TDD
[Test-Driven Development]

Consiste en implementar las pruebas de unidad
antes incluso de comenzar a escribir el código de un módulo.

Las pruebas de unidad consisten en
comprobaciones (manuales o automatizadas)
que se realizan para verificar que el código
correspondiente a un módulo concreto de un sistema software
funciona de acuerdo con los requisitos del sistema.

Tradicionalmente,
las pruebas se realizan a posteriori

Los casos de prueba se suelen escribir después de implementar el módulo cuyo funcionamiento pretenden verificar.

Como mucho, se preparan en paralelo si el programador
y la persona que realiza las pruebas [*tester*] no son la misma persona.

En TDD,
las pruebas se preparan antes de comenzar a escribir el código.

Primero escribimos un caso de prueba
y sólo después implementamos el código necesario
para que el caso de prueba se pase con éxito

Aunque pueda parecer extraño, TDD ofrece algunas ventajas:

- Al escribir primero los casos de prueba, definimos de manera formal los requisitos que esperamos que cumpla nuestra aplicación.

  Los casos de prueba sirven de **documentación** del sistema.

- Al escribir una prueba de unidad, pensamos en la forma correcta de utilizar un módulo que aún no existe.

  Hacemos hincapié en el diseño de la **interfaz** de un módulo antes de centrarnos en su implementación (algo siempre bueno: "la interfaz debe determinar la implementación, y no al revés").

- La **ejecución** de los casos de prueba se realiza de forma **automatizada** (por ejemplo, con ayuda de JUNIT)

  Al ejecutar los casos de prueba detectamos si hemos introducido algún error al tocar el código para realizar cualquier cambio en nuestra aplicación (ya sea para añadirle nuevas funciones o para reorganizar su estructura interna [refactorización]).

- Los casos de prueba nos permiten **perder el miedo** a realizar modificaciones en el código

  Tras realizar pequeñas modificaciones sobre el código, volveremos a ejecutar los casos de prueba para comprobar inmediatamente si hemos cometido algún error o no.

- Los casos de prueba definen claramente cuándo termina nuestro **trabajo** (cuando se pasan con éxito todas los casos de prueba).

El **proceso de construcción de software** se convierte en un ciclo:

1. Añadir un nuevo caso de prueba
   que recoja algo que nuestro módulo debe realizar correctamente.

2. Ejecutar los casos de prueba
   para comprobar que el caso recién añadido falla.

3. Realizar pequeños cambios en la implementación
   (en función de lo que queremos que haga nuestra aplicación).

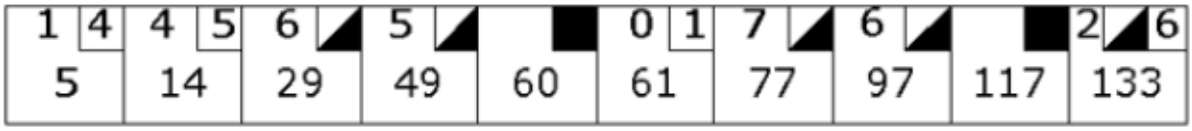
4. Ejecutar los casos de prueba
   hasta que todos se vuelven a pasar con éxito.

5. Refactorizar el código para mejorar su diseño (eliminar código
   duplicado, extraer métodos, renombrar identificadores…)

6. Ejecutar los casos de prueba
   para comprobar que todo sigue funcionando correctamente.

7. Volver al paso inicial

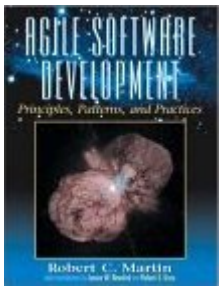
Caso práctico: Bolera



El problema consiste en…
obtener la puntuación de un jugador en una partida de bolos.

Una posible solución…
y el proceso seguido para obtenerla en

"The Bowling Game. An example of test-first pair programming"
© Robert C. Martin & Robert S. Koss, 2001
http://www.objectmentor.com/resources/articles/xpepisode.htm



Robert C. Martin:
"Agile Software Development: Principles, Patterns, and Practices".
Prentice Hall, 2003. ISBN 0-13-597444-5.