The Big Ball of Mud - Resenha

Nome: Arthur Henrique T e S Bacelete
Professor: João Paulo Aramuni

De início o autor nos introduz um termo relativamente cômico: o padrão “BIG BALL OF MUD”. O padrão (embora perverso) do “BIG BALL OF MUD” representa o quão ruim um software é ou pode se tornar a partir de alguns preceitos que não são seguidos pelos programadores. Esse efeito geralmente é sentido quando o software cresce de maneira desregulada, repetitiva, ou quando a estrutura (ou arquitetura) não está bem definida (ou os requisitos não estão bem entendidos), etc. E apesar de apresentar efeitos adversos, é um padrão que é recorrentemente utilizado no desenvolvimento de diversos sistemas. O autor aborda seis padrões comumente identificados em desenvolvimento de sistemas, são eles:

BIG BALL OF MUD
THROWAWAY CODE
PIECEMEAL GROWTH
KEEP IT WORKING
SWEEPING IT UNDER THE RUG
RECONSTRUCTION.

Segundo o autor, isso são padrões que buscam representar a lacuna entre o que pregamos e o que praticamos. O KEEP IT WORK, por exemplo, é abordado inicialmente pelo autor a partir de uma analogia de um ataque a uma cidade. Imagine que uma cidade foi atacada em uma determinada parte, um prédio ou um quarteirão. A tendência natural é que a cidade funcione normalmente, caso contrário, a governança da cidade está sendo feita de modo ruim e mal sucedido. E isso vai diretamente na ideia de modularidade e no sentido da redução de acoplamento de funções em um código, por exemplo. Adiante o autor também aborda conceitos fundamentais (ou forças globais) que podem levar a ruína, ou à glória de um sistema: **tempo, custo, experiência, habilidades, complexidade e mudança.**

Uma outra analogia interessante sobre como um software pode se tornar um caso como uma grande bola de lama (“BIG BALL OF MUD”) é o caso das favelas. As favelas são construídas com poucos recursos, de maneira desorganizada e por trabalhadores, em sua maioria, desqualificados. Portanto, ele nos recomenda: foque primeiro nos recursos e funcionalidades, depois direcione-os para arquitetura e performance. E, adiante, ao abordar o padrão “PIECEMEAL GROWTH”, ele também cita que um software deve **comportar mudanças, mesmo mudanças que vão contra a essência original do projeto** e o autor cita que o software não deve, portanto, ter uma arquitetura rígida, equivocada e fora de moda, pois os usuários precisam de mudança com o tempo. E, de novo, de maneira bastante elegante ele compara os planejamentos não abordados nos padrões arquitetônicos da França e Brasília com os de Houston, por exemplo, onde a cidade se tornou atrativa para o público. Como o autor mesmo diz:

“Therefore, incrementally address forces that encourage **change** and **growth**. Allow opportunities for growth to be exploited **locally**, as they occur.”

E conclui dizendo que o crescimento fracionário, ou PIECEMEAL GROWTH, pode ser uma oportunidade incrível para o sistema vigente, através das mudanças que são quebradas em pequenos pedaços gerenciáveis (“small, manageable chunks”). O autor segue uma linha de raciocínio bastante didática, como aconteceu também com a seção do padrão KEEP IT WORKING, que ele basicamente aborda as dificuldades enfrentadas no desenvolvimento de um software através de uma ampla gama de modificações com o tempo. Para isso ele diz: **faça o que for preciso para manter o software funcionando, continue trabalhando**. Ou seja, em vez de realizar grandes reformulações de uma só vez, aplique mudanças graduais, pois reformular tudo de uma vez costuma gerar riscos e dificultar a identificação de problemas. No padrão SWEEPING IT UNDER THE RUG ele nos diz: limpe a bagunça, e se você não consegue, pelo menos a isole (ou “esconda” debaixo do tapete). O que ele quer dizer é que desentendimentos no código são difíceis de reparar, ou estender e tendem a crescer de forma descontrolada, se você não reparar. Ele conclui a abordagem dos padrões com o padrão RECONSTRUCTION, e nos cita, se seu código declinou a um ponto que é trabalhoso demais de entender ou reparar, talvez seja a hora de reconstruí-lo novamente.

De maneira geral, o autor conseguiu abordar de forma muito clara e didática boas práticas (e as ruins também) que o software pode apresentar durante o seu desenvolvimento e como podemos lidar com elas. Eu particularmente gostei bastante do texto e a forma que ele abordou tudo.