No Silver Bullet - Resenha

Nome: Arthur Henrique T e S Bacelete

De início, Frederick P. Brooks, Jr. nos apresenta a famosa metáfora da "bala de prata" para ilustrar a busca incessante por uma única inovação tecnológica ou de gestão que possa, de forma mágica, resolver os problemas de produtividade, confiabilidade e simplicidade no desenvolvimento de software. O autor, no entanto, argumenta de forma cética, mas não pessimista, que tal solução milagrosa não existe e que a própria natureza do software torna improvável que venha a existir.

Para embasar seu argumento, Brooks divide as dificuldades da engenharia de software em duas categorias:

essência e **acidentes**. A essência representa as dificuldades inerentes à natureza do software, enquanto os acidentes são os problemas que acompanham sua produção, mas não são intrínsecos a ela. Ele aponta quatro propriedades essenciais do software que o tornam inerentemente difícil:

- **Complexidade:** O software é mais complexo do que qualquer outra construção humana, pois não há partes que se repetem. A complexidade aumenta de forma não linear com o tamanho do sistema, o que leva a dificuldades de comunicação, atrasos e falhas.
- **Conformidade:** O software deve se conformar a instituições e sistemas humanos que são arbitrários e complexos, o que não pode ser simplificado apenas com o redesenho do software.
- **Mutabilidade:** O software de sucesso está constantemente sujeito a pressões por mudança, seja por novas aplicações, usuários ou tecnologias.
- **Invisibilidade:** O software não possui uma representação espacial, o que o torna difícil de visualizar e entender, dificultando o processo de design e a comunicação entre as mentes.

O autor argumenta que os grandes avanços do passado, como as linguagens de alto nível, o tempo compartilhado e os ambientes de programação unificados, atacaram dificuldades acidentais, e não essenciais. Por isso, embora tenham trazido ganhos significativos, o retorno de novas ferramentas que seguem a mesma linha será cada vez menor. Brooks analisa então as "esperanças de prata" da época, como Ada, programação orientada a objetos, inteligência artificial e programação gráfica, e conclui que nenhuma delas seria a solução definitiva, pois não resolvem a complexidade essencial do software.

Apesar da ausência de uma "bala de prata", o autor propõe alguns ataques promissores à essência do problema do software:

- **Comprar em vez de construir:** A solução mais radical é não construir software, mas sim comprá-lo pronto, aproveitando o mercado de massa que dilui os custos de desenvolvimento.

- **Refinamento de requisitos e prototipagem rápida:** A parte mais difícil é decidir o que construir. Por isso, é fundamental um processo iterativo de extração e refinamento de requisitos com o cliente, usando protótipos para tornar a estrutura conceitual real e testável.
- **Desenvolvimento incremental:** Em vez de construir software como um edifício, devemos "cultivá-lo" de forma orgânica. O sistema deve ser desenvolvido de forma incremental, começando com um esqueleto funcional e adicionando funcionalidades aos poucos, o que permite ter sempre um sistema funcionando e facilita o backtracking.
- **Grandes designers:** A melhoria da arte do software está centrada nas pessoas. Grandes projetos vêm de grandes designers, e as organizações devem se esforçar para identificar, cultivar e recompensar esses talentos da mesma forma que fazem com os grandes gestores.

De maneira geral, Brooks nos oferece uma visão sóbria e realista sobre a engenharia de software. Ele desmistifica a crença em soluções mágicas e nos direciona para o que realmente importa: focar na complexidade conceitual do software e investir em processos e pessoas que possam lidar com essa complexidade de forma eficaz. A leitura é essencial para qualquer profissional da área, pois nos lembra que o progresso é feito de forma gradual e com muito esforço, e não através de atalhos.