PUC Minas

Autor(a): Isabella Luiza Dias dos Santos

Síntese do Artigo: “The Big Ball of Mud” - Brian Foote e Joseph Yoder

Há algum tempo, uma arquitetura predominantemente inserida nos códigos dos programadores é a chamada “Big Ball Of Mud”. Apesar de existirem diversos tipos de arquitetura, como a Pipeline e a arquitetura em camadas, há uma certa insistência por parte dos desenvolvedores em implementar essa arquitetura considerada desordenada e extensa, que acarreta em diversos problemas para os envolvidos nos desenvolvimentos de softwares.

Visto isto, os autores chegam aos seguintes questionamentos: Por que ela é tão popular e por que a maioria dos programadores insistem em utilizá-lá, sendo que ela prejudica o desenvolvimento do projeto? Para responder esses questionamentos, foram analisadas seis padrões de arquitetura que auxiliarão na melhor compreensão acerca dessas questões. São eles: Big Ball Of Mud, Throwaway Code, Piecemeal Growth, Keep It Working, Sweeping It Under The Rug e Reconstruction.

É muito comum que um programa desordenado acabe surgindo de um código jogável, que é aquele código que foi implementado na intenção de ser usado apenas uma única vez, e então, é descartado. Entretanto, ele acaba ganhando uma certa vida própria. Com isso, uma vez que um problema ocorre, a intenção de alterar essa parte ao invés de alterar todo o projeto de forma adequada, é tida como o melhor passo a ser seguido, considerando a rapidez como determinante para esta decisão. Assim sendo, um simples programa descartável acaba se tornando um “Big Ball Of Mud”.

Ao abordar o Big Ball Of Mud, os autores fazem uma comparação entre as favelas, que são construções que, apesar de possuírem um mal planejamento, cumprem uma necessidade imediata que é a de oferecer moradia com recursos disponíveis. Da mesma forma, um sistema "Big Ball of Mud" é uma arquitetura de software que surge como reflexo das necessidades urgentes de, por exemplo, prazos curtos e orçamentos limitados.

Os autores citam que o "Big Ball of Mud" é, na verdade, um padrão de arquitetura dominante porque é o caminho mais fácil de ser seguido para entregar um produto de maneira rápida. O risco se manifesta em esquemas que nunca são descartados, consolidando o sistema como uma "bola de lama" permanente.

Ao debater sobre o Throwaway Code, os autores o comparam a uma construção temporária, cuja intenção é a de ser substituída por algo permanente, mas que, por diversas razões, acaba nunca sendo removida e se mantêm.

No desenvolvimento de software, isso ocorre como um código escrito rapidamente para resolver um problema de imediato, com o intuito de ser reescrito de forma correta posteriormente. A principal razão por trás dessa prática é a falta de tempo e a urgência por resultados rápidos. Se essa prática tiver êxito, o impulso de continuar usando é muito grande, e acaba se tornando fixo ao sistema.

Ao dissertar sobre Piecemeal Growth, os autores fazem uma comparação acerca da maneira que a maioria das cidades crescem, de forma incremental, adicionando estruturas conforme a necessidade vai surgindo. Na questão do software, esse padrão representa a evolução constante de um sistema através de pequenas modificações. O risco em relação a esse desenvolvimento frequente é que este é o caminho mais comum para a criação de um "Big Ball of Mud", já que cada nova correção pode desgastar a arquitetura geral do programa. Para evitar esse cenário, é preciso estabelecer um equilíbrio entre a refatoração e a consolidação.

Ao abordar o Keep It Working, é feita uma comparação a respeito da infraestrutura de uma cidade, como o sistema de esgoto, cuja prioridade não é fazer grandes alterações, mas garantir que continuem operacionais sob qualquer circunstância. Na questão do software, quando empresas se tornam muito dependentes de sistemas para seus movimentos, havendo a necessidade de mudanças pesadas e, necessária paralisação, os sistemas poderão ficar prejudicados, uma vez que haverá uma maior dificuldade na identificação do causador do problema. Logo, é fundamental a adoção da abordagem de incrementos menores para que o problema seja facilmente localizado, e assim, não danificar o sistema por inteiro.

Ao discutir sobre o Sweeping It Under The Rug, é apresentado como exemplo o sarcófago de concreto construído sobre o reator de Chernobyl. O propósito do sarcófago foi isolar o problema das radiações e esconder o caos que foi instalado para que não contaminasse o restante do ambiente. No cenário de software, esse padrão é aplicado a partes do sistema que se tornaram um "código espaguete" perigoso de ser alterado. A estratégia aqui é isolar esses elementos críticos do resto do sistema, criando bloqueios ao redor deles. Assim, um problema de grande dimensão é transformado em algo controlável.

Ao abordar sobre o Reconstruction, é usado uma analogia da demolição de um antigo estádio como forma de explicar a necessidade de recomeçar algo do zero, quando a arquitetura original se torna um impasse para novas exigências. Na questão de software, isso ocorre quando um código se torna irreparável, ou quando não consegue se adaptar às necessidades futuras. Entender detalhadamente as falhas do sistema anterior é fundamental para orientar o desenvolvimento do novo programa. Contudo, para evitar a construção de uma arquitetura exageradamente aperfeiçoada, é necessário manter uma postura crítica e cuidadosa ao longo do processo.

Por fim, é constatado que é fundamental a compreensão, por parte das equipes de desenvolvedores, acerca do reconhecimento de tudo o que pode levar ao enfraquecimento da arquitetura do software, para que assim, possam saber como combatê-los. Como discutido pelos autores, sistemas duráveis exigem aprendizado constante sobre arquitetura conforme o projeto evolui.