Técnicas de Diseño
75.10

Trabajo Práctico 2.3:

Framework para tests unitarios (4ª parte)

Grupo 15:
Emiliano Suárez
Federico Rodriguez
Leandro Miguenz

Agregados hechos al TP original

- Para poder validar que un test no supere cierto límite de tiempo de ejecución y, en tal caso, que falle (por timeout), se agregó el atributo timeOut a la clase TestComponent y el método checkTimeOut a TestCase.

- Se creó el paquete "stores", con las clases PersistenceManager y SerializedObjectManager, que se encargan de serializar y deserializar los objetos de Suite. Al correr los tests, serializa y deserializa, volviendo a mostrar los mismos resultados de la corrida anterior, que fueron almacenados en disco. Para posibilitar esto, se hizo que TestComponent y Assertion implementen la interfaz Serializable. TestComponent (el elemento genérico del patrón composite) tiene un atributo protegido del tipo PersistenceManager.

- SerializedObjectManager hereda de la clase abstracta PersistenceManager, lo que hace que el modelo sea extensible, permitiéndole al usuario crear sus propios “store” que implementen diferentes maneras de almacenar la información de los tests, y setearlos a la suite.

- A TestComponent se le agregó un atributo booleano, que puede ser seteado por el usuario para decidir si correr sólo los tests fallidos de la corrida anterior. En caso de que no haya una corrida previa, se lanza una excepción.

Problemas encontrados

Debido a que en este framework no se crea una instancia de una clase por cada test (como ocurría en el nuestro), sino que los tests son métodos y el método run de la suite corre todos los tests que contiene, no pudimos encontrar una manera simple y eficiente de extender el diseño preexistente para implementar realmente la funcionalidad que le permita al usuario correr sólo los tests que fallaron.

Se consideró hacer una refactorización importante para modificar esto, y que efectivamente se tenga un test individual por cada clase, cada uno de los cuales implementaría su propio método run, pero ante la falta del tiempo necesario, no nos fue posible realizar semejante cambio. Se tuvo en cuenta también que tal refactorización hubiera implicado modificar todos los tests que están hechos, debido a los cambios que se hubieran producido en la interfaz de usuario.