Pedro Henrique Pires Rodrigues - Engenharia de Software - 4º periodo
Resenha: Big Ball of Mud

O artigo Big Ball of Mud, por Brian Foote Joseph Yoder. Trata sobre a arquitetura de software de mesmo nome, que consiste em um código desorganizado. Este que é composto por quebra padrões e princípios como SOLID, código duplicado e fortemente acoplado, documentação pouca ou desatualizada, entre outros fatores que tornam seu entendimento, e por consequência sua manutenção muito mais difícil. Nesse sentido, a obra busca trazer reflexões sobre esses softwares de modo a entender o que leva um software a ter uma arquitetura desorganizada, e qual a relevância disso para a produção de um sistema.

A tradução do nome do artigo para português é “grande bola de lama”, o que refere ao fato de que um software que encaixe nessa definição chega a esse ponto, em sua maioria das vezes, por uma série de fatores, esses que são apresentados no artigo na sequência:

THROWAWAY CODE
PIECEMEAL GROWTH
KEEP IT WORKING
SWEEPING IT UNDER THE RUG
RECONSTRUCTION

Sendo apresentados, na ordem, quase que como um passo a passo para se ter um código mal escrito.

O primeiro item da lista “THROWAWAY CODE”, traduzindo para o português: “código descartável”. É o nome dado ao código feito de forma rápida, para tardiamente ser substituído por um mais elegante. Este vem da necessidade de se fazer uma entrega, cujo tempo oferecido não é suficiente para se fazê-la da melhor forma possível. A partir dessa necessidade, é feita uma “gambiarra” capaz de mitigar o problema, mas que quando não melhorada pode virar uma grande bola de lama.

Quanto ao PIECEMEAL GROWTH, que pode ser traduzido para “crescimento aos poucos”, se refere às constantes mudanças nas regras de negócio de um programa. Essas que de pouco em pouco vão tornando o código cada vez mais poluído com o tempo, contribuindo para a desorganização da arquitetura.

Paralelamente o KEEP IT WORKING, “mantenha funcionando”, é uma estratégia usada quando por mais que seja necessária uma grande melhora no código não há como simplesmente abandonar o software que já está rodando. Nesse caso, por mais que algumas modificações no código atual piorem ainda mais sua legibilidade, por muitas vezes ainda serão executadas para que o programa continue funcionando como está.

SWEEPING IT UNDER THE RUG, ou “jogando para debaixo do tapete”, faz uma alusão ao fato de varrer a casa, mas ao invés de jogar a sujeira fora, jogá-la para debaixo do tapete de forma que não fique visível. Apesar de por um lado

melhorar a legibilidade do código, é mais próximo de um método paliativo que de uma solução de fato, e se não for revista, vai colaborar para um código mais sujo.

Por fim RECONSTRUCTION, ou “reconstrução”, é a total reelaboração do software. Esta já é uma definição mais direta, quando se chega em um certo ponto, é inevitável que se refaça o sistema do zero, do contrário, a manutenção ficará ainda mais cara.

Mas para que um software chegue a se tornar uma grande bola de lama, existem algumas forças que colaboram para isso, segundo os autores essas forças são: Tempo, experiência, rotatividade, habilidade, complexidade, mudança e custo.

“A pressa é inimiga da perfeição” esse ditado popular evidencia de forma clara o problema que a falta de tempo gera. Um software profissional costuma ser feito por meio de prazos que devem ser cumpridos, e algumas das vezes a falta de tempo que esses prazos geram, fazem com que tenhamos que produzir códigos de pior qualidade para que possamos entregá-lo a tempo.

Um ponto interessante do artigo, é que ele não fala somente da experiência no sentido de conhecimento técnico, mas também do domínio do negócio. Nesse sentido, o artigo destaca os dois pontos como algo importante a se considerar, tanto ao falar do desenvolvedor quanto ao falar do próprio arquiteto do software em si.

Alta rotatividade pode não ser benéfica para o ciclo de vida de um software, pois a constante troca dos envolvidos na produção do mesmo pode causar “estragos na memória institucional”.

Alguns desenvolvedores têm mais facilidade quanto a criação de abstrações ou em como aplicar princípios, como SOLID, do que outros programadores. Além de possuirem preferências diferentes em questão de tecnologias ou linguagens.

O artigo também afirma o fato de existirem sistemas que são complexos por natureza, de modo que suas regras de negócio por si só já geram alta complexidade. Para os profissionais lidarem com isso é necessário muito tempo de experiência e conhecimento do domínio, o que só se é obtido com o tempo.

Mudanças, algumas são críticas de forma a ser necessário até mesmo uma revisão do design no sistema, mas muitas vezes é resolvido com um simples “THROWAWAY CODE”, para não atrasar a produção.

Custo, pessoalmente, eu acredito que esse seja o principal fator em todas as escolhas feitas durante o desenvolvimento de um software profissional. O fato de uma tarefa ser muito custosa faz com que ela seja sempre evitada ao máximo, ainda mais quando muitos, segundo o autor, veem uma boa arquitetura como um luxo e não como algo obrigatório.

Um software que sofreu uma total reconstrução a um tempo atrás foi o próprio Facebook, seus algoritmos, principalmente o de recomendação, não estavam preparados para lidar com todos os dados que estavam recebendo. Não era esperado que o projeto fosse crescer ao ponto que cresceu, tornando a arquitetura da época inviável para o tamanho que o software tinha alcançado. Esse acontecimento se encaixa que praticamente todos os pontos levantados pelo autor, onde por mais caro que fosse a completa reelaboração do software, era praticamente impossível mantê-lo com aquela arquitetura.

O artigo conclui dizendo que não está condenando os trabalhos que porventura se tornaram grandes bolas de lama, lidar com essas forças que nos forçam a recorrerem a soluções menos elegantes é um problema constante na vida de todo produtor de software. Ainda sim, acredito que se atentar aos “passos” que geram essa grande bola de lama, e tomar cuidado com as forças que levam ao uso desses, é algo que podemos fazer para evitar a deterioração da arquitetura dos software que produzimos. Logo facilitando a manutenção desses sistemas a longo prazo.