Resumo do artigo
Is Design Dead? - Martin Fowler

Sungwon Yoon
Nro USP 9822261

Design Planejado e Evolucionário

O autor chama de design planejado a abordagem tradicional de criar software que dedica tempo suficiente para projetar software no início do projeto, enquanto que chama de design evolucionário que vai surgindo à medida que o projeto avança, como na abordagem XP, sem uma fase formal de planejamento, mas ainda mantendo alguma estrutura.

Uma crítica ao design evolucionário no desenvolvimento ágil é que ele é visto como um conjunto de soluções temporárias. Fowler explica que, nesse contexto, a ideia é "escrever o código primeiro e resolver os problemas depois", o que pode levar a complicações à medida que o tempo passa, tornando mais difícil adicionar novas funcionalidades e aumentando os custos de correção de erros.

Por outro lado, isso não significa que o planejamento de design seja isento de problemas. Muitas vezes, no planejamento de design, há uma separação entre os designers e os programadores, o que pode resultar em problemas que os designers não conseguem prever os problemas que ocorrerão ao decorrer do desenvolvimento. Além disso, Fowler argumenta que há constante mudança de requisitos, o que pode tornar os designs planejados menos flexíveis e, no final, levar à adoção da abordagem "codificar primeiro e corrigir os problemas depois".

As Práticas para Utilização do XP

Para o autor, a solução para esses problemas está nas práticas usadas no XP (Extreme Programming). Isso inclui Test-Driven Development (TDD), que envolve criar testes antes de implementar os requisitos e garantir que esses testes assegurem o funcionamento do código existente, e a integração contínua, onde a integração das funcionalidades desenvolvidas por diferentes equipes deve ocorrer regularmente. Ele também destaca os benefícios significativos da prática de refatoração no XP, afirmando que a refatoração no contexto do XP é mais eficaz em comparação com a reformulação desorganizada.

Simplicidade

O autor destaca a importância da simplicidade, ou seja, implementar apenas o que é necessário na fase atual. Em relação aos padrões de design, aconselha o seguinte:

- Invista tempo aprendendo sobre patterns
- Concentre-se em quando aplicar o pattern (não muito cedo)
- Concentre-se em como implementar o pattern na sua forma mais simples primeiro, depois adicione complexidade
- Se você colocar um pattern, e depois perceber que ele não está dando conta - não tenha medo de retirá-lo

Desenvolvendo uma Arquitetura

O autor define a arquitetura de software como uma ideia de elementos que formam o núcleo do sistema, as peças que são difíceis de alterar. Ele reconhece a importância da arquitetura de software inicial, mas enfatiza que não deve ser vista como definitiva e que deve ser flexível para mudanças quando problemas são identificados (exemplo: EJB).

UML

Segundo o autor, documentação tende a ser vista de forma negativa e complexa pelos adeptos de XP, por isso, o uso do UML também foi desvalorizado. Nesse contexto, dá orientações para a boa utilização de diagramas.

- O valor primordial é a comunicação, então eles devem ser elaborados com o leitor em mente, enfatizando a inclusão do que é importante e a exclusão do que não é

- Em projetos mais complicados, é bom fazer sessões rápidas de design com a equipe, sendo curtas e abordando apenas os aspectos essenciais e produzindo rascunhos, não designs finais

- O design pode haver erros e ajustes necessários durante a implementação, e essas mudanças no design não exigem necessariamente que os diagramas sejam atualizados. Se o diagrama ajudou a compreender o design, ele já cumpriu seu papel, mesmo que seja descartado posteriormente

- Apesar dos diagramas, o código é a fonte mais detalhada de informações

Você Quer ser um Arquiteto Quando Crescer?

O autor destaca a importância de assumir um papel de liderança técnica. Ele argumenta que, quando você se torna um arquiteto, pode parecer afastado do desenvolvimento e não envolvido em codificação. Ele acredita que desenvolvedores experientes devem aprimorar suas habilidades e desempenhar um papel ativo na equipe, fornecendo orientação e ensinando, em vez de se afastarem do desenvolvimento como alguém distante.

Reversibilidade

O autor discute a importância da reversibilidade tanto na metodologia ágil quanto na arquitetura. Em outras palavras, do ponto de vista do design evolutivo, as decisões que podem ser facilmente desfeitas são menos importantes do que aquelas que são irreversíveis. Também menciona que é significativo experimentar quão difícil será fazer mudanças futuras como parte dessa avaliação.

Design está Acontecendo?

O autor argumenta que é difícil avaliar se o design evolutivo está ocorrendo ou não. Embora tenham sido propostas medidas quantitativas e objetivas para a estrutura do programa, essa área deve ser subjetiva e qualitativa. Também afirma que pessoas que não são desenvolvedores teriam dificuldade em compreender esse aspecto. Assim, além de todas as métricas propostas, as opiniões de desenvolvedores são relevantes.