UNIVERSIDADE FEDERAL DO MATO GROSSO DO SUL

UFMS

SISTEMAS DE INFORMAÇÃO

O que é preciso para desenvolver um software

Victor Hugo Aquino Agustinho

Ponta Porã – MS

2024

O que é preciso para desenvolver um software?

Desenvolver um software hoje em dia abrange várias etapas e requer uma combinação de conhecimentos, habilidades e recursos. Além disso, é importante ter uma equipe com diversas habilidades, como programadores, designers, gerentes de projeto e profissionais de testes. Algumas ferramentas e tecnologias também podem ser necessárias, como sistemas de controle de versão (por exemplo, Git) e ambientes de desenvolvimento integrado (IDEs).

Aqui estão os principais elementos envolvidos no processo:

1. **Definição do Problema**: Identificar claramente o problema que o software deve resolver e quais são os requisitos dos usuários.

2. **Planejamento**: Criar um plano de desenvolvimento que inclua metas, cronogramas e recursos necessários. É importante decidir sobre a metodologia a ser usada (Agile, Waterfall, etc.).

3. **Análise de Requisitos**: Coletar e documentar os requisitos funcionais e não funcionais. Isso pode incluir entrevistas com usuários, pesquisas e análise de mercado.

4. **Design**: Desenvolver a arquitetura do software, incluindo a estrutura de dados e a interface do usuário. Diagramas como UML podem ser úteis neste estágio.

5. **Desenvolvimento**: Escrever o código usando as linguagens de programação adequadas (Java, Python, C#, etc.). É aqui que a implementação real do software ocorre.

6. **Testes**: Realizar testes para garantir que o software funcione corretamente e atenda aos requisitos. Isso inclui testes unitários, testes de integração e testes de aceitação pelo usuário.

7. **Implementação**: Colocar o software em produção, o que pode envolver a configuração de servidores, instalação em dispositivos ou integração com outros sistemas.

8. **Manutenção**: Após o lançamento, o software precisará de atualizações, correção de bugs e melhorias com base no feedback dos usuários.

9. **Documentação**: Criar documentação técnica e de usuário, que será essencial para a manutenção do software e para ajudar os usuários a entenderem como utilizá-lo.

Gerência de Configuração de Software (SCM)

O SCM (Software Configuration Management, ou Gerenciamento de Configuração de Software) é uma prática fundamental na área de qualidade de software, pois envolve a identificação, controle e rastreamento de alterações em artefatos de software.

O que é SCM?

O SCM é um conjunto de processos e ferramentas que suportam a gestão da configuração de software, permitindo que as equipes controlarem as versões de softwares e seus componentes, documentações, e requisitos. O SCM envolve um conjunto de práticas e ferramentas que ajudam equipes a identificar, organizar e gerir as modificações e versões de software, documentação, requisitos e outros artefatos. É um componente essencial para manter a integridade e qualidade do produto de software, especialmente em ambientes colaborativos e de alta complexidade.

Qual a importância do SCM na Qualidade de Software?

- Controle de Versões: Ajuda a manter o controle sobre diferentes versões do software, permitindo rastrear alterações e revertê-las quando necessário.
- Auditabilidade: Proporciona um histórico completo das alterações, facilitando auditorias e conformidade com regulamentos.
- Aumento da Colaboração: Permite que equipes variadas trabalhem juntas sem conflitos, já que as mudanças podem ser integradas de forma eficiente.
- Gerenciamento de Mudanças: Facilita a implementação de mudanças de forma controlada, o que melhora a qualidade final do produto.

Quais os passos do processo de SCM?

- Identificação da Configuração: Definição de artefatos que precisam ser gerenciados.
- Controle de Mudanças: Processo formal para solicitar, avaliar e aprovar mudanças.
- Registro de Status: Monitoramento e apresentação do status dos itens de configuração.
- Auditorias de Configuração: Verificações realizadas para assegurar que o sistema corresponde à configuração documentada.

Quais ferramentas de SCM hoje no mercado?

- Ferramentas como Git, SVN, Mercurial e ferramentas de integração contínua (CI) como Jenkins e Travis CI são comumente usadas para facilitar o SCM.

Quais os desafios do SCM atualmente?

- Complexidade: À medida que os projetos crescem, gerenciar a configuração de forma eficiente torna-se um desafio.

- Treinamento: A equipe precisa de treinamento para utilizar ferramentas de SCM de forma eficaz.
- Cultura de Mudança: Implementar práticas de SCM pode exigir uma mudança na cultura da equipe.
- Integração de Processos Ágeis: Adaptar práticas de SCM a ambientes ágeis pode ser um desafio, já que a agilidade requer respostas rápidas a mudanças.

Quais as práticas recomendadas?
- Estabelecer uma política clara de gerenciamento de configuração.
- Automatizar processos sempre que possível.
- Manter uma documentação detalhada e acessível.

Então entendemos que ele é uma disciplina que controla e notifica as inúmeras correções, extensões e adaptações aplicadas durante o ciclo de vida do software de forma a assegurar um processo de desenvolvimento e evolução sistemático e rastreável, sendo indispensável quando equipes manipulam, muitas vezes em conjunto, artefatos comuns.

Existi um forte apelo para o uso da Gerência de Configuração de Software durante a etapa de manutenção, a sua aplicação não se restringe somente a essa etapa do ciclo de vida do software. O uso dos sistemas de Gerência de Configuração é fundamental para prover controle sobre os artefatos produzidos e modificados por diferentes recursos desde o planejamento e levantamento de requisitos até a construção e entrega do produto. O motivo da sua importância está geralmente associado aos problemas identificados quando a Gerência de Configuração não é utilizada no desenvolvimento de software.

Gerência de Configuração de Software abrange três sistemas principais: controle de modificações, controle de versões e controle de gerenciamento de construção.

O sistema de controle de versões permite que os artefatos sob Gerência de Configuração evoluam de forma distribuída, concorrente e disciplinada, evitando perdas ou sobreposições durante o desenvolvimento e a manutenção do artefato. Podemos citar como exemplos de ferramentas de mercado: CVS, Subversion, IBM Rational ClearCase e Microsoft Visual Source Safe.

O sistema de controle de modificações armazena todas as informações geradas durante o andamento das solicitações de modificação e relata essas informações aos participantes interessados e autorizados. Podemos citar como exemplos de ferramentas de mercado: Bugzilla, Jira, Trac e IBM Rational ClearQuest.

O sistema de gerenciamento de construção automatiza o processo de transformação dos diversos artefatos do software que compõem um projeto em um sistema executável propriamente dito. Este processo é nomeado construção

do software que, por exemplo, testa e empacota a aplicação java como um arquivo jar. Este processo ocorre de forma aderente às normas, procedimentos, políticas e padrões definidos para o projeto. Podemos citar como exemplos de ferramentas de mercado: Maven e Apache Ant.

As vantagens da utilização da Gerência de Configuração de Software são inúmeras. Dentre elas, podemos listar: ganho de produtividade e eficiência, diminuição do retrabalho e dos erros, aumento da disciplina no processo de desenvolvimento, aumento da memória organizacional, acesso às informações qualitativas e quantitativas referentes ao processo de desenvolvimento, possibilidade de estabelecer uma trilha de auditoria indicando por que, quando e por quem um artefato foi alterado; auxílio à gerência de projetos e garantia de ambiente estável no qual o produto deve ser desenvolvido.

Principais Componentes do SCM

Identificação da Configuração:

- Determina quais itens de software (código, documentação, bibliotecas, etc.) precisam ser gerenciados.
- Define a estrutura e a nomenclatura dos artefatos para facilitar o rastreamento.

Controle de Mudanças:

- Processos formais para solicitar, revisar e aprovar modificações no software e sua configuração.
- Envolve o gerenciamento de solicitações de mudança (change requests) e o impacto associado.

Rastreamento de Status:

- Mantém um registro detalhado das versões e alterações realizadas em cada item de configuração.
- Permite que as equipes saibam exatamente em que estado o projeto se encontra.

Auditoria de Configuração:

- Processos que garantem que os produtos desenvolvidos correspondam à documentação especificada e às versões aprovadas.
- Ajuda a verificar se todas as mudanças foram implementadas corretamente e documentadas.

Gerenciamento de Versões:

- Controle e gerenciamento das diferentes versões de software para garantir que todos os membros da equipe estejam trabalhando na versão correta.
- Facilita o trabalho em equipe, minimizando conflitos e desarmonias nas alterações.

Importância do SCM na Qualidade de Software

- Integridade e Confiabilidade: Garante que todos os componentes do software estejam em conformidade e que as modificações não introduzam erros.
- Colaboração Eficiente: Permite que múltiplos desenvolvedores trabalhem no mesmo projeto sem interferir uns nos outros, promovendo uma maior transparência e colaboração.
- Facilita a Manutenção: Com um histórico claro e estrutura organizada, é mais fácil identificar problemas, aplicar correções e manter o software.
- Redução de Riscos: Minimiza o risco de falhas ao fornecer um controle sistemático sobre as mudanças e suas repercussões.

Teste de Software

Objetivos do Teste de Software

- **Verificar Funcionalidade:** Assegurar que o software funcione conforme os requisitos.
- **Identificar Defeitos:** Detectar falhas e bugs antes da entrega.
- **Validar Requisitos:** Confirmar que todas as especificações foram atendidas.
- **Garantir Qualidade:** Assegurar que o software é confiável e atende aos padrões de qualidade.

Tipos de Teste de Software

Existem vários tipos de testes, cada um com seu foco e objetivos específicos:

- **Testes Funcionais:**
    - Validam se as funções do software atendem aos requisitos.
    - Exemplos: Testes de Unidade, Testes de Integração, Testes de Sistema.

- **Testes Não Funcionais:**
    - Avaliam características como desempenho, usabilidade, segurança e robustez.
    - Exemplos: Testes de Performance, Testes de Carga, Testes de Segurança.
- **Testes Manuais e Automatizados:**
    - **Manuais:** Testes realizados por testadores humanos que interagem com o software.
    - **Automatizados:** Testes executados por ferramentas e scripts, ideal para testes repetitivos.

Técnicas de Testes

As técnicas de teste podem ser categorizadas em várias abordagens:

- **Teste Baseado em Requisitos (ou Caixa Branca):**
    - Examina as funcionalidades com base nos requisitos do sistema.
    - Utiliza a análise de requisitos e documentação para desenvolver casos de teste.
- **Teste Funcional (ou Caixa Preta):**
    - Foca na entrada e saída do sistema sem considerar a lógica interna.
    - Testa se o sistema atende às expectativas com base em especificações.
- **Teste de Unidade:**
    - Verifica componentes individuais ou funções do software.
    - Pode ser automatizado e é normalmente feito pelos desenvolvedores.
- **Teste de Integração:**
    - Avalia como diferentes módulos ou serviços interagem entre si.
    - Pode ser realizado de forma incremental (teste de novos módulos) ou após a integração completa.
- **Teste de Sistema:**
    - Avalia o sistema completo em um ambiente que simula o ambiente de produção.

- **Teste de Aceitação:**
    - Realizado pelo cliente ou usuários finais para validar se o sistema atende suas necessidades.

Critérios de Teste

Os critérios de teste definem as condições sob as quais um teste deve ser executado. Os principais critérios incluem:

- **Critérios de Entrada:**
    - Definem as condições necessárias para iniciar os testes, como a conclusão de um desenvolvimento, disponibilidade de documentação e ambiente de teste.
- **Critérios de Saída:**
    - Determinam quando os testes podem ser considerados concluídos, como a cobertura de testes, a gravidade dos defeitos encontrados e se os requisitos foram atendidos.
- **Critérios de Cobertura:**
    - Avalia a extensão dos testes aplicados e pode incluir:
        - **Cobertura de Código:** Percentual de código executado durante os testes.
        - **Cobertura de Funcionalidade:** Percentual de requisitos testados.
        - **Cobertura de Decisão:** Percentual de decisões lógicas (if-else) testadas.

Processo de Teste de Software

Um processo de teste geralmente segue as etapas:

1. **Planejamento do Teste:**
    - Definição de estratégias, escopo e recursos necessários.
2. **Análise de Teste:**
    - Revisão dos requisitos e risco para identificar itens a serem testados.

3. **Design de Teste:**
    - Criação de casos de teste e preparação do ambiente.
4. **Execução de Teste:**
    - Realização dos testes e documentação dos resultados.
5. **Avaliação de Saída:**
    - Análise dos resultados para entender a qualidade do software.
6. **Fechamento:**
    - Revisão do teste, documentação de lições aprendidas e geração de relatórios finais.

Conclusão

Vimos que SCM é uma disciplina vital dentro do desenvolvimento de software que garante que o software e seus artefatos associados sejam controlados e gerenciados ao longo de seu ciclo de vida, sendo crucial para garantir a qualidade, a integridade e a manutenção eficaz do mesmo. A implementação adequada do SCM não apenas facilita a colaboração e o rastreamento de alterações, mas também contribui significativamente para a eficácia geral do processo de desenvolvimento de software. Atualmente SCM é um pilar da qualidade de software que não só melhora a eficiência do desenvolvimento, como também garante que o produto final atenda aos padrões de qualidade necessários.

Por sua vez o teste de software é uma prática essencial para garantir a qualidade e a funcionalidade dos sistemas. Utilizando técnicas variadas e definições claras de critérios de teste, as equipes podem minimizar riscos e entregar um produto que atenda às expectativas dos usuários. Se precisar de mais detalhes sobre algum aspecto específico do teste de software, estou à disposição!

Os sistemas de software estão em constante evolução, a manutenção do software, isto é, modificações em artefatos existentes, chega a consumir 75% do custo total do seu ciclo de vida. Aproximadamente, 20% de todo o esforço de manutenção é usado para consertar erros de implementação e os outros 80% são utilizados na adaptação do software em função de modificações em requisitos funcionais, regras de negócios e na reengenharia da aplicação. A Gerência de Configuração de Software surgiu da necessidade de controlar estas modificações, por meio de métodos e ferramentas, com o intuito de maximizar a produtividade e minimizar os erros cometidos durante a evolução.