MANUAL DE INSTALACIÓN Y USO

HERRAMIENTA DE GENERACIÓN AUTOMÁTICA DE PRUEBAS

Miguel Angel Gallardo Ruiz

Departamento de Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
Málaga, Agosto de 2022

Tabla de contenidos

1. Manual de Instalación

En esta sección se muestra un manual para que el usuario pueda replicar la parte de la generación de pruebas de este proyecto. En este apartado nos centraremos en concreto en la instalación.

Lo primero que debe hacer el usuario es descargar Eclipse e instalarlo, esta parte es trivial, lo único que debe hacer el usuario es dirigirse al navegador y buscar el ejecutable que más se ajuste a su computadora, en este caso para nosotros es la ultima versión como se observa en la Figura 1.

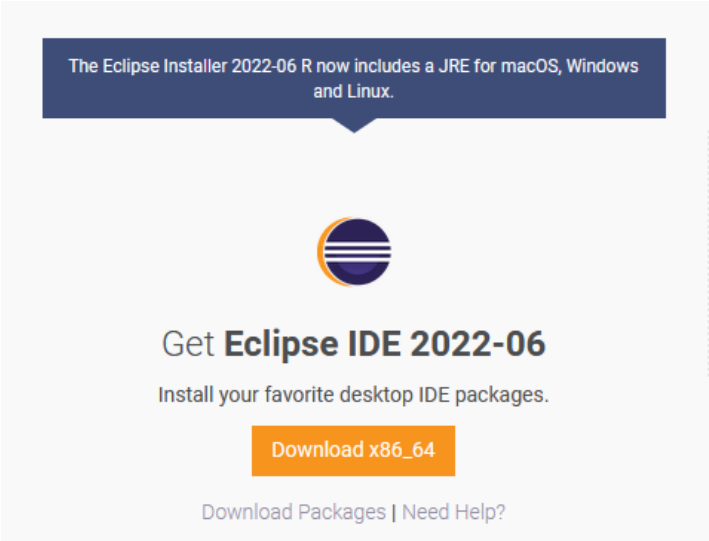

Figura 1: Versión de Eclipse

Una vez se tiene instalado el usuario debe instalar Xtext, para ello debe dirigirse al siguiente enlace:

`https://www.eclipse.org/Xtext/download.html`

Una vez se encuentra en esta pagina se debe dirigir como se observa en la Figura 2 al apartado 'Releases' y con el click derecho darle a la opción de copiar enlace.

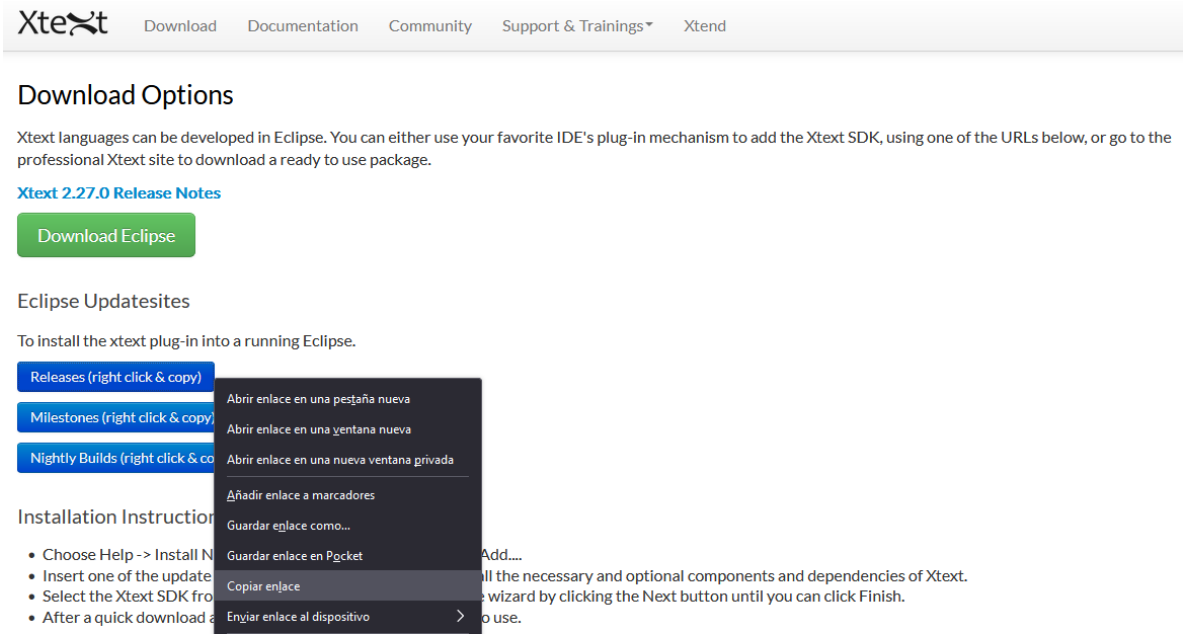

Figura 2: Release de Xtext

Con esta dirección copiada debe de irse al entorno de Eclipse y en el apartado ‘Help’ abrir ‘Install New Software’, como se ve en la 3

Aquí se abre la pestaña de la Figura 4 la cual permite la instalación de paquetes, aquí el usuario debe de seleccionar el botón ‘Add...’ el cual abre la pestaña de la Figura 5. En esta se debe escribir el nombre que se le quiera dar al paquete a instalar, en este caso Xtext, y pegar el enlace que previamente se ha copiado de Xtext. Una vez hecho esto el usuario le da al botón ‘Add’ y debe esperar a que carguen las opciones.

Una vez cargadas el usuario debe seleccionarlas todas como se observa en la Figura 6 y darle al botón ‘Next’. Cuando hayan cargado todas las opciones el programa Eclipse se debe de reiniciar y ya se puede empezar a usar Xtext.

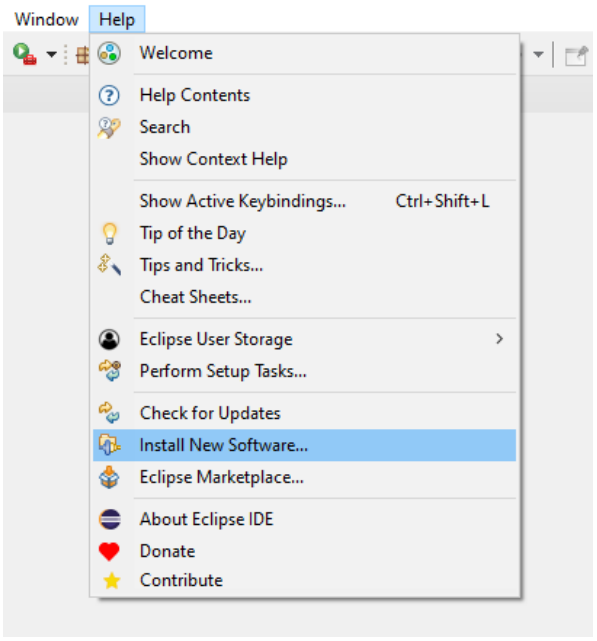

Figura 3: Apertura de la herramienta de instalación

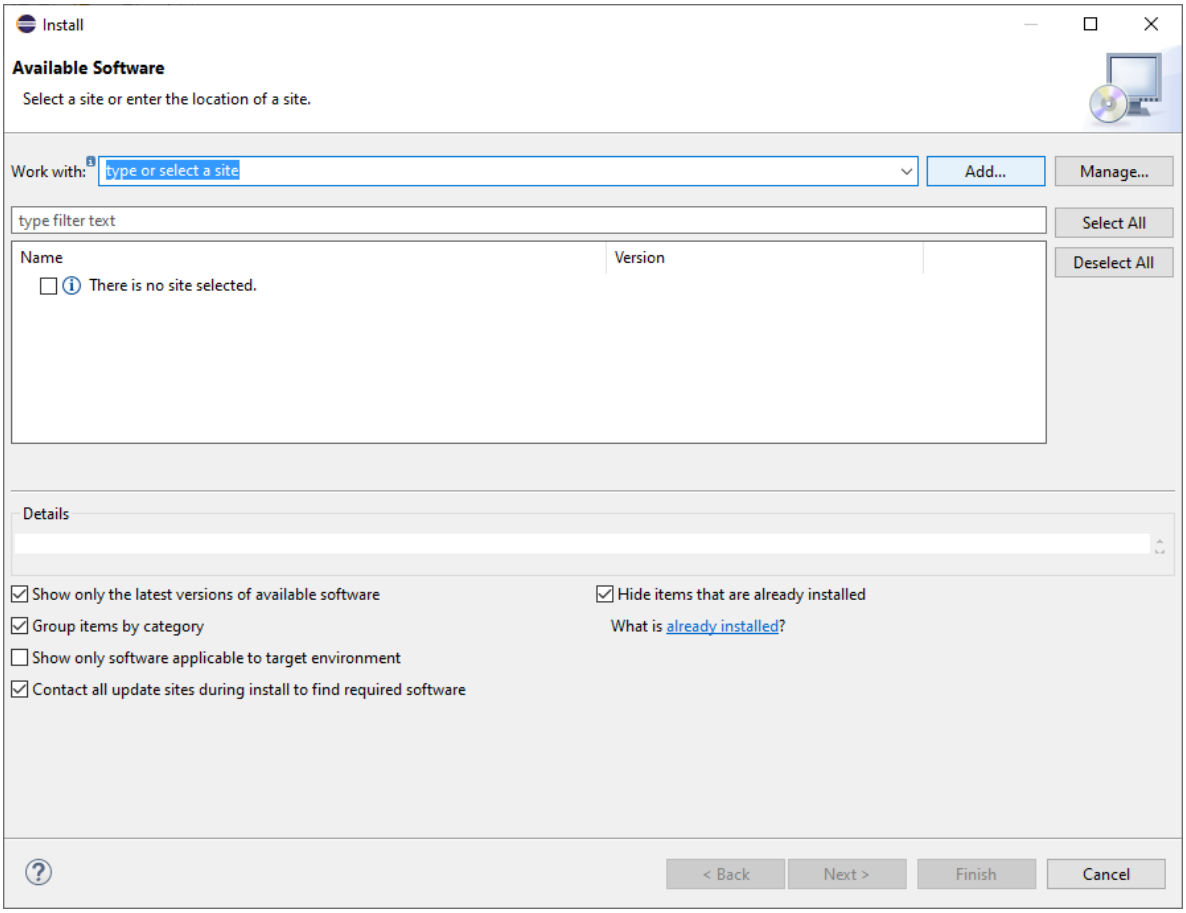

Figura 4: Herramienta de instalación

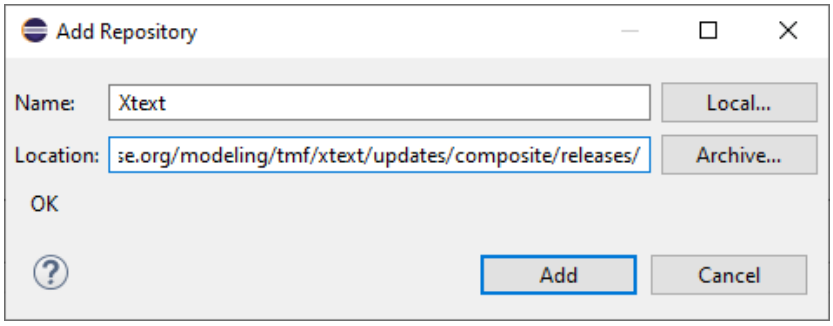

Figura 5: Importación del repositorio de Xtext

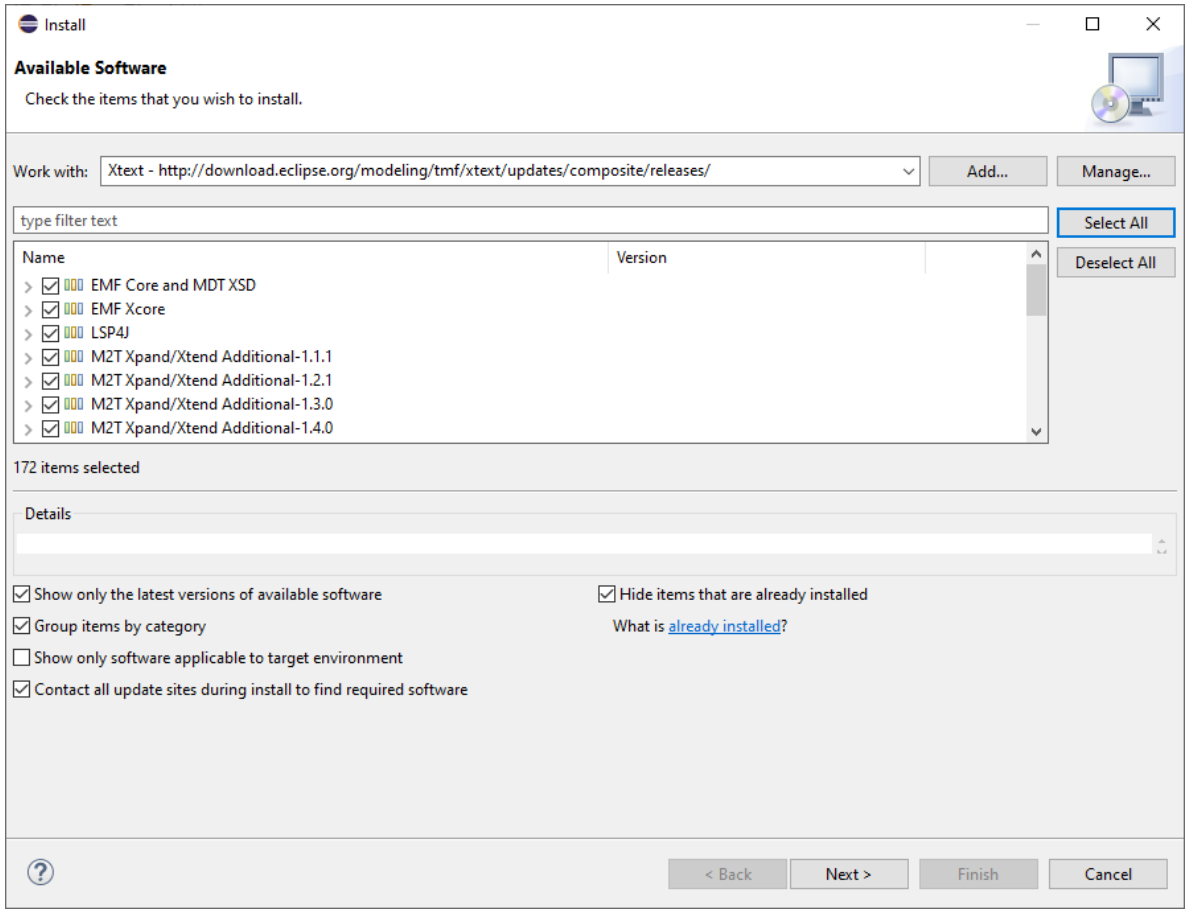

Figura 6: Instalación del repositorio de Xtext

2. Manual de Usuario

En esta sección se muestra un manual para que el usuario pueda replicar la parte de la generación de pruebas de este proyecto.

Lo primero que debe hacer el usuario es abrir Eclipse, una vez abierto deberá crear un nuevo proyecto, el cual debe elegir un proyecto Xtext, tal y como se observa en la Figura 7.

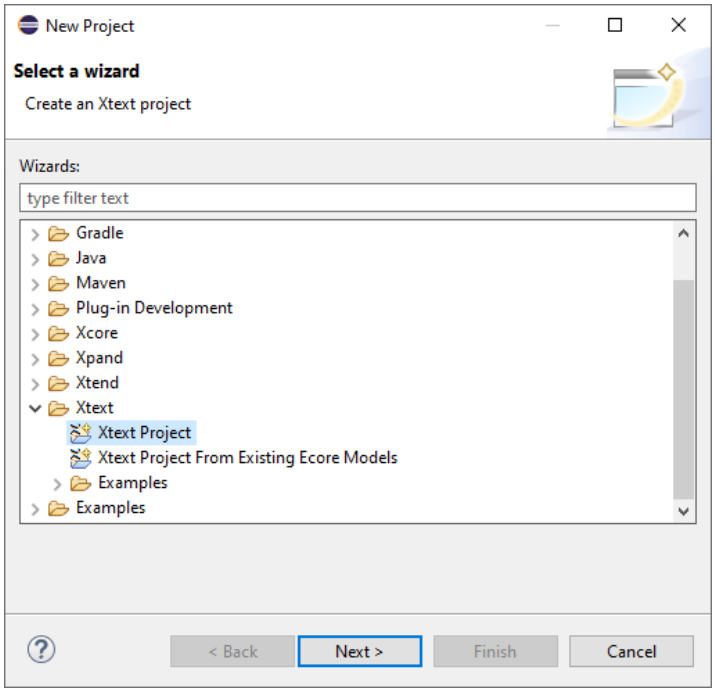

Figura 7: Selección del nuevo proyecto

Una vez se le da a continuar (Next), se abre una nueva pestaña como la que se ve en la Figura 8, en esta se define el nombre del proyecto, así como el nombre del lenguaje y la extensión que tendrá este.

Estos dos primeros pasos se tienen que realizar dos veces, ya que se requiere de un proyecto general y otro especifico para las operaciones a realizar.

Ahora que se tienen los dos proyectos creados, el siguiente paso es añadir el proyecto de las operaciones como dependencias del general. La manera de realizar esto se observa en la Figura 9, donde se abre el fichero 'MANIFEST.MF' y, en el apartado de dependencias darle al botón de añadir (Add). Se abre entonces una pantalla emergente donde se debe buscar el nombre del proyecto a añadir como dependencia y a continuación añadirlo (Add).

El usuario ahora puede escribir el código tanto del proyecto general como del proyecto de las operaciones en Xtext.

Lo único que hay que tener en cuenta es que el proyecto general debe añadir al de operaciones es su gramática y, añadir en el fichero con extensión '.mwe2', como se observa en la Figura 10, la localización del modelo generado para el proyecto de operaciones.

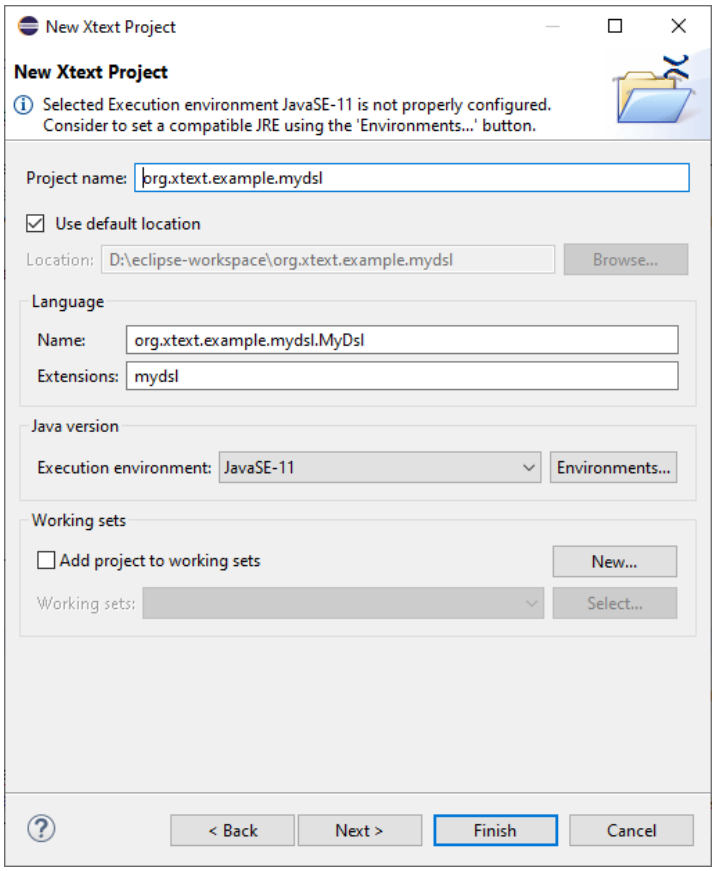

Figura 8: Definiendo las características del proyecto

Una vez se tiene esto, se puede proceder a compilar los proyectos. Para esto el usuario debe localizar el fichero '.xtext' principal de cada proyecto y compilarlos, como se ve en la Figura 11. El orden será; primero el de las operaciones y luego el general.

Ahora el usuario puede definir el código del fichero '.xtend' con el texto que quiera generar como salida.

Teniendo este terminado, se puede proceder a lanzar la aplicación como se observa en la Figura 12, esta nos abrirá un nuevo entorno Eclipse.

En este nuevo entorno que se ha abierto, se crea un nuevo proyecto vacío. Dentro de este se debe crear una carpeta la cual se llamará 'src', y dentro de esta carpeta un nuevo fichero al cual se le debe poner la extensión del proyecto general con el cual estamos lanzando esta aplicación y que se ha definido al principio, como se ve en la Figura 13.

El usuario puede escribir en este nuevo fichero, el cual usará las especificaciones definidas en el proyecto. Una vez el usuario escriba el código y se guarde, se genera automáticamente una nueva carpeta 'src-gen' en la cual se crea a su vez un fichero con extensión de texto con todo el lenguaje que se ha definido en el fichero '.xtend' haciendo uso de lo definido por el usuario en el fichero con las especificaciones.

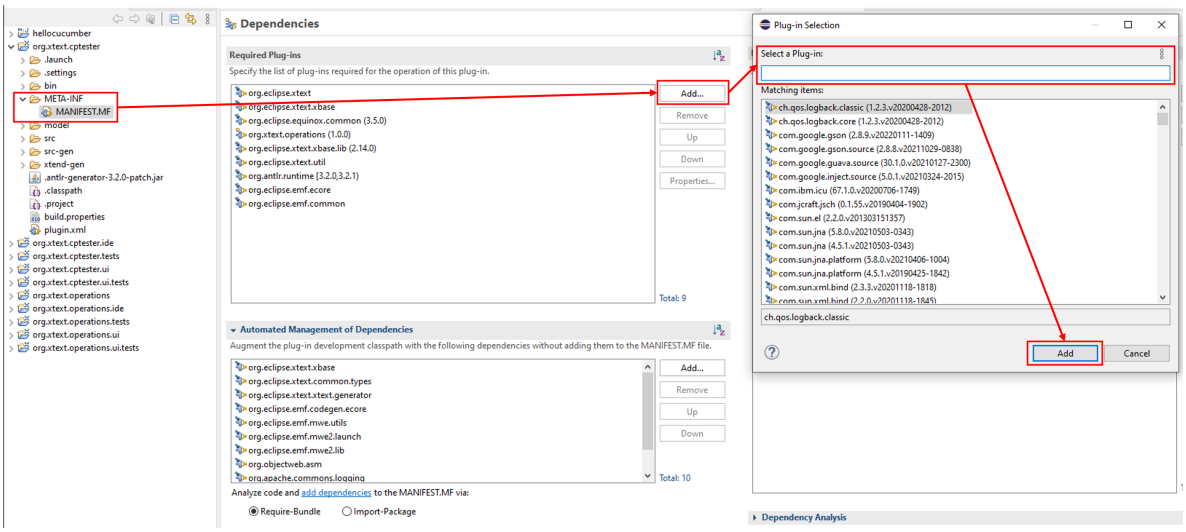

Figura 9: Adicción de las dependencias

```
bean = org.eclipse.emf.mwe.utils.StandaloneSetup {
    scanClassPath = true
    uriMap = {
        from = "platform:/plugin/org.eclipse.emf.ecore/model/Ecore.ecore"
        to = "platform:/resource/org.eclipse.emf.ecore/model/Ecore.ecore"
    }
    uriMap = {
        from = "platform:/plugin/org.eclipse.emf.ecore/model/Ecore.genmodel"
        to = "platform:/resource/org.eclipse.emf.ecore/model/Ecore.genmodel"
    }
    registerGenModelFile = "platform:/resource/org.xtext.operations/model/generated/Operations.genmodel"
```

Figura 10: Código del fichero '.mwe2' del proyecto general con las localizaciones del modelo del proyecto operaciones

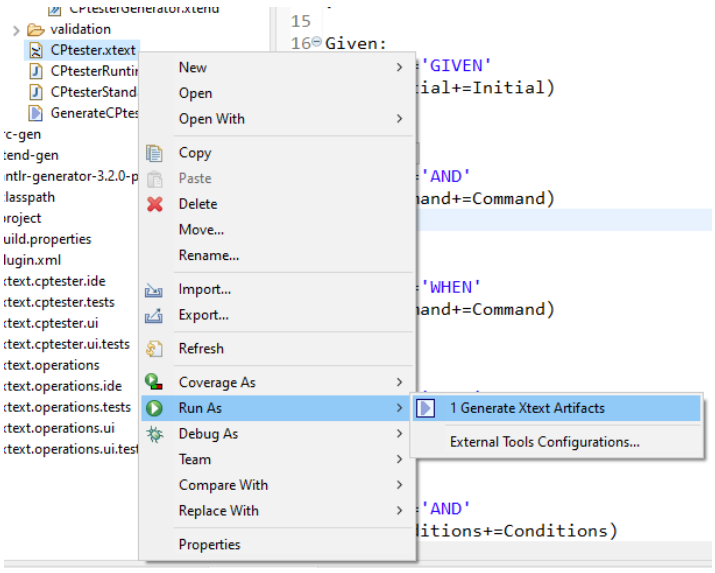

Figura 11: Compilación del codigo '.xtext'

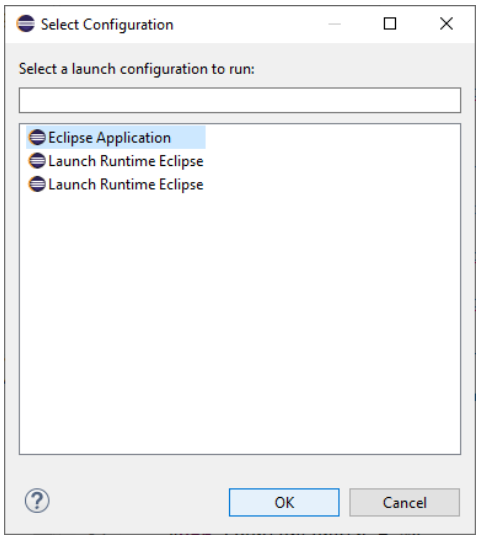

Figura 12: Ejecución del proyecto

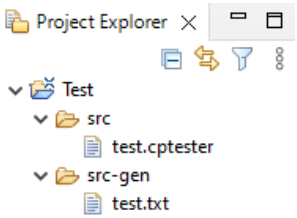

Figura 13: Generación de carpetas y ficheros en el proyecto

3. Ejemplo

En este ejemplo el usuario crea una serie de operaciones para realizar pruebas con el brazo.

La operación a realizar por el usuario en este caso consiste en la inicialización correcta del brazo con un tiempo de 1000ms, el apagado y posterior encendido del buzzer, un movimiento posterior de todos los servos a una posición de 45° con un tiempo de 2000ms, la rotación del servo numero 3 a una posición de 30° con un tiempo de 1000ms, la rotación del servo 1 a una posición de 90° con un tiempo de 2000ms y la comprobación de que se encuentra en esta posición con un margen de error de 2° y que la operación no ha tardado mas de 2000ms.

Todo este código generado por el usuario se puede ver en la figura 14

```
Scenario "Testings"
{
    GIVEN posInicial(1000)
    AND buzzerOff()
    AND buzzerOn(1)
    AND rotateAllServos(45, 45, 45, 45, 45, 45, 2000)
    AND rotateServo(3, 30, 1000)
    WHEN rotateServo(1, 90, 2000)
    THEN result(2000)
    AND NotLaterThan(2000) Query isAtSingle(1, 90, 2)
}
```

Figura 14: Código del usuario

A continuación, se va a ver el código pivote generado a través del código escrito por el usuario.

Como se puede observar en la Figura 15, en la parte izquierda se tiene el código en CPtester que el usuario especifica. A la derecha se puede ver el resultado del lenguaje pivote que se obtiene mediante Xtend.

Para este caso en concreto, se ve cómo el usuario define primero el nombre del escenarios a través de *"Scenario "Testings" "*, esto se traduce mediante la creación de *"StateMachine: Testings"*, este será el nombre del diagrama de estados que tendrá en Altova UModel una vez se importe.

A continuación, se tiene primero *"GIVEN posInicial(1000)"*, este se corresponde en el lenguaje pivote a *"Activity: Arm.posInicial(1000)"* como se ve en la Figura **??**. El usuario con eso lo que quiere es realizar la primera acción posible con el brazo, en este caso llevarlo a la posición inicial y en concreto, como viene definido, con un tiempo de 1000ms. También se observa como se generan, aparte del estado con la operación, distintas transiciones desde el estado actual hasta el siguiente si todo va bien, así como transiciones a estados de tipo "Warning" si hay algún error.

La siguiente instancia es *"AND buzzerOff()"*, la cual apaga el buzzer, la operación en el lenguaje pivote correspondiente a esto es *"Activity: Arm.Buzzer.buzzerOf()"*. Al igual que la operación anterior, se genera el estado correspondiente y todas las transiciones a los distintos estados.

Del mismo modo, tenemos la siguiente instancia *"AND buzzerOn(1)"*, que encenderá el led con un periodo de 1000ms. La operación que se genera es *"Activity: Arm.Buzzer.buzzerOn(1)"* dentro del estado correspondiente y con la consiguientes posibles transiciones generadas también de manera automática.

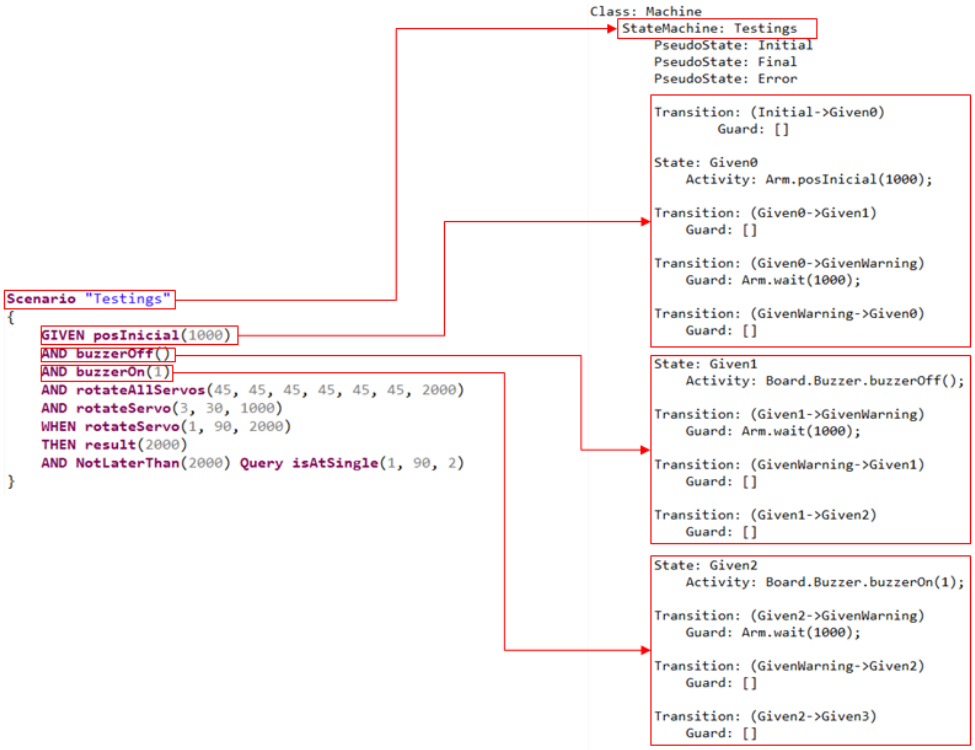

Figura 15: Primera parte del código generado

En la siguiente Figura 16, tenemos la segunda parte del código que se genera.

Siguiendo las operaciones comentadas anteriormente, ahora nos encontramos con *"AND rotateAllServos(45, 45, 45, 45, 45, 2000)"*, la cual se traduce en el lenguaje pivote como *"Activity: Arm.rotateAllServos(45, 45, 45, 45, 45, 2000)"* en el estado correspondiente y con las transiciones pertinentes a el.

Por ultimo en el apartado AND, nos encontramos con *"AND rotateServo(3, 30, 2000)"* que se corresponde con la operación con su estado y transiciones, excepto la ultima transición que se genera de manera automática al siguiente estado correspondiente con la operación principal de este ejemplo, el cual se corresponde con el WHEN. En este caso la transición cogerá la posición de la ultima operación dada para usarla como guarda a la hora de transitar al siguiente estado. También se observa la creación de estado WARNING que se ha estado haciendo referencia hasta ahora en las anteriores transiciones.

Ahora nos encontramos con *"WHEN rotateServo(1, 90, 2000)"* que se corresponde con la operación principal a realizar, esta se traduce en lenguaje pivote como *"Activity: Arm.BaseServo.ServoOperations 90, 2000)"*. A continuación, tenemos las operaciones *"THEN result(2000)"* y *"AND NotLaterThan(2000) Query isAtSingle(1, 90, 2)"*, las cuales correspondientes con la transición al estado de ERROR si no se realiza en el tiempo indicado así como la transición al estado FINAL si se cumple la condición indicada por el usuario con respecto a la posición del ángulo.

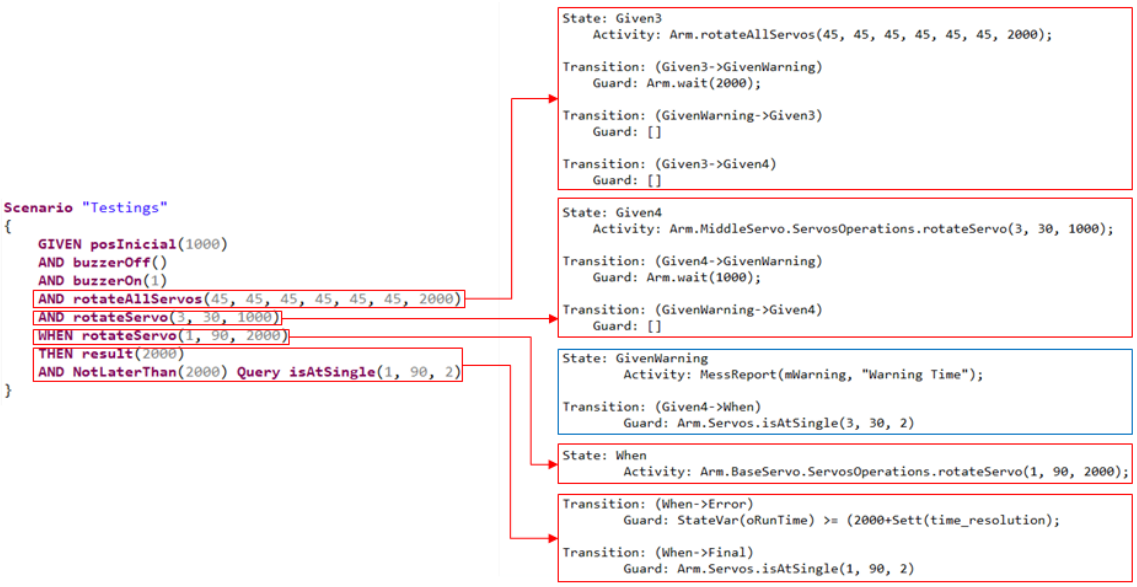

Figura 16: Segunda parte del código generado