Notas sobre o capitulo 8 de Clean Code – by Lucas Trevizan

Limites

Quase nunca o código do nosso software é apenas nosso ***(inclusive já dependi de código do cliente para poder fazer uma implementação)*** e esse capitulo vai abordar como manter limpo o código dentro dos limites do nosso software.

O uso de código de terceiros

Quando implementamos o código de terceiros enfrentamos um dilema. As **interfaces fornecidas tem que ser o mais abrangente possível** para serem utilizadas em diversos casos diferentes, dado essa informação, o que podemos fazer quando essa interface provém métodos que não queremos ter na nossa aplicação, ou que não queremos que alguém utilize em outro momento do programa? Ele fornece a nós mais do que precisamos ou queremos.

Uma boa saída que nos é dada, é encapsular essa Interface de modo privado em uma classe, dessa maneira mantemos a implementação dentro dos limites que queremos.

Explorando e aprendendo sobre limites

A ideia de se utilizar códigos de terceiros é conseguir mais funcionalidades sem precisar "reinventar a roda". Mas e se não for claro para nós como devemos usar essas bibliotecas, podemos usar a abordagem dos ***testes de aprendizado***.

Basicamente isso é chamar a API externa e focaríamos em testar o que achamos que sabemos daquela API.

Aprendendo sobre o log4j

*****Engraçado ler sobe o log4j aqui, recentemente descobriram uma vulnerabilidade nele – 23/12/2021*****

Aqui tem um bom exemplo de como houve uma dificuldade inicial para se adaptar ao lo4j e como escrever testes de unidade ajudou a identificar o comportamento da biblioteca em assuntos que não estavam tão claros.

Basicamente ele aplicou o conceito de ***"testes de aprendizado"*** na prática para entender melhor o log4j.

Os testes de aprendizagem são melhores que de graça

Se vamos ter que fazer a **implementação de uma biblioteca externa**, devemos **de qualquer jeito saber como ela funciona**. Então por que não simplesmente escrever testes de aprendizado simulando o que essa biblioteca deveria fazer no seu fonte? **É DE GRAÇA.**

Fora isso teríamos testes para rodar toda vez que a biblioteca tivesse um update e isso possibilitaria para nós ter o conhecimento **RÁPIDO** se algo tiver quebrado nossa aplicação, ou se ela ainda funciona como deveria.

O uso de código que não existe ainda

Novamente aqui é mencionado o limite através de encasulamento, num bom exemplo fornecido de que eles precisavam lidar com uma API ainda não desenvolvida. Então como recurso criaram uma interface que encapsularia os dados que essa API mandaria e então mesmo sem a API foi possível desenvolver em cima dessa Interface.

Como eu disse anteriormente, também já precisei trabalhar **(e muito)** com códigos do cliente que eram completamente escondidos e apenas fornecidos para nós. Isso obviamente g**erou erros em métodos pouco funcionais**, a partir daí era fácil enxergar **ONDE MEU CÓDIGO DEVIA PARAR** e onde o deles começava, os limites ficam mais estabelecidos.

Limites limpos

Limites bem estabelecidos quando se usa códigos de terceiros é o fato diferencial para que seu código seja facilmente manutenível. Muito se abordou sobre encapsular esses códigos pra focarmos somente no que precisamos dele, sem conhecer suas implementações e isso faz com que tenhamos limites bem estabelecidos e nunca nos tornemos refém do código de alguém.

É melhor controlar do que ser controlado.