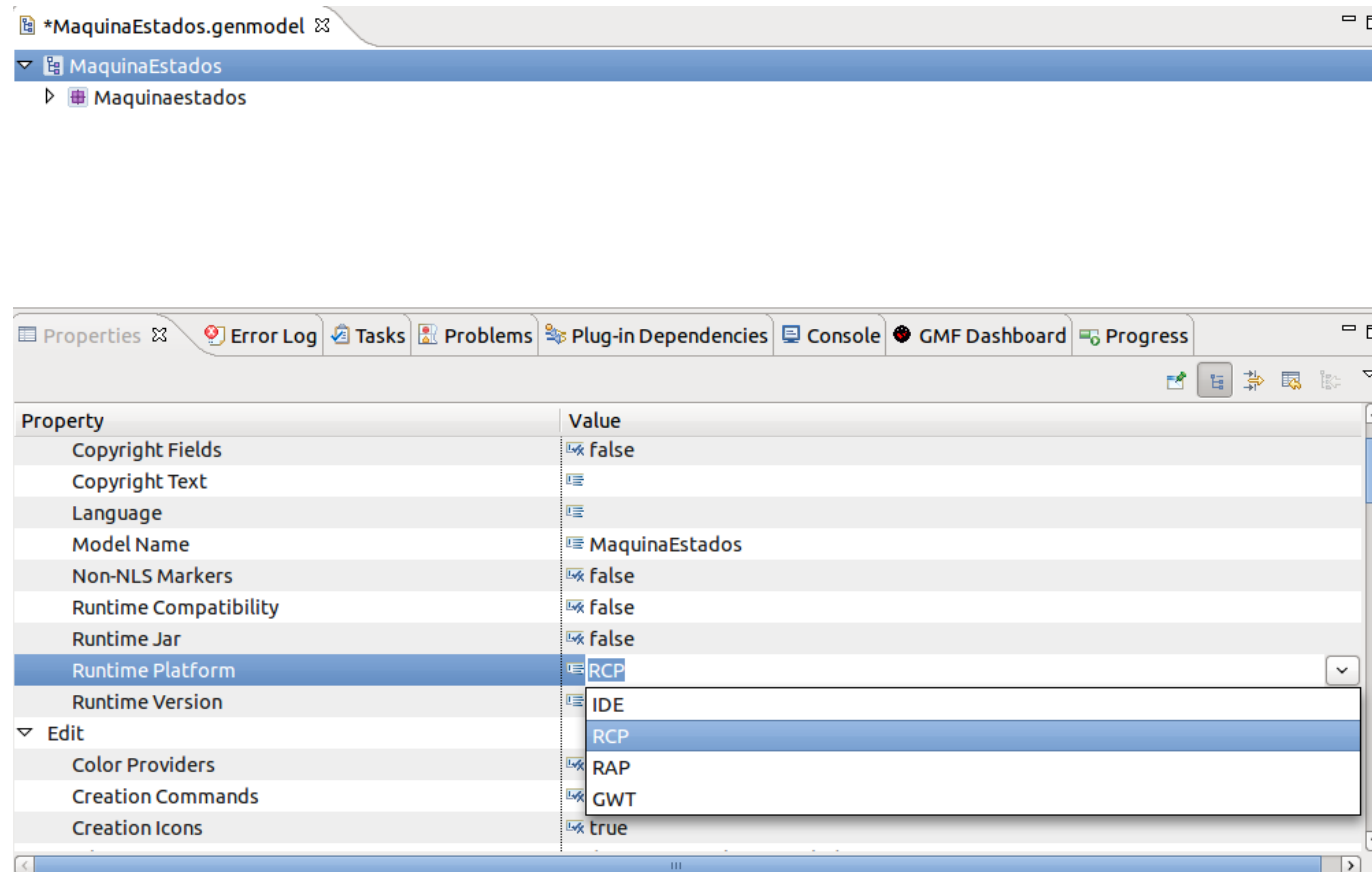
The 'Export' dialog has 'Cancel' and 'Finish' buttons at the bottom.

CONSTRUCCIÓN DE EDITORES DE MODELOS CON  
EMF



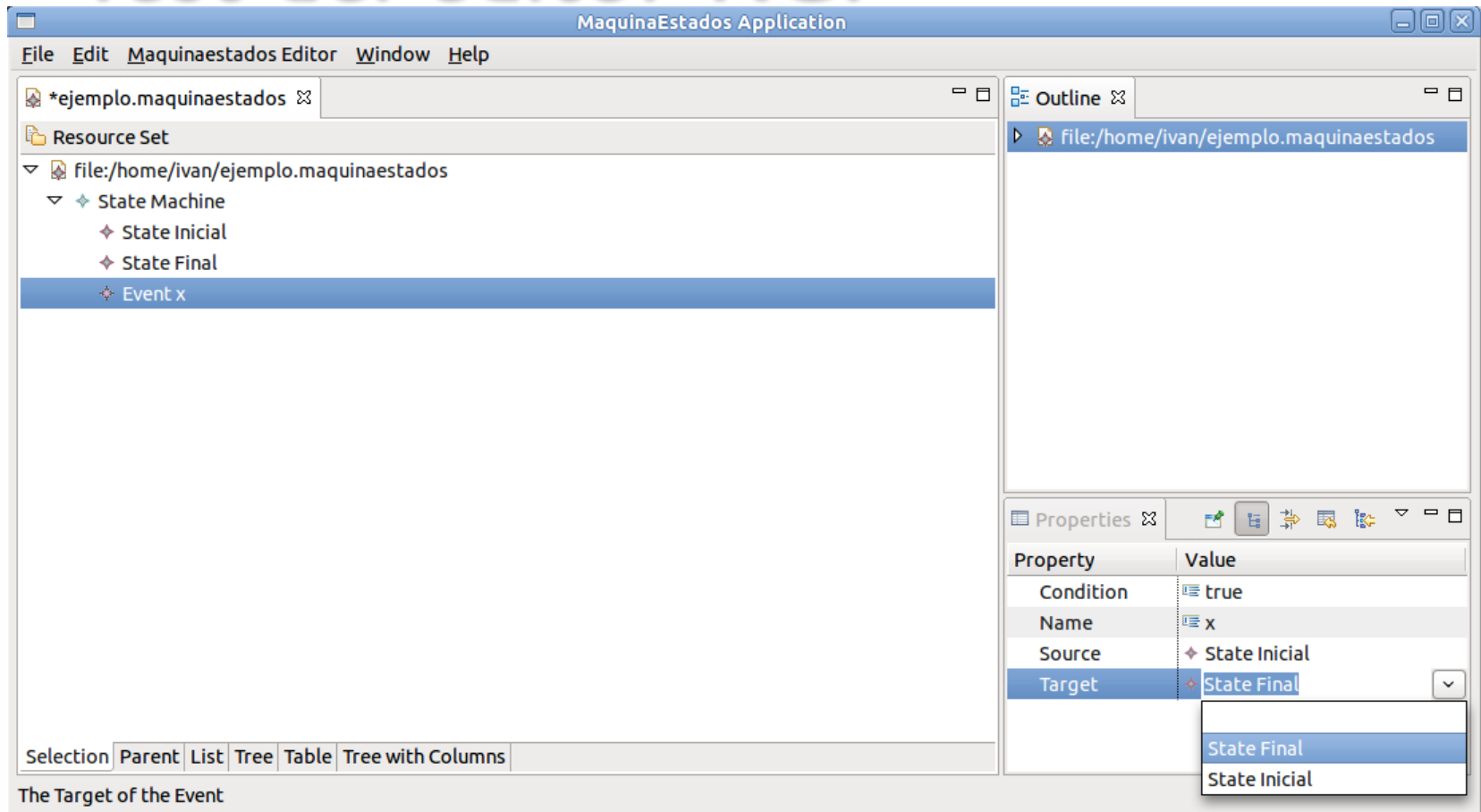
# **DESPLIEGUE DE EDITORES COMO PRODUCTOS**

# Generación del editor RCP



Eliminamos los proyectos generados automáticamente (*edit*, *editor* y *tests*), cambiamos “Runtime Platform” del *genmodel* a “RCP” y volvemos a generar el código fuente.

# Test del editor RCP



[Project *Editor*] Run as → Eclipse Application  
Nuestro nuevo editor de máquinas de estado aparece como una nueva aplicación de escritorio independiente del IDE de Eclipse.

PL2 - Construcción de editores de modelos

07/11/13 con EMF

24

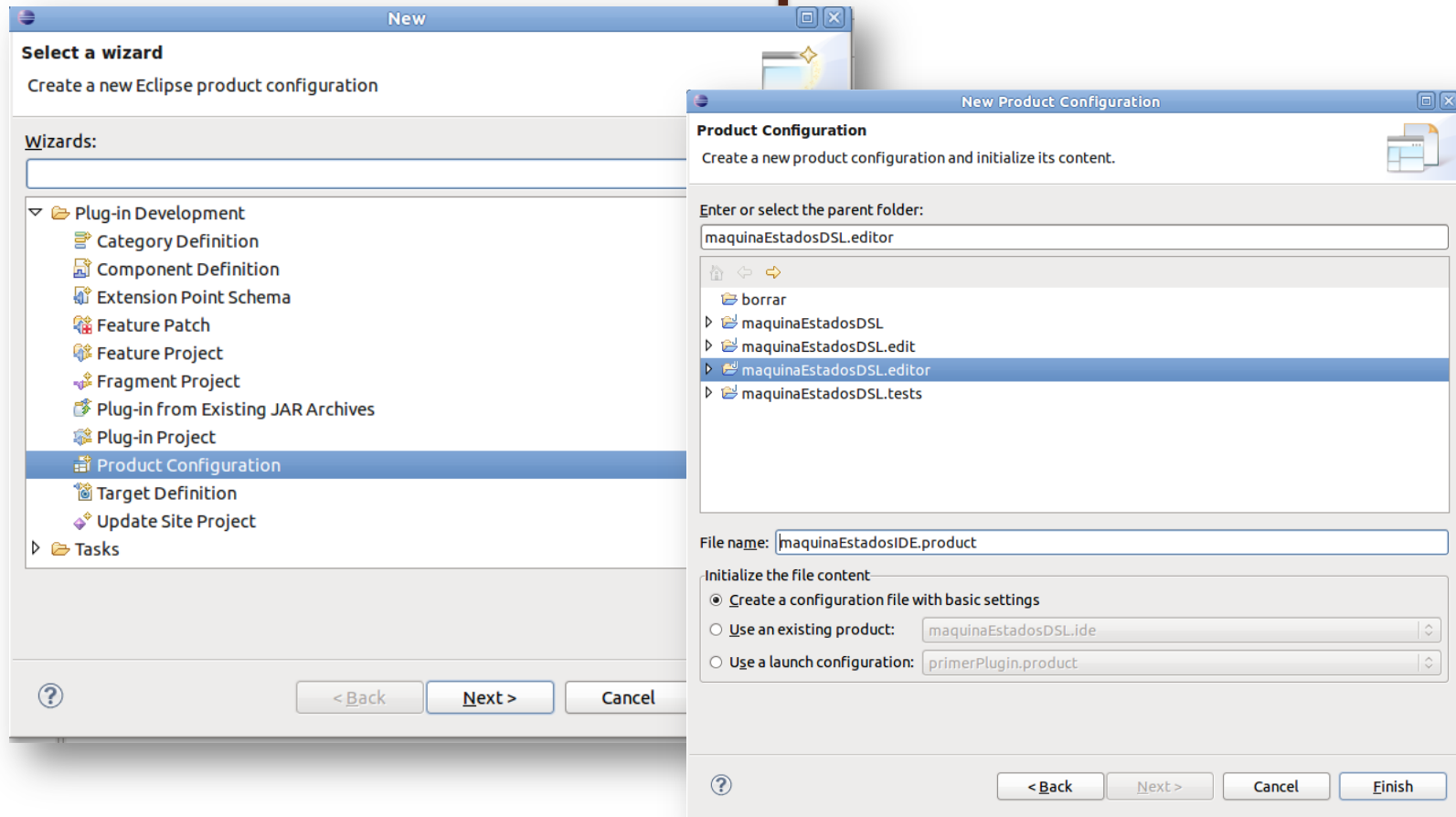


# Personalizar mensajes del RCP

```
_UI_Application_title = MaquinaEstados Application
_UI_Menu_File_label = &File
_UI_Menu_New_label = &New
_UI_Menu_Open_label = &Open...
_UI_Menu_Open_description = Opens a model object file
_UI_Menu_OpenURI_label = &Open URI...
_UI_Menu_Edit_label = &Edit
_UI_Menu_Window_label = &Window
_UI_Menu_Help_label = &Help
_UI_Menu_About_label = MaquinaEstados &About...
_UI_About_title = MaquinaEstados Application
_UI_About_text = MaquinaEstados Application about box goes here.
```

```
[plugin.properties]
```

# Creación de un producto



File → New → Product Configuration

Los productos definen todos los recursos que formaran parte del despliegue de nuestra aplicación RCP

# Configuración del producto

The screenshot shows the 'Overview' tab of the Eclipse IDE's Product Configuration dialog. The 'General Information' section contains fields for ID, Version (1.0.0), and Name (Maquina Estados Editor [PL2]), with a checked checkbox for 'The product includes native launcher artifacts'. The 'Product Definition' section shows 'Product' as 'maquinaEstadosDSL.editor.maquinaEstadosIDE' and 'Application' as 'maquinaEstadosDSL.editor.MaquinaEstadosEditorAdvisorApplication', with radio buttons for 'plug-ins' (selected) and 'features'. The 'Testing' section lists four actions: 'Synchronize', 'Launch a RAP Application', 'Launch an Eclipse application', 'Launch a RAP Application in Debug mode', and 'Launch an Eclipse application in Debug mode'. The 'Exporting' section explains using the 'Eclipse Product export wizard' and lists two steps for multi-platform export. The bottom navigation bar includes 'Overview', 'Dependencies', 'Configuration', 'Launching', 'Splash', 'Branding', and 'Licensing'.

**Overview**

**General Information**  
This section describes general information about the product.

ID:

Version:

Name:

The product includes native launcher artifacts

**Product Definition**  
This section describes the launching product extension identifier and application.

Product:

Application:

The [product configuration](#) is based on:  plug-ins  features

**Testing**

- [Synchronize](#) this configuration with the product's defining plug-in.
- Test the product by launching a runtime instance of it:
  - [Launch a RAP Application](#)
  - [Launch an Eclipse application](#)
  - [Launch a RAP Application in Debug mode](#)
  - [Launch an Eclipse application in Debug mode](#)

**Exporting**

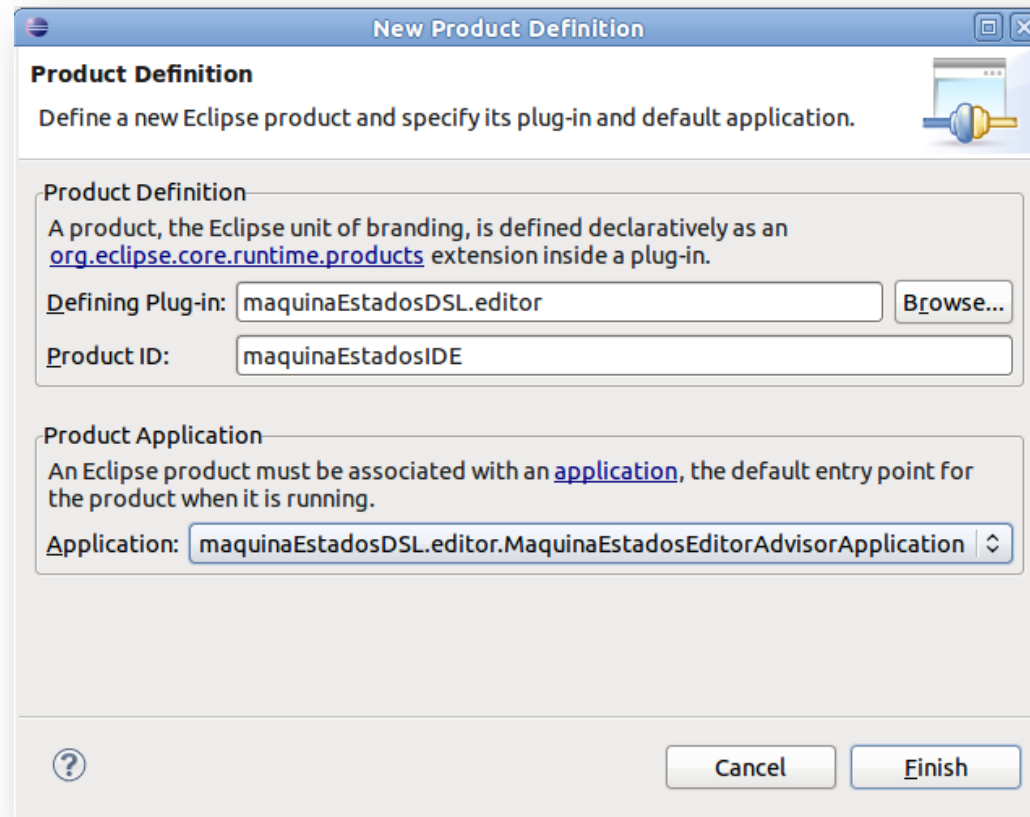
Use the [Eclipse Product export wizard](#) to package and export the product defined in this configuration.

To export the product to multiple platforms:

- Install the RCP delta pack in the target platform.
- List all the required fragments on the [Dependencies](#) page.

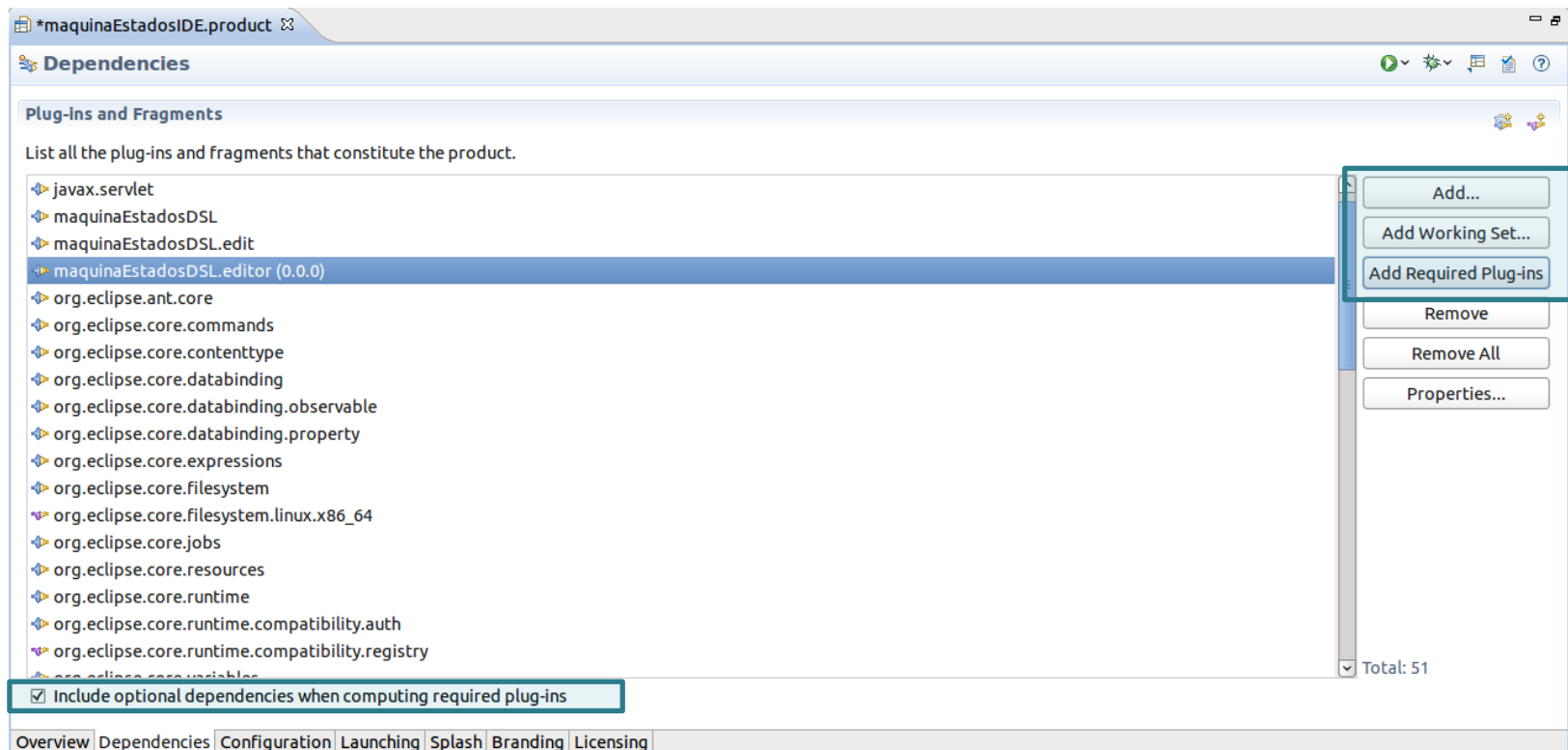
Overview Dependencies Configuration Launching Splash Branding Licensing

# Definición del producto



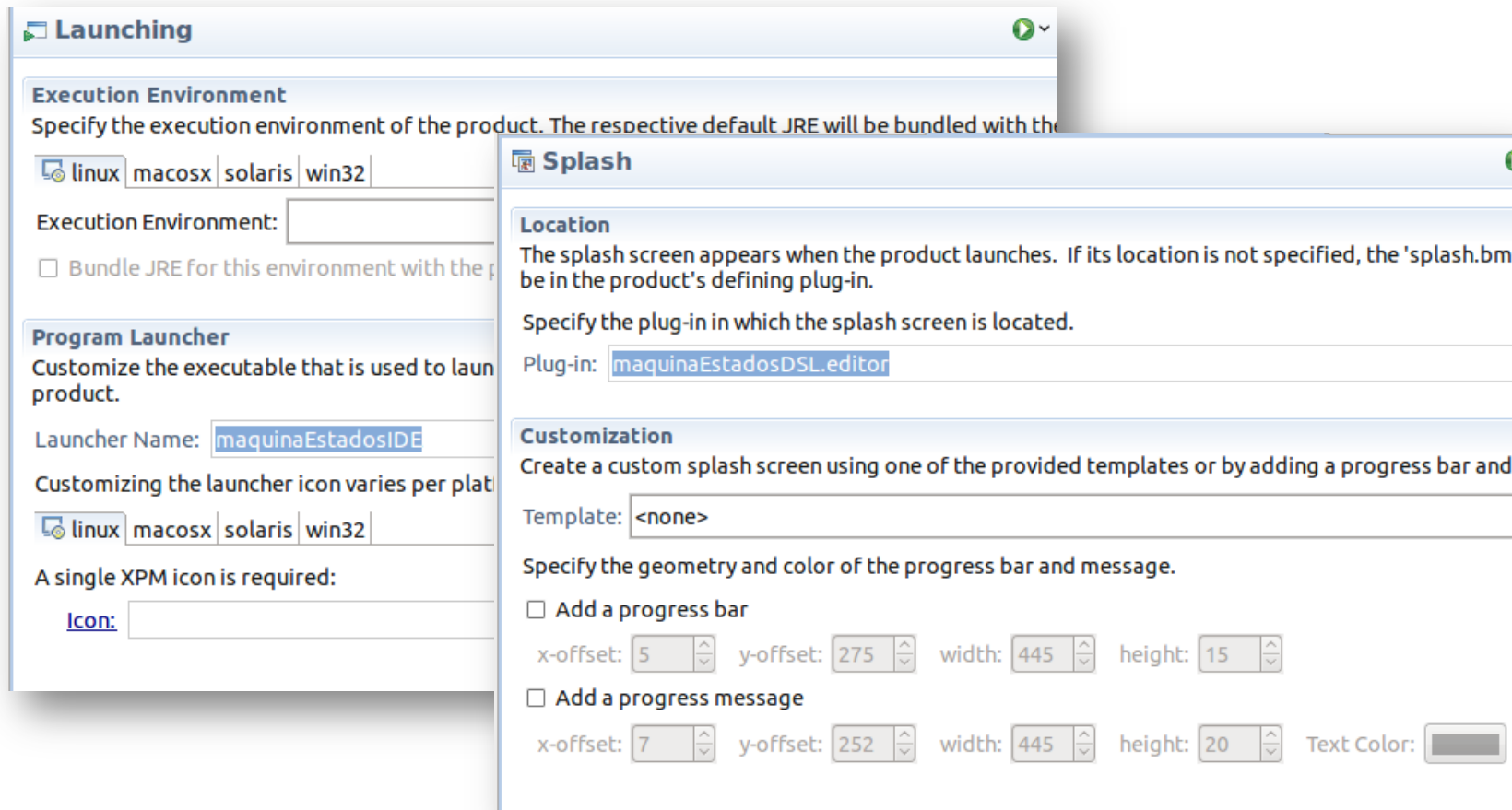
Para definir completamente nuestro producto debemos seleccionar nuestro plugin (proyecto Editor) y un identificador (nombre del fichero .product)

# Dependencias del producto



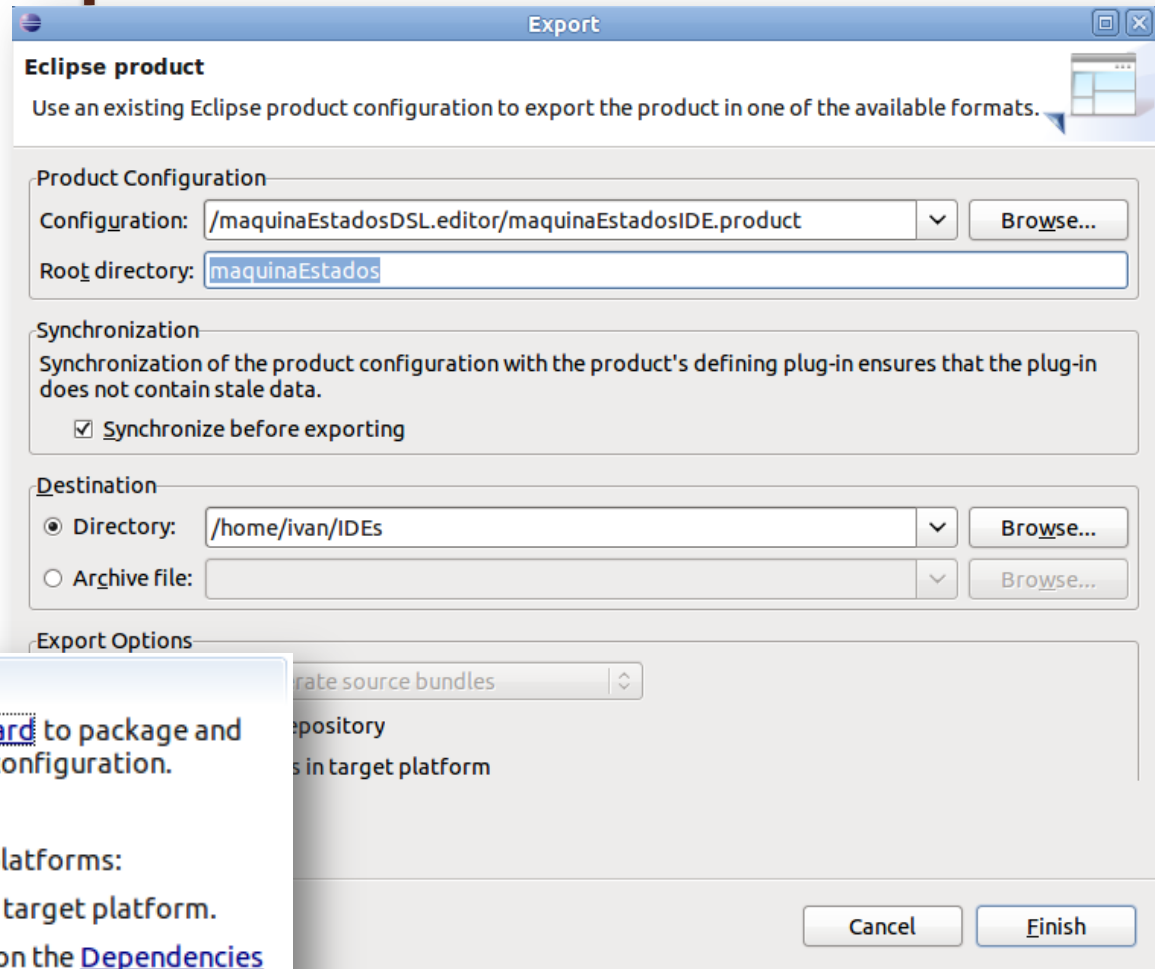
Debemos indicar que nuestro nuevo producto dependerá principalmente del plugin *Editor*. Luego, tenemos que añadir (de forma automática) las dependencias necesarias.

# Aspectos adicionales del producto



Podemos especificar los distintos entornos de ejecución de nuestro producto, pantalla de inicio, iconos, licencia, etc.

# Exportar producto



## Exporting

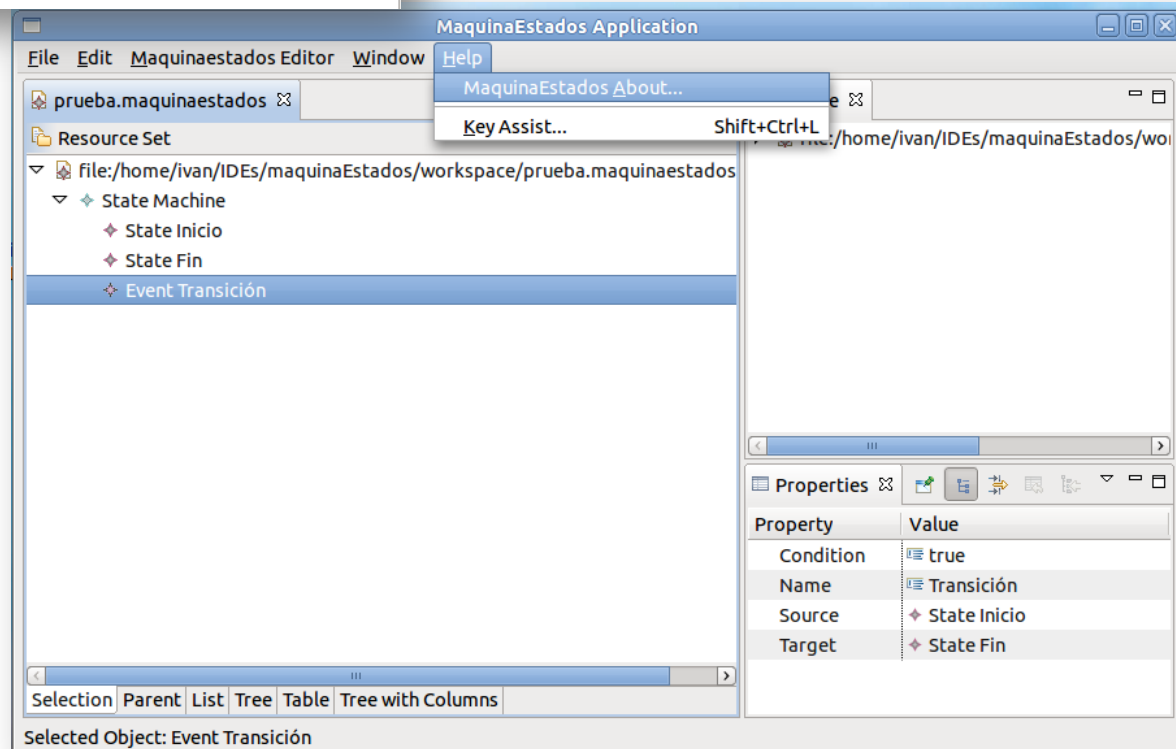
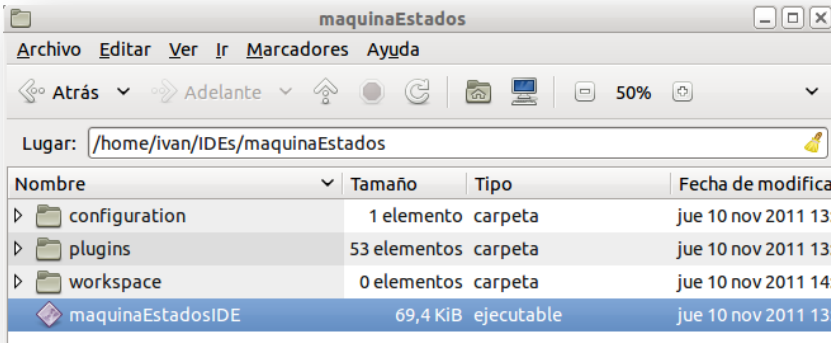
Use the [Eclipse Product export wizard](#) to package and export the product defined in this configuration.

To export the product to multiple platforms:

1. Install the RCP delta pack in the target platform.
2. List all the required fragments on the [Dependencies](#) page.



# Exportar producto (III)







CONSTRUCCIÓN DE EDITORES DE MODELOS CON  
EMF

 **RESUMEN**

# ¿Qué hemos aprendido hoy?

- Desarrollar editores sencillos de modelos basados en árbol, utilizando Eclipse EMF.
- Desplegar los editores como plugins del propio IDE.
- Desplegar los editores como productos independientes.



Procesadores de Lenguajes 2

# Construcción de editores de modelos con EMF

Curso 2013-2014

**Iván Ruiz Rube**

ivan.ruiz@uca.es