RESENHA CRITICA DO ARTIGO "BIG BALL OF MUD"

Resenha da obra: FOOTE, Brian; YODER, Joseph. Big Ball of Mud. Nomeda **Department of Computer Science University of Illinois at Urbana-Champaign**, 1997.

A presente resenha se presta a analisar o artigo Big Ball of Mud de Brian Foote e Joseph Yoder.

O artigo se divide em Resumo, introdução, nos capitulos: *BIG BALL OF MUD, THROWAWAY CODE, PIECEMEAL GROWTH, KEEP IT WORKING, SWEEPING IT UNDER THE RUG e RECONSTRUCTION*, conclusão e reconhecimentos.

O texto abordado gira em torno de um paradoxo: Mesmo que a maioria dos engenheiros de software enfatizem arquiteturas limpas e bem organizadas, a grande maioria dos softwares (e softwares grandes) são "bolas de barro"(Big ball's of mud) - Sem estruturas definidas, aglomerados de codigos e mas praticas, mas que funcionam apesar da desorganização - .

Os autores identificam seis padrões: *BIG BALL OF MUD, THROWAWAY CODE, PIECEMEAL GROWTH, KEEP IT WORKING, SWEEPING IT UNDER THE RUG e RECONSTRUCTION* e se prestam a discorrer sobre cada um nos capítulos do artigo.

BIG BALL OF MUD (Grande bola de lama): Um padrão que a princípio parece ser um anti-padrão, mas que os autores argumentam ser um padrão legítimo em certas circunstâncias. É feito sem planejamento, comparado no texto a favelas, construído sem arquitetura definida, muitas vezes com códigos ou ferramentas impróprias, que geram custos de manutenção elevados. Uma BIG BALL OF MUD se trata de um código nascido da conveniência, pode ser causado por: falta de experiência, falta de tempo, complexidade do problema e requisitos inesperados.

THROWAWAY CODE (Código Descartável): Caracterizado no artigo como um código rápido e sujo, destinado a ser usado apenas uma vez, mas que acaba persistindo e muitas vezes gerando sistemas construídos ao redor dele. Surge de protótipos que nunca foram descartados, soluções temporárias, demonstrações ou falta de tempo para reescrita adequada.

PIECEMEAL GROWTH (Crescimento Fragmentado): Um software bem-sucedido que apresenta um crescimento além do esperado, adicionando mais requisitos que podem ir contra sua estrutura inicial, comprometendo a arquitetura por um crescimento não planejado - comparado com cidades que crescem por demanda populacional. Os autores notam que este crescimento incremental também pode ser uma estratégia válida de desenvolvimento quando bem gerenciado.

KEEP IT WORKING (Mantenha Funcionando): Estratégia onde, uma vez que um sistema está funcionando, é imperativo mantê-lo funcionando. São sistemas que precisam estar operacionais 24/7, onde modificações que possam quebrar o sistema são evitadas, priorizando pequenas mudanças incrementais que preservem a funcionalidade.

SWEEPING IT UNDER THE RUG (Varrer para Debaixo do Tapete): "Se você não pode fazer uma bagunça desaparecer, pelo menos pode escondê-la". Trata-se de isolar trechos de

código mal elaborados em módulos específicos e criar interfaces limpas para interagir com eles, disfarçando a complexidade interna e preparando terreno para eventual refatoração.

RECONSTRUCTION (Reconstrução): Quando o código declinou a um ponto crítico, a estratégia se torna descartá-lo e construir outro, aproveitando as lições aprendidas - tanto os acertos quanto os erros do código anterior. Pode ser causado por obsolescência, necessidade de mudanças estruturais ou falta de documentação para compreender o código original. Trata-se da solução final quando os outros padrões falham.

O artigo faz um trabalho pioneiro em categorizar padrões que são frequentemente ignorados ou menosprezados pela engenharia de software por serem considerados anti-arquiteturais. Os autores adotam uma abordagem neutra, não demonizando essas práticas, mas analisando objetivamente os benefícios e malefícios de cada padrão dentro de seus contextos específicos. Apesar de ter 28 anos, o artigo mantém notável relevância, antecipando conceitos que se tornaram centrais no desenvolvimento moderno. Por exemplo, o padrão THROWAWAY CODE se relaciona com discussões sobre dívida técnica, enquanto PIECEMEAL GROWTH assemelha práticas ágeis de desenvolvimento incremental. A abordagem dos autores sobre KEEP IT WORKING ressoa com práticas modernas de DevOps e integração contínua.

É importante reconhecer que alguns aspectos do artigo refletem o contexto tecnológico de 1997. As ferramentas de desenvolvimento, práticas de versionamento e metodologias ágeis evoluíram significativamente, oferecendo novas estratégias para lidar com os problemas identificados pelos autores. No entanto, essa evolução não invalida os "insights"fundamentais do artigo; pelo contrário, muitas soluções modernas podem ser vistas como refinamentos dos padrões descritos.

A atualidade das observações de Foote e Yoder demonstra que eles identificaram forças fundamentais no desenvolvimento de software (pressões de tempo, limitações de recursos, mudanças de requisitos e complexidade inerente dos problemas) que transcendem tecnologias específicas. Isso torna o artigo uma obra importante para compreender desafios contemporâneos do desenvolvimento de software.