

## Resenha 'Big Ball of Mud'

O artigo "Big Ball of Mud", escrito por Brian Foote e Joseph Yoder, faz uma crítica bem interessante sobre algo que é super comum na engenharia de software: aquela arquitetura toda bagunçada que eles chamam de "Big Ball of Mud". Esse termo é usado para descrever sistemas de software que parecem não ter uma arquitetura clara e definida, resultando numa estrutura bem confusa e difícil de entender. Eles exploram o que caracteriza esse tipo de arquitetura, as razões pelas quais ela surge, as consequências disso, e ainda sugerem algumas formas de lidar com o problema, embora, sejamos honestos, nem sempre é tão simples.

Basicamente, um "Big Ball of Mud" acontece quando um sistema vai crescendo sem muito planejamento, resultando em uma espécie de "código espaguete". Isso quer dizer que não há uma organização clara, e os limites dentro do sistema ficam turvos, o que torna a manutenção um verdadeiro pesadelo. Além disso, essas soluções desorganizadas acabam gerando algo que chamamos de "dívida técnica", que é quando você acumula problemas futuros por resolver as coisas rápido e sem o devido cuidado, mas, infelizmente, isso é super comum no dia a dia.

Várias razões explicam o surgimento dessa bagunça. Muitas vezes, a pressão por prazos curtos faz com que os desenvolvedores optem por soluções rápidas e nem sempre sustentáveis. Além disso, a falta de experiência ou de conhecimento arquitetônico por parte dos desenvolvedores também pesa. Quando o sistema cresce aos poucos, sem um plano arquitetônico firme, ele acaba ficando cada vez mais complicado. Sem falar que, muitas vezes, é mais tentador atender às necessidades imediatas do que se preocupar com a qualidade de longo prazo.

No artigo, os autores mencionam quatro padrões que ajudam a formar o "Big Ball of Mud". Um deles é o "Throwaway Code", aquele código que deveria ser temporário, mas acaba ficando lá para sempre. Outro é o "Piecemeal Growth", que é o crescimento desorganizado do sistema com o tempo. "Keep It Working" é aquela mentalidade de "deixa funcionando, tá bom assim", mesmo que tudo esteja uma bagunça. Por fim, o "Shearing Layers" fala sobre como diferentes partes do sistema evoluem de maneiras descoordenadas, gerando inconsistências.

Os impactos de um "Big Ball of Mud" são bem chatos. O sistema fica mais caro de manter, os desenvolvedores perdem produtividade, e a escalabilidade do sistema é comprometida. Tentar entender ou alterar o código vira uma missão quase impossível, e aí aparecem mais bugs e falhas. Adicionar novas funcionalidades ou melhorar o desempenho sem bagunçar o que já existe é uma tarefa ingrata.

Mas o artigo dá algumas dicas de como resolver isso, ou pelo menos minimizar o estrago. Refatorar o código com frequência é uma boa prática, já que ajuda a reduzir a dívida técnica. Outra sugestão é implementar uma arquitetura em

camadas, separando a apresentação dos dados e da aplicação, para tornar tudo mais modular. Revisões de código e seguir boas práticas também são super importantes para manter o sistema mais organizado. Dividir o sistema em módulos ou microserviços pode ajudar a controlar a complexidade, e a automação de testes é fundamental para garantir que as mudanças não quebrem o sistema.

Mesmo com todas essas desvantagens, os sistemas "Big Ball of Mud" são super comuns, principalmente porque muitas vezes o foco é entregar resultados rápidos. E convenhamos, é difícil justificar o tempo e o custo de refatorar o sistema quando a funcionalidade está funcionando (mesmo que seja meio capenga). Infelizmente, sem uma responsabilidade clara entre as equipes, a tendência é que a arquitetura vá se degradando cada vez mais.

O artigo conclui que, apesar de todo mundo querer fugir de uma arquitetura "Big Ball of Mud", ela é quase inevitável diante das pressões práticas que a gente enfrenta. A chave está em reconhecer os sinais da bagunça e aplicar práticas de manutenção e reescrita adequadas para manter o sistema funcional e sustentável a longo prazo. Mas não dá para mentir, é um trabalho que nunca acaba e exige disciplina.