Aluno: Vitor Veiga Silva

The Big Ball of Mud

O artigo “The Big Ball of Mud”, de Brian Foote e Joseph Yoder, fala de uma verdade que todo programador já viu de perto: a maior parte dos sistemas no mundo real não segue modelos como aprendemos, como Pipeline ou Arquitetura em camadas. Em vez disso, os sistemas acabam virando uma grande mistura caótica e desorganizada de código, feita às pressas, com vários remendos e improvisos. Essa forma de organização ou a falta dela é que os autores do artigo chamam de Big Ball of Mud(“Grande Bola de Lama”). É aquele código onde cada parte parece ter sido encaixada onde coube, mas sem muita reflexão sobre o futuro do sistema. Ele até funciona, mas não é claro e principalmente não é fácil de manter.

Os padrões que constroem a bagunça, no artigo, Foote e Yoder identificam cinco padrões que frequentemente aparecem nesse tipo de arquitetura que são:

1- Código Descartável (Throwaway Code): É aquele código feito para ser temporário, criado com o pensamento que depois será refatorado. O problema é que na maioria das vezes o código nunca é refatorado e esse código improvisado continua ali, sustentando partes críticas do sistema por meses ou até por anos.

2- Crescimento Fragmentado (Piecemeal Growth): O sistema começa a receber novos requisitos sem um plano geral as funcionalidades são adicionadas diretamente no código existente, sem uma organização clara, apenas para atender demandas urgentes.

3- Mantenha-o Funcionando (Keeping it Working): Quando algo quebra, a prioridade é corrigir rapidamente, e não melhorar a qualidade do código. Pequenas gambiarras resolvem o problema de forma imediata, mas acumulam complexidade e fragilidade.

4- Escondendo a Bagunça (Sweeping it under the Rug): Quando o caos do código fica insuportável, uma solução comum é criar camadas de abstração para escondê-lo. A parti visível parece até organizada, mas por dentro o sistema continua confuso.

5- Reconstrução (Reconstruction): Chega a um momento em que a bagunça é tão grande que a única solução é começar de novo. O sistema antigo é descartado, e

um novo é construído com base nas lições aprendidas com os erros do antigo ou pelo menos assim se espera.

A Grande Bola de Lama não é algo que nasce por incompetência pelo ao contrário, ela é fruto de pressões reais como: prazos apertados, orçamento limitado, mudanças constantes nas necessidades do cliente, falta de experiência ou de mão de obra suficiente. Em outras palavras, ela é uma consequência natural do mundo real, e não apenas um erro técnico.

No artigo os autores fazem comparações com “favelas” e “código espaguete” que ajudam a visualizar bem o conceito. É como uma construção improvisada: pode até funcionar, mas foi feita com o que havia disponível naquele momento e sem um projeto sólido acaba tornando difícil expandir ou reformar. O problema é quando mais o sistema cresce dessa maneira, mais difícil e caro fica fazer qualquer alteração. Pequenas mudanças podem gerar efeitos colaterais imprevisíveis, e o tempo gasto para implementar novas funcionalidades aumenta exponencialmente.

Foote e Yoder reforçam que a arquitetura não é só sobre diagramas e padrões. Ela depende das pessoas que estão desenvolvendo o sistema. Ferramentas adequadas, comunicação clara e colaboração entre os membros da equipe são fundamentais para evitar que um projeto acabe se tornando uma Grande Bola de Lama. Os autores concluem que a arquitetura é o resultado de transformar experiência em sabedoria. Ou seja, quanto mais cedo o time de desenvolvimento aprender com seus erros e acertos, mais capaz será de tomar decisões que evitem (ou pelo menos controlem) a lama no futuro.

Um exemplo é uma empresa que lança sua plataforma de e-commerce com prazos curtos e pouco orçamento e para agilizar cria um checkout improvisado (Código Descartável) que no final acaba virando definitivo. Com o crescimento da plataforma, novos recursos são adicionados sem planejamento (Crescimento Fragmentado), tornando o código cada vez mais confuso. Quando surgem bugs, a equipe opta por correções rápidas em vez de refatorar ó código (Mantenha-o Funcionando). E com o tempo, o sistema fica tão complicado que criam uma API para isolar a parte mais crítica (Escondendo a Bagunça). Anos depois, o código antigo se torna um grande obstáculo e a única saída é reescrever o código do zero (Reconstrução).