**Informe TP2.2**

Para utilizar la librería, el usuario debe crear sus test implementando la clase abstracta “TestCases” hecha para este propósito. Anteriormente, el usuario implementaría ITestCase pero la hemos cambiado porque al agregar la posibilidad de poder crear TestSuite nos hemos decidido por una interfaz más genérica “ITest” que provee funcionalidad para ambas implementando el patrón “Composite” lo cual nos soluciona el problema de insertar un test dentro de otro.

A este respecto, el usuario no implementa directamente ITest porque preferimos filtrar algunos métodos mediante la clase abstracta TestCase que serían irrelevantes para la implementación de cases por el usuario.

En este escenario, el usuario puede crear cases implementando TestCases y a su vez puede agregarlas a un TestSuite. La TestSuite es suficientemente genérica como para poder recibir cases o suites indistintamente. Este comportamiento es posible ya que como se dijo anteriormente TestCase y TestSuite implementan la interfaz ITest.

Ahora bien, para poder correr los test., el usuario debe tener una instancia de TestRunner y agregarle el test (ya sea Suite o Case) o tests anidados mediante el método “AddTest”. Luego, el TestRunner provee la funcionalidad para iniciar la corrida de test: “Run” que correrá todos los test que se le hayan agregado.

Se deberá especificar el tipo de output al “Run” por parámetro. Utilizamos el patrón Adapter a este propósito: cinco posibilidades para el output, 1) por consola “ConsoleComponent”, 2) por archivo plano “FileComponent”, 3) archivo XML “XMLComponent”, 4) consola y archivo plano ”DuoComponent” y 5) consola, archivo plano y XML “TriComponent”.

Una vez concluida la corrida de los test, generará un archivo único en el directorio actual con el formato “TestReport\_<fecha>” para el archivo plano y “XMLTestReport\_<fecha>” para el archivo XML.

También se deberá especificar un IFilter a utilizar, la librería trae una implementación del mismo (Filter) la cual permite filtrar por nombre de suite, test, tags y también, si se le especifica un inputComponent, por test valido de la última corrida.

Para esto último, la librería provee dos component capases de ser utilizados como input/output, storeFileComponent y storeDbComponent.

Para el caso de necesitar un output XML, el framework utiliza una jerarquía de nodos (dada por la interface “IXmlNode”) para guardar la información de de la corrida de cada test. El XMLComponent consta de un “XMLRoot” donde se agregaran nodos “XMLSuite” y “XMLCase” según corresponda. Una vez terminada la corrida, recorrerá la jerarquía de nodos y generará el archivo en base un schema.

Las clases que implementan ITest reciben un contexto “fixture” por parámetro en sus métodos “Run”, “Setup” y “TearDown”. El fixture es una herramienta que se le provee al usuario para que pueda configurar su ambiente de test. Es inicialmente instanciado por el test Runner. La responsabilidad del manejo del contexto es del usuario, es decir el mismo se encargará de configurarlo y volverlo a su estado inicial utilizando los métodos “Setup” y “Teardown” para este propósito.

De una manera similar al fixture, el reporte es instanciado por el testRunner e injectado a los test mediante el método “Run” cada suite se encargará de setearle al reporte los resultados de sus TestCases y a su vez de pasarle el reporte a sus Suites hijos.   
La librería también provee una clase dónde se encuentran los métodos Assert que soporta el framework.