**Código Muerto.**

**Descripción del problema:**

* Una variable
* Sentencia
* Parámetro
* Campo
* Método o clase (ya no se utiliza)

**Consecuencias:**

* Menor comprensibilidad
* Menor capacidad de mantenimiento
* Posibles riesgos de seguridad

**Causas:**

* Refactorización del código para mejorar la calidad
* Mejoras en el diseño para mejorar su calidad
* Cambio en los requisitos, provocan cambios en el diseño, y en el código

**Evitarlo:**

* Tener un buen IDE que arroje un mensaje que diga algo está "muerto"
* Paseos por el código

**Reconocimiento:**

* Mismo conocimiento.

**Extracción:**

* Eliminar el código muerto
* Hacer comentarios sobre la decisión de diseño
* Utilizar un sistema de control de versiones, como Git
* Mantener el diseño actualizado

**Generalidad especulativa.**

**Descripción del problema:**

* Programación basada en especulaciones sobre lo que podría ser necesario

**Consecuencias:**

* Menor comprensibilidad
* Menor capacidad de mantenimiento

**Causas:**

* Los desarrolladores adquieren hábitos y crean ciertos tipos de clases o métodos
* Los desarrolladores tienen en cuenta los cambios en el sistema (es bueno), pero reaccionan de forma inapropiada.

**Evitar:**

* Dar prioridad a las características
* Permitir el cambio en los lugares adecuados

**Reconocimiento:**

* Tener cuidado con las jerarquías y clases de generalización o especialización profundas que no sirven para nada

**Extracción:**

* Eliminar las abstracciones innecesarias
* Colapsar las jerarquías de clases cuando sea necesario o posible

**Campo temporal**

**Descripción del problema:**

* Los campos temporales obtienen sus valores sólo en determinadas situaciones.

**Consecuencias:**

* Menor extensibilidad
* Menor comprensibilidad
* Menor capacidad de mantenimiento

**Causas:**

* Depende de los antecedentes de los desarrolladores (primeros lenguajes), pueden tener los hábito de definir todas sus variables en la parte superior de sus clases o métodos, en lugar del contexto en el que se necesitan:
* No pensar en el alcance o el acoplamiento
* No pensar desde una perspectiva orientada a objetos

**Evitar:**

* No defina una variable en la parte superior del método si sólo se necesita dentro de un bucle o bloque condicional
* Definir también una variable
* No defina un atributo de objeto si sólo se necesita en un único método
* No defina un atributo de clase estática si sólo se necesita dentro del contexto de un objeto
* Dar prioridad a las características

**Reconocimiento:**

* Acoplamiento innecesario
* Variables que no se utilizan o son nulas

**Extracción:**

* Repensar las abstracciones y encapsulaciones
* Encapsular las estructuras de datos