Resenha “The Big Ball of Mud” - Pedro Henrique Maia Alves

O artigo começa introduzindo o que significa a metáfora da “Big Ball of Mud”, que são sistemas que crescem de maneira orgânica, sem um planejamento arquitetural, adquirindo uma estrutura desordenada à medida que novos requisitos são implementados. Apesar da bagunça em que esses softwares são escritos, eles funcionam e levam os desenvolvedores a um dilema: “Vale a pena reescrever?”, “É melhor reescrever gradualmente?”, “Devemos manter funcionando e aceitar as imperfeições”. Portanto, o artigo examina por que esses sistemas surgem, como sobrevivem e o que pode ser feito a respeito.

Por que estas “Grandes Bolas de Lama” surgem? Alguns motivos fazem com que estes problemas apareçam em projetos reais, como pressões por entregas, pois entregar dentro do prazo é mais valorizado que qualidade arquitetural bem definida. Evoluções incrementais adicionam gradualmente complexidade ao sistema que, caso não receba refatorações periódicas, acumula complexidade a ponto de tornar irreconhecível sua estrutura inicial. Equipes sem experiência geralmente não sabem aplicar padrões de projetos ou arquiteturas, ou seja, os códigos crescem de maneira não estruturada e sem regras. Refatorar ou manter? Muitas vezes, sistemas bagunçados são difíceis e caros de substituir, quer dizer que a solução é manter o que existe.

Algumas características são perceptíveis em sistemas Bolas de Lama, estas são:

- **Ausência de modularidade clara** – Componentes não têm fronteiras bem definidas; responsabilidades se misturam.
- **Acoplamento excessivo** – Módulos dependem fortemente uns dos outros, tornando difícil modificar qualquer parte isoladamente.
- **Baixa coesão** – Funcionalidades não relacionadas aparecem juntas no mesmo lugar.
- **Código duplicado** – Soluções parecidas são reescritas várias vezes em diferentes partes do sistema.
- **Nomes enganosos ou inconsistentes** – Funções e variáveis não refletem corretamente o que fazem.
- **Documentação insuficiente** – As regras de funcionamento estão “na cabeça” de poucos desenvolvedores experientes.
- **Mudanças perigosas** – Alterar algo pode quebrar funcionalidades inesperadas.

No artigo, foi identificado, alguns padrões de projetos que se relacionam com Grandes Bolas de Lama que transformam sistemas ou os condicionam a este problema. *Keep it Working* prioriza em manter o sistema funcionando acima de tudo,

mesmo que o código seja difícil de entender, se ele entrega valor, deixe como está. *Shearing Layers* onde camadas de negócios e interface mudam com frequência enquanto o núcleo se mantém, criando áreas do código intocadas e desatualizadas. *Sweeping It Under the Rug* problemas relacionados à estrutura são conhecidos porém muito trabalhosos de resolver, portanto, são ignorados pois não há tempo de resolver/não compensa arrumar. *Throwaway Code* trechos de códigos escritos rapidamente para fazer testes ou protótipos que acabam indo parar na produção, sem serem excluídos. *Big Ball of Mud* é um padrão não aconselhável, porém algumas empresas abraçam este modelo, pois a flexibilidade imediata é mais importante que a estrutura arquitetural.

Algumas estratégias são utilizadas para lidar com estes sistemas, para manter sua utilidade e até melhorar sua estrutura. Refatoração contínua é utilizada para limpar o código afetado a cada funcionalidade implementada, isto impede que o sistema se degrade ainda mais e aos poucos melhora a organização. Extrair blocos de códigos em módulos independentes reduz o acoplamento e melhora a capacidade de manutenção. Testes automatizados garantem que o comportamento atual de um software seja preservado, sem testes qualquer modificação se torna um risco alto. Documentação, mesmo que seja simples e útil, são vitais para novos desenvolvedores entrarem no projeto. Em alguns casos aceitar a realidade da situação é a melhor estratégia, focando em mantê-lo funcionando e aplicando correções mínimas ao longo do tempo.

O artigo conclui que estas “Grandes Bolas de Lama” são muito mais comuns no cenário atual do que descrito nos livros e cursos. Enquanto a teoria valoriza arquiteturas limpas e bem definidas, a prática mostra que sistemas bagunçados funcionam e continuam funcionando por anos. Estas bolas começam com sistemas pequenos, mas que sob pressão e mudanças rápidas, evoluem de forma inesperada e desordenada. O ponto central é que nem sempre é viável implementar a perfeição arquitetural e de padrões de projetos, nos contextos do cotidiano, a prioridade é manter o sistema entregando valor mesmo que signifique conviver com imperfeições.

No dia a dia do mercado, acredito que implementar módulos bem definidos, testes automatizados, implementar pequenos padrões de projeto e documentações úteis, ajudam um sistema a viver por mais tempo e ajuda o time de desenvolvimento a trabalhar de maneira mais eficiente, podendo navegar pelas diferentes partes do software e implementar novos requisitos sem haver dificuldades.