Capítulo 1

Ingeniería del Dominio

En este capítulo se describe la fase de *Ingeniería del Dominio* (en inglés, *Domain Engineering*) de nuestra línea de productos software. Dentro de dicha fase .....

1.1. Transformaciones de Modelo UML a C#

1.2. Generadores de Código C#

Tras explicar a grandes rasgos las transformaciones de modelo a código en la sección 1.2, se procede a comentar la implementación de los generadores de código correspondientes. La figura 1.1 muestra la jerarquía de los generadores de código implementados. El punto de partida es el generador de código llamado ProjectCreation, a partir de el se invoca a la plantilla ClassFilesCreation que determina si un elemento es:

- Una clase o interfaz, en tal caso llamaría a su vez al generador de código ClassCreation.
- Una clase con herencia múltiple, en este caso se requiere un tratamiento especial que veremos más adelante en detalle, en tal caso llamaría a su vez a los generadores de código ClassCreationMultipleHierarchyCase, ChildClassHierarchyCase e InterfaceCreation.

Una vez se ha determinado el tipo de elemento de los descritos en la enumeración anterior, se procede a generar el fichero fuente correspondiente a dicho elemento. Para ello, y en un orden secuencial estricto, se hacen llamadas a las plantillas descritas en forma de árbol en la figura 1.1. Por ejemplo, para la generación de una clase el proceso sería tal como se muestra a continuación:

1. ProjectCreation

2. ClassFilesCreation, se determina que es una clase.
3. ClassCreation, se indica la ruta para guardar el fichero que se va a generar.
4. ModelCreation, genera el namespace del proyecto.
5. ClassDeclaration, genera la declaración de la clase parcial.
6. PropertiesGeneration, genera las propiedades de la clase.
7. MethodsCreation, genera los métodos de la clase.
8. UtilityMethodsCreation, genera los métodos de utilidad de la clase.
9. Una vez generados todos los elementos de la clase, se genera el fichero fuente correspondiente.

Después de haber generado todos los paquetes, clases e interfaces del proyecto se deben generar los ficheros necesarios para que dicho proyecto pueda ser abierto con el programa Visual Studio sin problemas. Para ello es necesario generar una serie de ficheros específicos de dicha plataforma, dichos ficheros se describen a continuación:

- SlnFileCreation, genera el fichero .sln que contiene información del entorno del sistema.
- CsprojectFilesCreation, genera el fichero .csproj correspondiente para incluir los directorios creados en el proyecto Visual Studio.
- AssemblyInfoFileCreation, genera el fichero .cs que contiene la información general sobre las directivas de ensamblado.

Aunque no aparezca en el diagrama, todos los generadores de código utilizan además el fichero Operations, que es un fichero EOL que dispone de operaciones básicas que se utilizan recurrentemente en el resto de generadores, por nombrar algunos ejemplos, dicho fichero contiene funciones del estilo:

- *abstract*, retorna el texto abstract cuando el elemento sobre el que está siendo utilizado es abstracto.
- *typeCollection*, retorna el tipo de colección correspondiente al elemento sobre el que se está actuando.
- *hierarchy*, retorna las clases de las que hereda el elemento en cuestión.

Adicionalmente a los generadores de código EGL y EOL, en los sucesivos párrafos se habla en detalle de los Java Tools adicionales que se han elaborado para hacer más gráfico y sencillo al usuario el proceso de ejecución de los generadores de código.

Java Tool

Un Java Tool es una porción de código java que tras ser empaquetada como plug-in se puede incorporar en el directorio de instalación de Epsilon y utilizar las funciones implementadas en las plantillas de generación de código creadas. Se han desarrollado cuatro Java Tools en total:

- pluginCreateDirectories, genera los directorios en el sistema.
- pluginCreateMessageWindow, genera la ventana con los errores y/o información oportunos (figura 1.3).
- pluginCreateSelectDirectoryWindow, genera un entorno de exploración por el equipo para elegir el destino de la generación del proyecto (figura 1.2).
- pluginCreateWindowAndShowMessages, genera una ventana al final de la ejecución con información para el usuario (figura 1.4).
- pluginWriteInFile, genera el fichero que contendrá la información para mostrar al usuario al final la ejecución.

1.3. Sumario

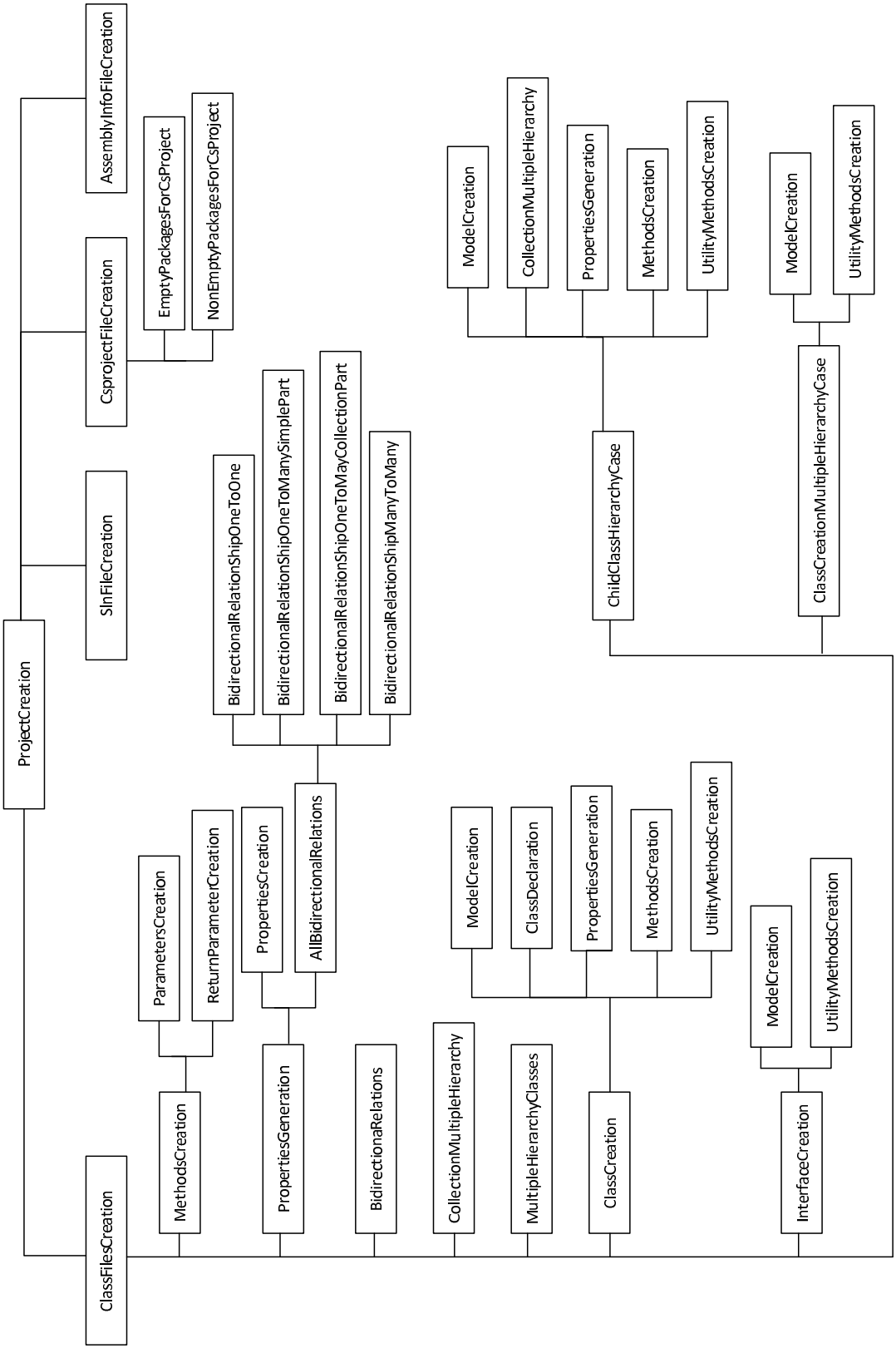

Figura 1.1: Jerarquía de los generadores de código implementados

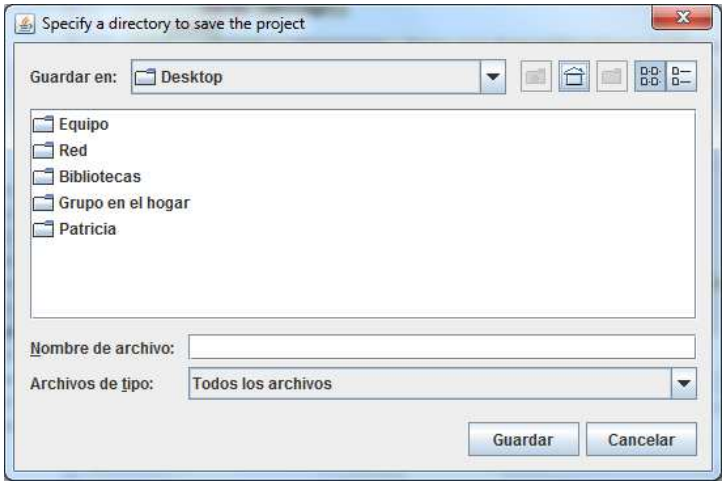

Figura 1.2: Selección de directorio donde emplazar el proyecto a generar

Figura 1.3: Ventana de error

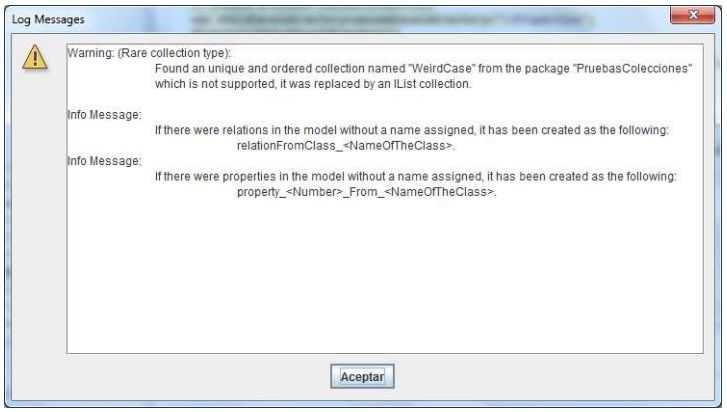

Figura 1.4: Ventana al final de la generación de código