Resenha do Artigo Big Ball of Mud

Aluna: Ana Flávia de Carvalho Santos

Introdução

O artigo "Big Ball of Mud" começa falando sobre um problema que muitos desenvolvedores conhecem bem: a diferença entre a teoria e a prática na hora de criar softwares. Existem padrões de arquitetura muito bons, que prometem organização e eficiência, mas, na realidade, muitos sistemas acabam virando um "Big Ball of Mud", ou seja, um emaranhado de código confuso e difícil de lidar, ou, uma bola de lama. A pergunta que os autores fazem é: por que isso acontece com tanta frequência?

Em vez de fingir que o problema não existe, o artigo propõe a compreensão de como um sistema organizado se transforma nessa grande bola de lama. Os autores investigam as causas, as consequências e, principalmente, o que podemos fazer para evitar ou consertar a bagunça. Eles levantam questões importantes, como, por exemplo, se essa bagunça é inevitável em qualquer software grande ou se existem maneiras de evitá-la desde o começo. Portanto, o artigo nos convida a pensar sobre os desafios da arquitetura de software na prática.

Padrões

O "Big Ball of Mud" é o nome dado a um sistema de software que se desenvolveu sem uma estrutura clara. Ele se torna um amontoado de códigos interligados, com conexões complicadas e falta de organização. Isso geralmente acontece quando há pressão para entregar rápido, sem planejamento de longo prazo e ao usar soluções rápidas que causam problemas não esperados.

"Throwaway Code" é quando você cria código rapidamente, sem se preocupar muito com a qualidade ou design, pensando em usá-lo só uma vez e depois jogá-lo fora. O problema é que muitas vezes esse código "temporário" acaba sendo usado no sistema principal porque funciona, mesmo que mal, e por falta de tempo ou recursos para refazê-lo. Isso contribui para o "Big Ball of Mud", pois introduz código mal escrito e desorganizado no sistema.

"Piecemeal Growth" é quando o software cresce aos poucos e de forma desorganizada, com novas funções sendo adicionadas sem um plano geral de como elas se encaixam na estrutura. É como cidade sem planejamento, como o próprio

artigo menciona, adicionando ruas aleatoriamente. Isso acontece pela necessidade de atender a novas demandas rapidamente, mas a longo prazo, torna o sistema frágil e desunido.

"Keep it Working" é a prioridade máxima de manter o sistema funcionando, mesmo que isso signifique usar soluções rápidas e ruins para corrigir problemas. A pressão para manter o sistema disponível, o medo de quebrar o sistema com grandes mudanças e a falta de tempo para investigar as causas dos problemas motivam essa prática. Embora seja compreensível em algumas situações, pode levar ao acúmulo de soluções temporárias e aumentar a complexidade do sistema a longo prazo.

"Sweeping it Under The Rug" é quando você esconde ou disfarça problemas no código em vez de realmente resolvê-los. Fazer interfaces bonitas para esconder código ruim, isolar partes problemáticas sem realmente corrigi-las e até mesmo ignorar os problemas esperando que eles sumam sozinhos são exemplos dessa prática. Isso pode acontecer por vergonha, falta de tempo ou pela sensação de que o problema é muito grande para ser enfrentado.

"Reconstruction" é a decisão, muitas vezes drástica, de jogar fora o sistema existente e reconstruí-lo completamente do zero. Isso se torna necessário quando o sistema está tão deteriorado que os custos de manutenção se tornam insustentáveis, quando a tecnologia se torna obsoleta ou quando as mudanças necessárias são tão profundas que adaptar o código existente se torna inviável. Apesar de ser uma medida extrema, a "Reconstrução" pode ser a única saída para sistemas que se tornaram verdadeiros "Big Balls of Mud".

Como a Big Ball of Mud é criada

Um "Big Ball of Mud" geralmente não é planejado. Ele surge devagar, impulsionado por decisões que parecem inofensivas, mas que, com o tempo, criam um emaranhado de códigos e dependências. No começo, a equipe, sob pressão para entregar rápido, pode usar o "Throwaway Code". O importante é fazer funcionar, mesmo que seja com soluções rápidas e não muito boas. O problema é que esse código "temporário" muitas vezes acaba sendo usado no sistema principal porque não há tempo ou recursos para refazê-lo.

Com o passar do tempo, novas funções precisam ser adicionadas, e a equipe, ainda sob pressão, escolhe o caminho mais rápido: o "Piecemeal Growth". Sem um plano geral, o sistema cresce de forma desorganizada, com novos módulos e conexões

surgindo sem considerar o impacto na estrutura como um todo. Essa falta de planejamento leva à duplicação de código, interfaces inconsistentes e dependências confusas, tornando o sistema cada vez mais difícil de entender e manter.

Para complicar ainda mais, entra a necessidade do "Keep it Working". Falhas e instabilidades são inaceitáveis, especialmente em sistemas críticos, o que faz a equipe priorizar soluções rápidas e localizadas em vez de soluções mais robustas e duradouras. A cada nova solução temporária aplicada para corrigir um problema urgente, o sistema se torna um pouco mais frágil, um pouco mais complexo e um passo mais próximo de se tornar uma bola de lama incontrolável.

Diante da crescente complexidade e da pressão por resultados imediatos, a equipe pode tentar "Sweeping it Under the Rug". Esconder o código problemático atrás de interfaces mais apresentáveis, isolar partes disfuncionais sem realmente corrigi-las ou simplesmente ignorar os problemas esperando que eles sumam sozinhos se tornam mecanismos de defesa para lidar com a sensação de caos. No entanto, essa prática apenas mascara os problemas, tornando-os ainda mais difíceis de serem encontrados e corrigidos no futuro.

Eventualmente, o "Big Ball of Mud" atinge um ponto crítico. A cada nova função, o custo e o tempo de desenvolvimento aumentam exponencialmente. A equipe se torna refém do próprio código, incapaz de realizar mudanças significativas sem quebrar algo. A documentação se torna obsoleta e a comunicação entre os membros da equipe, na tentativa de explicar aquele código, se torna um desafio enorme. Nesse ponto, a "Reconstruction" - a decisão radical de jogar fora o sistema antigo e começar do zero - começa a ser considerada como a única saída viável.

A decisão de reconstruir um sistema não é fácil. Envolve custos elevados, prazos desafiadores e a difícil tarefa de convencer os stakeholders de que, às vezes, é preciso recuar para poder avançar de forma sustentável. No entanto, a reconstrução oferece uma oportunidade única de romper com o ciclo vicioso do "Big Ball of Mud", aplicando as lições aprendidas com os erros do passado e construindo um sistema mais robusto, flexível e preparado para receber atualizações e manutenção.

Conclusão

Em um cenário ideal, o “Big Ball of Mud” não deveria acontecer dentro do contexto de desenvolvimento de software e arquiteturas bem estruturadas deveriam ser priorizadas e seguidas à risca para melhorar a qualidade de código e de entregas feitas. Porém, sabemos que o mundo atual pede soluções cada vez mais aceleradas, e

muitas vezes o mercado não cede o tempo necessário para construir um software de qualidade. Muitas vezes os desenvolvedores podem estar cientes de que uma grande bola de lama está sendo formada, mas os stakeholders não tem a visão à longo prazo para entender que o produto demora para ficar pronto, eles priorizam uma solução rápida. Além disso, há também situações em que lidamos com softwares críticos, e principalmente o padrão “Keep it Working” é o mais priorizado nesses casos, pois o sistema não pode ficar inativo.

Diante desses motivos, observa-se que muitas vezes o “Big Ball of Mud” acaba sendo justificado, mesmo que não idealmente. A melhor estratégia para que não seja necessário chegar ao ponto da “Reconstruction” e ter que começar do zero, é criar nos desenvolvedores e stakeholders essa visão de que, a longo prazo, a arquitetura bem estruturada e a consequente demora para produção é uma solução muito melhor do que ter uma aplicação que será temporária e precisará ser refeita. No final, os custos são mais altos se for necessário que o código seja reconstruído.

Levando em conta que em alguns momentos alguns desses padrões são realmente necessários para situações urgentes, outra solução para mitigar os problemas do “Big Ball of Mud” seria o foco em refatoração das dívidas técnicas deixadas por eles. Os problemas, urgências e imprevistos acontecem, mas deve-se saber lidar com eles à longo prazo.

Em conclusão, arquiteturas estruturadas e código limpo são ideais e sempre que possível devem ser priorizados, entretanto em alguns casos soluções mais rápidas são necessárias e os padrões mais sustentáveis são deixados de lado. Nesses casos, é importante estar ciente da bola de lama que pode estar sendo criada e considerar o problema à longo prazo para mitigar os problemas que vêm com ela.