Documentação de software

O Software é uma parte vital da vida de todo humano vivendo atualmente. Além dos celulares e computadores que existem para usar softwares diferentes. Todo carro moderno usa software, seu mercado usa software, o governo usa software para gerenciar sua vida, e seu banco usa para gerenciar seu dinheiro.

Em cada um desses casos uma falha pode ser catastrófica, causando estresse, perdendo dinheiro ou no pior dos casos levando a morte de uma pessoa. Em algo tão importante falhas são inaceitáveis, e todo software tem que ser visto com a mentalidade de que ele está gerenciando algo importante e precisa ser construído para minimizar as chances de falhas.

Por isso com a passagem do tempo e com os softwares se tornando mais e mais importantes técnicas que diminuem os erros como testes automatizados, arquitetura de software e código limpo se tornam mais comuns, até o dia atual em que um programa construído sem essas técnicas é considerado não profissional e perigoso pela comunidade.

Entre essas técnicas arquitetura de software vem como a ideia de criar um documento que descreve o software que pode ser usado para verificar o que é necessário para sua construção e como ele será quando terminado. Isso em uma questão de arquitetura de um prédio, por exemplo, pode permitir ao arquiteto ver o custo de materiais, a mão de obra e maquinário necessário, as técnicas utilizadas, e ao ver tudo isso ele pode tomar decisões e mudar o prédio antes de sua construção começar visto que mudanças no predio muito mais difíceis de fazer quando ele está em construção e quase impossíveis quando ele está terminado.

Para ter ideia de como isso funcionaria, vamos dizer que você conversa com a sua tia para desenvolver um software para controlar a venda de lanchonete que ela é dona, ela te diz que quer ter um site com um menu e as informações da loja, e um programa que controle as vendas dos lanches.

Você sem mais nenhuma comunicação com ela passa uma semana e faz do zero o que foi pedido, ao entregar isso para sua tia você descobre que ela tem dificuldade em utilizar o software de controle de estoque, tem uma parte de cadastro de clientes que ela não utiliza, e o computador que ela utiliza é um windows, você fez o programa em um mac e ao tentar correr o software no computador dela ele não inicia. Esses problemas são tão grandes e para consertar precisa mudar

O problema número um que acontece são falhas de comunicação entre o cliente e programador, resultando em um software que não faz o que é necessário e precisando de refatorações grandes. Para resolver essa função a documentação pode além de ser um documento legal serve como uma ponte de comunicação entre eles permitindo o cliente expressar o que ele precisa e o programador ser capaz de entender os requisitos para gerar um software.

Como o custo de uma mudança aumenta quanto mais for construído isso significa que é importante ter uma idéia do que o software necessita no início de sua construção e sempre checar com o cliente para verificar mudanças necessárias o mais cedo possível. Isso pode

levar à conclusão de que gerar um documento de software que encompassa tudo e seguir ele religiosamente geraria um software sem erros.

Mas na prática é visto que mudanças de plano são constantes mesmo com uma documentação extensa, por isso nas ideologias ágeis eles se focam em projetos que se comunicam constantemente com o cliente, entregando partes do programa final para ele e verificando os próximos passos e gerando a documentação conforme o desenvolvimento do projeto. Essa forma de documentação menos rígida em conjunto com a ideologia ágil é a metodologia de construção mais utilizada no mercado atualmente.

Para isso nos anos 80 e 90 começaram a ser criados padrões de software e documentação para gerenciar o processo de criação de um software, a UML foi criada para ser um agregado de múltiplas técnicas que poderia ser utilizado na grande maioria de aplicações, com um foco em orientação a objeto

Para isso com a evolução da criação de software nos anos 80 e 90 foram criadas várias maneiras de modelar um software gerando uma documentação que poderia ser utilizada para gerar um produto que melhor segue o que o cliente quer

Relatório 16/06:

Feito a introdução e mais alguns parágrafos de partes diferentes

Utilizado:

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/#:~:text=UML%2C%20short%20for%20Unified%20Modeling,business%20modeling%20and%20other%20non%2D

https://en.wikipedia.org/wiki/Unified_Modeling_Language

Relatório 23/06:

Feito um pouco mais sobre arquitetura e documentação

Utilizado:

https://en.wikipedia.org/wiki/Agile_software_development

Atividades da equipe:

Eu to solo