Cartilha :

Engenharia de Software

Evolução Da Engenharia de Software

Participantes

FABIO RIBEIRO-RA: N062290
GUSTAVO CAMPUS-RA: G050GD3
JUSTING VINICIUS-RA: R015440
KAUA KOKI-RA: G819675
PEDRO GORIN-RA: N087730
THIANES PIERRE-RA: N307EDO
PEDRO COSTA-RA: G7492H1

JOÃO VITOR RODRIGUES-RA: G72CHG5
PAULO VICTOR-RA: N0753DO
EDUARDO BORGES-RA: G804090
LUCAS SANTANA-RA: F3562I4
MAURO HENRIQUE-RA: R023893
CAMILA THOMAS-RA: G87ADC2
BEATRIZ OLIVEIRA-RA: N296429
GABRIEL DOS SANTOS-RA: G86DAB6

O que Engenharia de Software ?

A Engenharia de Software é uma prática de aplicar princípios de engenharia ao desenvolvimento de software. Envolve a criação de sistemas e aplicações que atendem a requisitos funcionais e não funcionais, como segurança, desempenho e manutenibilidade

Com a crescente dependência de software em praticamente todos os aspectos da vida moderna, a Engenharia de Software se tornou uma disciplina crítica.

Origem e Criação

Quem criou:

O termo "Engenharia de Software" foi lançado pela primeira vez por Margaret Hamilton, que liderou o desenvolvimento do software para o projeto Apollo da NASA

Por que foi criado :

Na década de 1960, os projetos de software eram frequentemente desorganizados, sem uma metodologia clara. O desenvolvimento de software enfrentou problemas importantes, como cronogramas não cumpridos e orçamentos estourados. A ideia era formalizar e profissionalizar o desenvolvimento de software para melhorar a qualidade e a confiabilidade dos produtos.

Benefícios da Engenharia de Software

Benefícios

- **Maior Controle de Projetos:** A utilização de modelos e metodologias permite um controle mais rigoroso sobre prazos e custos.

- **Satisfação do Cliente:** Com a entrega contínua e a adaptação às mudanças, os clientes têm mais oportunidades de influência no produto final, resultando em maior satisfação.

- **Inovação**: A Engenharia de Software permite a inovação contínua, com a rápida implementação de novas funcionalidades e correções.

Efeitos Causados

- **Melhoria da Qualidade do Software:** A aplicação de práticas de Engenharia de Software resultou em um aumento significativo na qualidade do software, falhas e bugs.

- **Eficiência em Processos:** A sistematização das etapas de desenvolvimento levou a uma maior eficiência, diminuindo o ritmo e o custo dos projetos.

- **Aumento da Colaboração:** Metodologias ágeis promoveram a colaboração entre equipes de desenvolvimento e partes interessadas, melhorando a comunicação e o entendimento dos requisitos.

Desenvolvimento ao Longo dos Anos

Anos 1970: Surgimento de modelos de desenvolvimento como o modelo em cascata, que enfatizava a sequência de etapas: análise, design, implementação, testes e manutenção.

Anos 1980: O foco se expandiu para incluir métodos ágeis e abordagens voltadas a objetos, promovendo maior flexibilidade e colaboração.

Anos 1990: A introdução da Programação Extreme (XP) e do Scrum trouxe novas práticas para gerenciamento de projetos e desenvolvimento

Anos 2000 até hoje: A Engenharia de Software evoluiu para incluir DevOps, integração contínua e entrega contínua, além de uma maior ênfase em segurança e experiência do usuário.

Empresas que Utilizam Engenharia de Software

Utilize práticas ágeis em seus projetos de desenvolvimento de software, como a criação de aplicativos e serviços em nuvem. A cultura de inovação da empresa baseia-se em princípios de Engenharia de Software.

Com a introdução do Azure DevOps, a Microsoft promoveu práticas de integração contínua e entrega contínua, além de adotar metodologias ágeis em sua linha de produtos.

O modelo de negócios da Amazon é construído em torno da agilidade e inovação, utilizando práticas de DevOps para gerenciar sua vasta infraestrutura em nuvem e garantir a escalabilidade e confiabilidade dos serviços.

A empresa usa desenvolvimento ágil e integração contínua para gerenciar seu software, permitindo atualizações rápidas e melhorias contínuas em produtos.

Muitas startups adotam práticas de Engenharia de Software desde o início, permitindo uma rápida iteração e ajuste de suas ofertas em resposta ao feedback dos clientes.

Conhecida por sua arquitetura de microserviços, a Netflix utiliza práticas de DevOps para garantir a entrega contínua de seu serviço de streaming. Isso permite que uma empresa implemente novas funcionalidades rapidamente e mantenha uma experiência de usuário consistente.

Problemas no Processo de Desenvolvimento de Software

Crise do Software (Anos 1960-1970):

- **Descrição:** Durante as primeiras fases de desenvolvimento de software, muitos projetos ficaram atrasados, orçamentos foram estourados e a qualidade do software era insatisfatória. Isso resultou em sistemas falhos que não atendiam às necessidades dos usuários.

- **Exemplo:** O sistema de controle do projeto Apollo da NASA teve sérios problemas no início. Embora o software tenha sido bem sucedido a longo prazo, os desafios iniciais levaram a um foco maior em Engenharia de Software.

Mudanças de Requisitos:

- **Descrição:** Mudanças frequentes nos requisitos do cliente durante o desenvolvimento podem levar a atrasos e retrabalhos, tornando o projeto mais complexo e caro.

- **Exemplo:** Muitas equipes de desenvolvimento enfrentam dificuldades ao tentar implementar alterações solicitadas pelos clientes nas etapas finais do processo, resultando em frustrações e produtos que não atendem às expectativas.

Problemas de Escalabilidade:

- **Descrição:** À medida que um software cresce, pode se tornar difícil escalá-lo sem comprometer o desempenho ou a arquitetura.

- **Exemplo:** A Netflix, em sua evolução inicial, desafiou a escalabilidade de seu sistema, resultando em interrupções de serviço e problemas de desempenho.

Falta de Documentação:

- **Descrição:** A falta de comunicação entre equipes, stakeholders e clientes pode levar a mal-entendidos sobre requisitos e expectativas, resultando em produtos que não atendem às necessidades reais dos usuários.

- **Exemplo:** Projetos grandes frequentemente enfrentam problemas de comunicação entre desenvolvedores, designers e gerentes de produto, levando a discrepâncias entre a visão do projeto e a implementação.

Falta de Documentação:

- **Descrição:** A falta de uma abordagem rigorosa para testes pode resultar em software cheio de bugs e vulnerabilidades, comprometendo a experiência do usuário e a segurança do sistema.

- **Exemplo:** Muitas empresas, incluindo gigantes como Microsoft, enfrentaram problemas com bugs em lançamentos de produtos, levando a correções emergenciais e perda de confiança do cliente.

Sobrecarga de Trabalho:

- **Descrição:** Em ambientes de alta pressão, como startups e empresas de tecnologia, as equipes podem ser sobrecarregadas, resultando em burnout e redução da qualidade do trabalho.

- **Exemplo:** Empresas como Uber e Airbnb, durante seus crescimentos explosivos, enfrentaram críticas sobre a carga de trabalho em suas equipes de engenharia, o que afetou a retenção de talentos.

Segurança e Vulnerabilidades:

- **Descrição:** A complexidade crescente do software torna-o vulnerável a ataques cibernéticos. Falhas de segurança podem levar a vazamentos de dados e comprometer a confiança do usuário.

- **Exemplo:** Empresas como Equifax e Target sofreram evidências de segurança graves que resultaram na perda de dados de milhões de clientes, destacando a importância de práticas robustas de segurança no desenvolvimento de software.

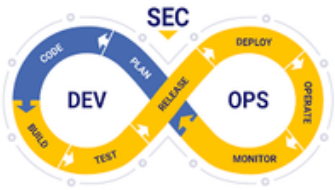
Futuro da Engenharia de Software

Automação Avançada com Inteligência Artificial (IA)

- **Impacto:** A Inteligência Artificial está começando a automatizar várias partes do processo de desenvolvimento de software, como geração de código, testes e detecção de bugs.

- **Ferramentas de IA:** Ferramentas como GitHub Copilot e outras assistentes de codificação estão ajudando desenvolvedores a escrever código mais rapidamente e com menos erros, sugerindo linhas de código baseadas em padrões e aprendizado contínuo.

- **Teste Automatizado:** A IA será usada para prever falhas e otimizar testes de software, diminuindo o tempo necessário para garantir a qualidade dos sistemas.



DevOps e DevSecOps com Foco em SegurançaArtificial (IA)

- **DevSecOps:** A integração de segurança ao ciclo de desenvolvimento desde o início, conhecida como DevSecOps, está se tornando cada vez mais importante. Com o aumento de ataques cibernéticos, a segurança deve ser uma prioridade desde a fase de codificação.

- **Automação de Segurança:** Ferramentas que automatizam a detecção de vulnerabilidades de segurança durante o desenvolvimento serão fundamentais, permitindo que os desenvolvedores abordem problemas de segurança em tempo real.

Aviso: A segurança precisa ser integrada nas fases iniciais do desenvolvimento, e não apenas no final. A prática de "Shift Left Security" significa que as equipes devem incluir verificações de segurança desde a fase de design e codificação, ao invés de deixá-las para depois. Ignorar isso pode resultar em vulnerabilidades críticas que são mais difíceis e caras de corrigir mais tarde no processo.

Aviso: Embora a automação seja uma peça-chave no DevSecOps, ela não deve substituir a criação de políticas de segurança personalizadas para as necessidades específicas do sistema. Ferramentas automáticas podem detectar vulnerabilidades comuns, mas as especificidades de cada projeto, como requisitos de compliance e regulamentações setoriais, exigem atenção humana.

Desenvolvimento Low-Code e No-Code

- **Expansão do Acesso:** Plataformas de low-code e no-code estão democratizando o desenvolvimento de software, permitindo que pessoas sem conhecimento avançado de programação criem aplicativos e sistemas.

- **Aumento da Produtividade:** Isso permitirá que as empresas lancem rapidamente novos produtos e serviços, com menor dependência de engenheiros de software altamente especializados.

- **Impacto para Profissionais de Engenharia:** Embora o desenvolvimento low-code reduza a necessidade de escrever código em certos cenários, engenheiros de software continuarão sendo essenciais para projetos complexos, arquiteturas robustas e otimização de soluções.

Bibliografia

Pressman, Roger S., & Maxim, Bruce R. (2014). Engenharia de Software: Uma Abordagem Profissional. 8ª ed. McGraw-Hill Brasil.

Bass, Len, Clements, Paul, & Kazman, Rick. (2012). Software Architecture in Practice. 3ª ed. Addison-Wesley Professional.

Humble, Jez, & Farley, David. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional.

Leffingwell, Dean. (2016). Scaled Agile Framework for Lean Enterprises: SAFe 4.0 Reference Guide. Addison-Wesley Professional.

Spinellis, Diomidis. (2006). Code Reading: The Open Source Perspective. Addison-Wesley Professional.