Art Of Enbugging

Enbuggear (enbugging) el código significa poner bugs en el mismo. El texto habla de que es muy raro el caso de se agreguen bugs de forma indirecta, o sea, que son agregados siempre por el programador.

Una de las mejores formas de dejar afuera los bugs es mantener una buena "separation of concerns", es decir, que cada clase tenga unas responsabilidades aisladas, bien definidas, y que sean claras.

Uno de los objetivos principales es escribir "código tímido", es decir, que no revele mucho de si mismo a nadie más, y que tampoco hable mas de lo necesario con los demás objetos.

1 Tell Don't Ask:

El código procedural tiende a obtener la información, y de ahí tomar decisiones basadas en la información obtenida. En cambio, la programación orientada a objetos simplemente le dice al objeto que hacer.

Entonces, no se busca preguntar al objeto por su estado, y de ahí tomar una decisión, sino simplemente decirle al objeto que hacer, esto sería la responsabilidad del objeto. El no cumplir esto viola el encapsulamiento, y deja la puerta abierta a los bugs.

2 Ley de Demeter para funciones:

Mientras más objetos te hables, mayor es el riesgo de romper algo cuando se cambie alguno de los objetos que se comunican, por lo tanto, la idea es que se hablen lo menos posible.

La Ley sugiere que un objeto solo debería comunicarse con:

- El mismo.
- Los parámetros pasados por algún método.
- Cualquier objeto que haya creado.
- Cualquier objeto que esté relacionado directamente (atributo).

También, los métodos no deberían dar más información de la que deben. Viendo en siguiente ejemplo:

```
my_television.front_panel. switches.power.on();
```

Acá estas diciendo que la televisión tiene un panel que esta en frente, y que este panel tiene interruptores, y que estos mismos lo puedo cambiar de tal forma para que la televisión se prenda. Entonces, debería ser:

```
my_television.power_up();
```

De esta forma, no estás dando información acerca de los atributos de la televisión, aunque notar, que lo único que haría este método sería llamar a un método del panel del frente, para que haga lo mismo con los interruptores.

La desventaja de hacer esto, es que terminas escribiendo muchos métodos chicos que solo llaman a otros métodos, estos métodos se llaman "wrapper", es decir, son una envoltura para no develar información demás. Esto, indudablemente trae un costo de eficiencia, pero si no se tuviese, habría un acoplamiento (una dependencia) de las clases superiores, y esto es grave. El acoplamiento de las clases superiores solo es aceptable cuando la velocidad, y eficiencia del código es primordial.