Qualidade e teste de software

Aula 7: Plano de Testes e Casos de Teste

Apresentação

A elaboração de um bom plano de teste e de casos de teste é fundamental para o desenvolvimento do sistema. Para nos ajudar nesse processo de uma forma prática, vamos conhecer a ferramenta TestLink.

Primeiro, é necessário que você entenda o que é e o que fazer para elaborar um plano de testes.

Nesta aula, você também aprenderá como planejar e executar os testes, conhecendo a importância da utilização da UML e os casos de uso para a elaboração dos planos de teste.

Objetivos

- Descrever o desenvolvimento prático de elaboração e realização de planos e casos de teste com a ferramenta TestLink;
- Definir e praticar a elaboração de um plano testes, planejamento e execução dos testes;
- Reconhecer a importância da UML e dos casos de uso para a elaboração dos planos de teste.

(Fonte: selinofoto / Shutterstock).

Gestão de testes

A gestão de testes tem grande importância junto ao ciclo de vida de desenvolvimento do softwares.

Por isso, ao gerir os testes, nos permitimos documentar e evidenciar cada etapa da execução dos testes do projeto. A gestão de testes nos possibilita também uma visão completa de sua evolução e da qualidade do software.

Gestão de testes

Clique no botão acima.

Gestão de testes

Por que usar ferramentas de gerenciamento de testes?

Essas ferramentas oferecem um repositório central e padronizado, no qual os testadores poderão:

- Criar uma coleção de casos de teste (suítes com os casos de teste);
- Atribuir essas suítes aos seus respectivos testadores;
- Acompanhar a situação da execução dos testes;
- Emitir relatórios com métricas e estatísticas.

O que é uma suíte de teste?

Conforme o livro *Garantia da qualidade de software*, a suíte (ou situação) de teste finaliza o processo de detalhamento dos testes de validação e:

- Estabelece como será testado um determinado conjunto de casos de uso;
- Define quais procedimentos e em que ordem serão executados;
- Possibilita validar o comportamento dos vários cenários de determinados requisitos de software.

Para as suítes de teste, devemos abordar os seguintes itens:

- Identificação das suítes de testes;
- Definição das suítes de teste;
- Prerrequisitos de cada suíte;
- Definição dos procedimentos de execução dos testes.
- Cronograma detalhado.

O que é um caso de teste?

É o documento de registro de todo o planejamento dos testes, estabelecendo o que será testado.

Sua finalidade é identificar o maior número de cenários e variações de determinado requisito de software.

Determina as informações empregadas durante os testes dos cenários e os resultados esperados, estabelecendo a massa crítica de testes para validar os requisitos do software.

Itens abordados:

- Identificação das condições de testes;
- Identificação dos casos de testes;
- Definição de cada caso de teste identificado;
- Detalhamento da massa de entrada;
- Arquitetura do ambiente de teste;
- Critérios especiais para a geração da massa de testes (d+0, d-30, m+1, a+18);
- Responsáveis pelo levantamento;
- Cronograma detalhado;

- Definir agenda de levantamentos (como testar).

Existe uma norma (IEEE 829) que propõe um padrão de documentação a ser seguido pelas organizações:

- Esta norma, do Instituto de Engenheiro Eletricistas e Eletrônicos (ou Padrão 829 para Documentação de Teste de Software), é um padrão IEEE que especifica a forma de uso de um conjunto de documentos em oito estágios definidos de teste de software, cada estágio potencialmente produzindo seu próprio tipo de documento;
- Propõe um padrão de documentação que deveria ser obedecido por todas as organizações que trabalham com teste de software.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Utilizando a ferramenta TestLink

Além dos casos de teste, precisamos também do plano de teste.

Necessidade dos planos de testes:

Permitem que membros da equipe de teste acompanhem os casos de teste, executem e vejam os resultados esperados.

A ferramenta também permite a geração de relatórios que auxiliam o coordenador da equipe a priorizar e atribuir tarefas.

Permitem adicionar os resultados de cada um dos testes executados e também visualizar os resultados globais deles.

Para auxiliar nessa documentação de teste, existem várias ferramentas disponíveis, e uma delas é o TestLink. Leia o texto seguinte para conhecer sobre essa ferramenta.

TestLink

Clique no botão acima.

TestLink

Trata-se de uma aplicação open source voltada para a gestão de testes, desenvolvida e mantida por várias equipes ao longo dos anos. Oferece suporte para criação, execução e manutenção de casos de teste, planos de testes e requisitos.

Permite a geração de relatórios gerenciais e estatísticos sobre os testes executados e a integração com outras ferramentas de gerenciamento de bugs. (Caetano, 2007)

Características e funcionalidades do TestLink:

- É baseado em planos e casos de teste;
- Suporta o gerenciamento de múltiplos projetos;
- Possibilita a criação de um repositório (banco de dados) centralizado para todos os casos de teste e os resultados;
- Possibilita a integração com ferramentas de gestão de defeitos, como Mantis, Jira e Bugzilla;
- Possibilita a geração de relatórios;
- Permite autenticação via **LDAP** (é uma forma abreviada de falar *Active Directory* (que é um serviço de diretório feito pela Microsoft). Significa *Lightweight Directory Access Protocol* e consiste em um meio para consultar itens em qualquer serviço de diretório que a suporta.).

Ele é muito importante para o processo de desenvolvimento do software, porém é preciso mantê-lo sempre atualizado.

Com o uso do Testlink, é possível:

- Escrever, e consequentemente documentar, os casos de teste;
- Reduzir a duplicação de esforços, pois a existência de uma base de documentação permite o reaproveitamento de esforços passados, sem a necessidade de criar todos os casos de testes novamente.

Motivação para utilizar o TestLink

De acordo com o *Manual de TestLink*, a ferramenta possibilita:

- Criar vários projetos;
- Exportar e importar casos de teste facilmente;
- Atribuir casos de teste para usuários específicos;
- Gerar planos e relatórios de teste em vários formatos;

Desvantagem dessa ferramenta

Não dispõe de um mecanismo que mapeie e gerencie automaticamente casos de uso como casos de teste.

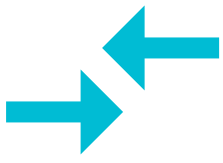
Saiba mais

Leia o artigo *TestLink: gerenciando atividades de teste*.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

O que é plano de testes?

Um documento de plano de testes (eventualmente definido na etapa de análise de requisitos) define quantos e quais testes serão realizados.



Um plano tem o papel semelhante ao de um "mapa". Sem ele, nenhum participante dos testes conhecerá seus objetivos, nem aonde quer chegar e, o pior de tudo, jamais terá a certeza de ter alcançado sua meta.

O propósito desse planejamento é monitorar a execução de atividades, sendo também importante conhecer o papel dos riscos no planejamento e diferenciar as estratégias de planos.

O planejamento engloba três atividades principais:

1

Definir um cronograma de atividades.

2

Fazer alocação de recursos.

3

Definir marcos de projeto: estabelecer os marcos a serem alcançados, com o objetivo de fazer o acompanhamento.

Esse planejamento é acompanhado da atividade de monitoração ou supervisão, que objetiva avaliar se o progresso que tem sido alcançado está em conformidade com o que foi estabelecido no plano.

O plano de teste é um dos oito documentos descritos na IEEE 829. De acordo com ela, a estrutura do plano de teste consiste de uma série de partes (ou seções) descritas a seguir, na elaboração do plano de teste.

O plano de teste é um dos documentos produzidos na condução de um projeto. Quem pode elaborar esse documento é o gerente de projeto ou o gerente de testes.

O que visa o plano de teste?

- Planejar as atividades a serem realizadas;
- Definir os métodos a serem empregados;
- Planejar a capacidade necessária;
- Estabelecer métricas e formas de acompanhamento do processo.

É importante que o plano contenha os seguintes itens:

- Uma introdução com identificação do projeto (definições, abreviações, referências), definição de escopo e objetivos;
- O conjunto de requisitos a serem testados;
- Os tipos de testes a serem realizados e ferramentas utilizadas;

- Os recursos utilizados nos testes;
- Um cronograma de atividades (e definição de marcos de projeto).

Atenção

O planejamento é necessário a fim de **antecipar** o que pode ocorrer e provisionar os recursos necessários nos momentos adequados.

Elaboração do plano de teste

Sem planejamento, fica mais difícil o desenvolvimento de qualquer projeto. O plano é como se fosse um mapa; com ele, podemos chegar ao nosso destino.

Clique nos botões para ver as informações.

Requisitos a serem testados

- Relacionar os itens de teste, descrevendo todos os elementos que serão testados;
- Descrever, isoladamente, funcionalidades a serem e a não serem testadas.

Vejamos alguns exemplos de requisitos a serem testados: desempenho, segurança, interface de usuário, controle de acesso, funcionalidades.

Estratégias e ferramentas de teste

- Descrever a estratégia do teste, geralmente para cada grupo de funcionalidades das seções anteriores;
- Abordar questões como atividades e ferramentas usadas no teste;
- Descrever o critério de sucesso ou falha do caso de teste e outra descrevendo o critério de suspensão e requisitos de reinício, como por exemplo atividades que devem ser feitas antes de reiniciar o teste após um evento de suspensão;
- Listar o conjunto de ferramentas utilizadas.

Equipe (responsabilidades e requisitos humanos) e infraestrutura

- Mostrar as necessidades físicas para a realização do teste. Exemplo: hardware e software, e como elas podem afetar a execução do teste;
- Mostrar os diferentes papéis desempenhados no projeto de teste;
- Os recursos humanos e requisitos de treinamento da equipe de teste.

Cronograma de atividade

- Descrição de marcos importantes das atividades (incluindo as datas de início e fim da atividade). Exemplo: projeto de testes, execução de testes ou avaliação de testes;
- Riscos e contingências.

Documentação

- Relação dos documentos pertinentes ao projeto;
- Aprovações;
- Assinatura dos líderes do projeto, aprovando o documento.

Planejamento e execução dos testes

O planejamento faz parte da primeira fase do processo de revisão. Devemos selecionar a equipe, alocar as funções, definir os critérios de entrada e de saída para os diversos tipos de revisão e selecionar quais partes dos documentos serão observados.

O planejamento tem que estar de acordo com o alinhamento estratégico da empresa.

(Fonte: SFIO CRACHO / Shutterstock).

Planejar teste

O objetivo de planejar teste é definir o plano de testes para o projeto. Devem ser identificados os tipos de testes e os artefatos que devem ser testados, sempre usando os fatores de criticidade das ameaças.

A coordenação dessa atividade é feita pelo gerente de teste e pelo gerente de projeto.

1

Para planejar o teste, devem ser identificados vários artefatos como: programas, rotinas, sub-rotinas, arquivos, equipamentos, softwares de transmissão, sistemas operacionais, instalações e outros. Dependendo do domínio, eles devem ser eleitos para fazer parte de um conjunto de artefatos sujeitos a testes de missão crítica.

2

Executar teste

Tem como objetivo testar os artefatos selecionados na macroatividade anterior, conforme definido no plano de testes. Dependendo do grau de criticidade do sistema, será necessária também a verificação desse artefato. Essa verificação deve ser executada através de uma revisão por pares além do teste, conforme planejado.

Planejamento e execução

Vamos tomar como exemplo um sistema crítico.

Quando iniciarmos a atividade de planejamento dos testes, devemos fazer a revisão dos requisitos, a homologação oficial dos mesmos com os especialistas do domínio e usuários responsáveis e/ou homologadores e o planejamento e a elaboração de testes de aceitação.

Atividades de testes do ciclo de desenvolvimento de uma aplicação que devem começar no início do projeto:

- Fixação da estratégia de teste e conceitos de testes;
- Análise de risco, levantamento das ameaças, estimativa dos gastos com os testes, recursos humanos, máquina e tempo necessário para executar os testes;
- Elaboração do plano de testes;
- Treinamento da equipe de testes, se necessário;
- Estabelecer processos e métricas (não confundir com medidas) de controle;
- Definir recursos de hardware, computadores (equipamentos), mesas, cadeiras etc.;
- Providenciar recursos de software, bases de dados (definir bases de teste), confidencialidade, confiabilidade, coerência, controle de versão, ferramentas de testes etc.

Deve-se sempre verificar o que foi feito para corrigir problemas, antes de seguir para o desenvolvimento.

Ao olharmos para a frente, devemos fazer perguntas tais como:

Os requisitos críticos podem ser testados?

Os custos previstos são sustentáveis?

Se as respostas forem corretas, então não haverá problemas para que esses requisitos sejam implementados.

Se não existir um caminho claro de como os requisitos vão ser testados, então é provável que não se tenha ideia de como fazer para implementar esses requisitos. (SPILLNER, 2002).

Os principais erros que são cometidos foram resumidos em um artigo publicado por Brian Marick em 1997. Foram identificados cinco grupos importantes de erros comumente cometidos por quem testa softwares, nos seguintes pontos:

Principais erros

Clique no botão acima.

Principais erros

1. No propósito da atividade de teste: ocorre quando o ator que controla a execução não entende bem o sentido de testar e não aproveita os resultados de forma eficaz.
   Os erros comuns são:
   - Atribuir a responsabilidade pela qualidade unicamente à equipe de teste;
   - Achar que a tarefa de equipe de testes é simplesmente encontrar erros;
   - Não encontrar os erros importantes;
   - Oferecer estatísticas de erros sem o contexto relevante.

2. No planejamento dos testes: esses erros estão relacionados à fase de planejamento dos testes:
   - Concentrar-se exageradamente em teste funcional;
   - Não enfatizar o teste de configuração;
   - Deixar o teste de carga para o final do processo;
   - Não testar a documentação nem a instalação;
   - Teimosia em aplicar um plano de teste ineficaz.

3. No pessoal contratado para testar: ao montar a equipe que conduzirá os testes, é importante se concentrar nos seguintes erros comuns e tentar evitá-los:
   - Usar o teste como um trabalho temporário para programadores novos;
   - Recrutar ex-programadores (os que não são os melhores) para a equipe de teste;
   - Usar testadores que não conhecem o domínio da aplicação;
   - Não contratar pessoal de suporte técnico e documentação;
   - Insistir que testadores sejam programadores;
   - Não utilizar uma equipe de teste diversificada.

4. Na execução dos testes em si - Durante a execução dos testes, um conjunto de problemas comuns é:
   - Dar mais atenção à execução dos testes do que ao seu projeto;
   - Não rever os projetos de teste;
   - Ser específico demais nas entradas e procedimentos do teste;
   - Não notar nem explorar irregularidades peculiares;
   - Elaborar suítes de teste compreendidas apenas por seus criadores;
   - Adicionar apenas testes de regressão quando problemas são encontrados;
   - Não manter um histórico de anotações para os próximos testes.

5. Na aplicação da tecnologia nos testes:
   A aplicação de tecnologia à atividade de teste é muitas vezes benéfica, mas nem sempre, e nunca desenfreadamente.
   A seguir, os erros que envolvem o foco tecnológico do teste:
   - Tentar automatizar todos os testes;
   - Esperar reexecutar testes manuais;
   - Usar ferramentas de automação de interface para reduzir o custo do teste;
   - Não analisar a cobertura de código em absoluto.

Atenção! Aqui existe uma videoaula, acesso pelo conteúdo online

Importância da UML e casos de uso para a elaboração dos planos de teste

Utilização da UML no processo iterativo e incremental

A UML é independente do processo de desenvolvimento.

- Vários processos podem utilizar a UML para modelagem de um sistema OO.

Quando os artefatos de software são construídos através da UML, eles evoluem à medida que as iterações são realizadas, assim:

- A cada iteração, novos detalhes são adicionados a esses artefatos;
- Além disso, a construção de um artefato fornece informações para adicionar detalhes a outros.

Defeitos no desenvolvimento de software

Defeitos normalmente são introduzidos na transformação de informações entre as diferentes fases do ciclo de desenvolvimento de um software. (PRESSMAN, 2005)

Exemplo simples de ciclo de vida de desenvolvimento de software

Clique no botão acima.

Exemplo simples de ciclo de vida de desenvolvimento de software

Os requisitos expressos pelo cliente são relatados textualmente em um documento de especificação de requisitos.

O documento é transformado em casos de uso que, no que lhe diz respeito, foi o artefato de entrada para o projeto do software e definição de sua arquitetura utilizando diagramas de classes da UML.

Para chegarmos ao produto final, uma série de transformações é realizada. Vamos mostrar uma **figura clássica** que expressa exatamente essa ideia por meio de uma metáfora.

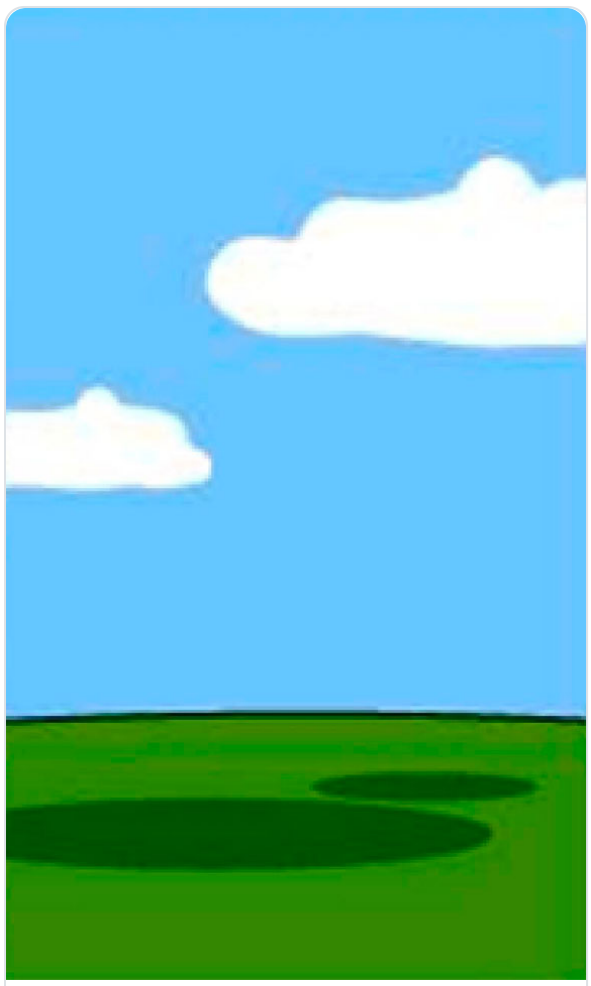
Como o cliente explicou...

Como o líder de projeto entendeu...

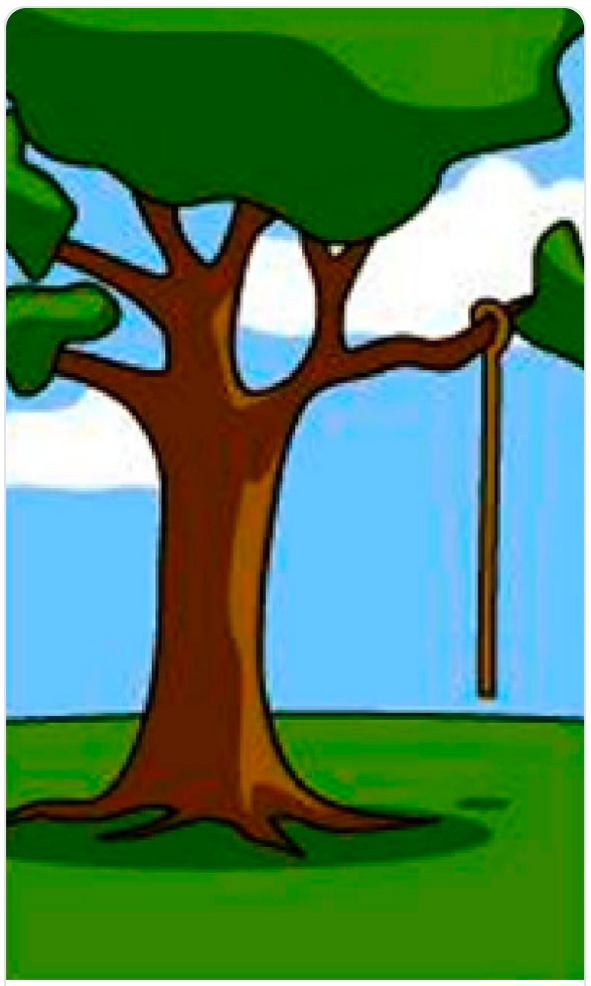
Como o analista projetou...

Como o programador construiu...

Como o consultor de negócios descreveu...

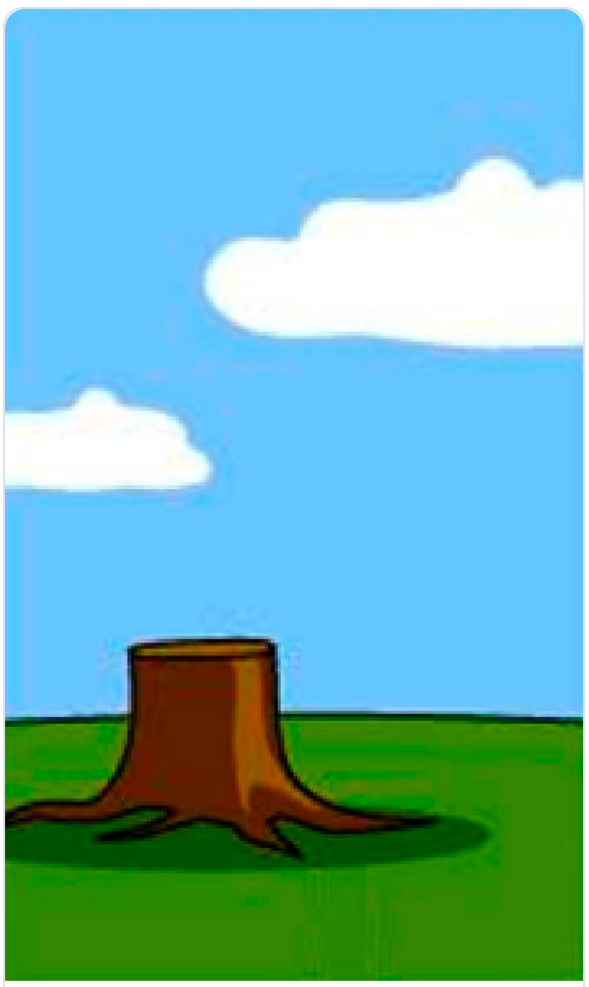


Como o projeto foi documentado...



Que funcionalidades foram instaladas...

Como o cliente foi cobrado...



Como foi mantido...

Como o cliente realmente queria...

Métodos de análise de documentos

Qualquer documento pode conter elementos essenciais para auxiliar na localização de novos casos de testes e no refinamento e ampliação do esforço de planejamento.

Supondo o uso da orientação a objeto juntamente com a linguagem UML como padrão de documentação, as principais fontes para extrair os casos de testes são: diagrama de estado e de atividades.

Clique nos botões para ver as informações.

Diagrama de estado

Representa um estado ou situação em que um objeto pode se encontrar no decorrer da execução de processos de um sistema. O objeto passará do estado inicial para o final por meio de uma transição.

Pense no diagrama de estado do ciclo de vida de um vídeo.

Exemplo

Cada transição de um estado para outro vídeo deverá ser considerada um caso de teste (cenário positivo), enquanto que as transições proibidas deverão ser inseridas como cenários negativos, que também deverão ser testadas.

Representação do diagrama de estado:

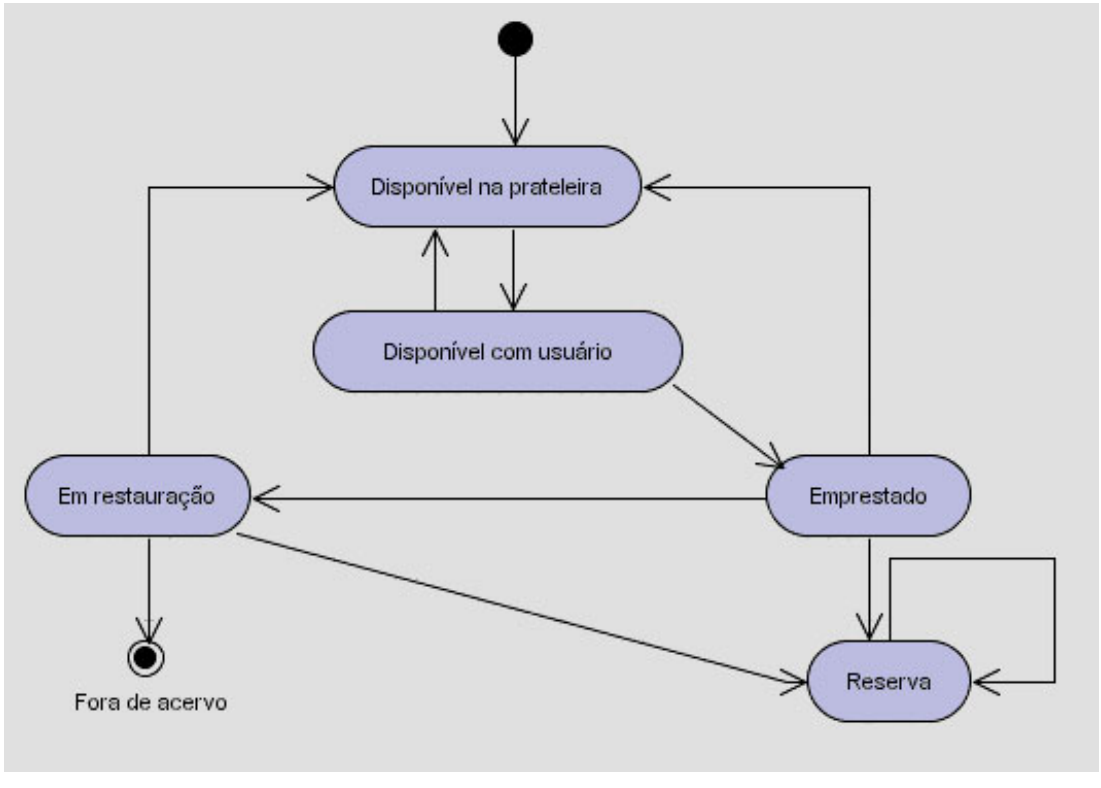
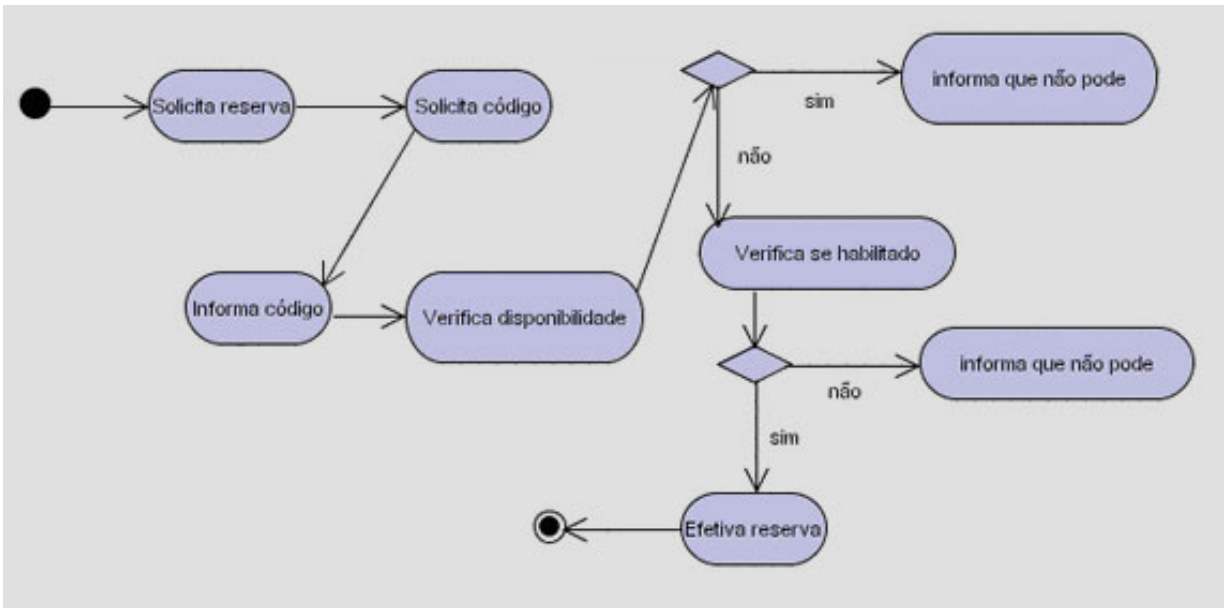


Diagrama de atividades

Representa todo o fluxo de processamento de um determinado evento de negócio, revelando todos os caminhos alternativos (cenários positivos) e as situações que impossibilitam a finalização desse evento (cenários negativos).

Deve revelar o conjunto completo de casos de testes que precisarão ser inseridos no planejamento de testes.

Representação do diagrama de atividades:



Modelo de casos de uso

Casos de uso são coleções de cenários relacionados de sucesso e fracasso, que descrevem atores usando um sistema como meio para atingir um objetivo. (Larman, 2007)

Saiba mais

Deve-se observar que casos de uso não são diagramas. Eles são modelados basicamente redigindo textos. Entretanto, a UML define um diagrama para representar casos de uso.

Os casos de uso são escritos em diferentes tipos de formalidade (resumido, informal e completo), dependendo da necessidade, mas é no modo completo que todos os passos e variantes são descritos, com precondições e garantias de sucesso.

O cenário de sucesso principal (ou fluxo básico) descreve o caminho de sucesso que satisfaz os interessados.

As extensões (ou fluxos alternativos) descrevem instruções condicionais ou desvios do cenário de sucesso. Uma extensão é composta por uma condição e um tratamento. A condição deve ser algo que possa ser detectado pelo sistema ou ator. O tratamento é uma sequência de passos. (Larman, 2007)

Os casos de uso devem ser expressos em níveis de intenções e responsabilidades, independente de tecnologias, especialmente relacionadas à interface com o usuário. Deve-se especificar **o que** o sistema deve fazer, e não **como** deve ser feito. (Larman, 2007)

A documentação dos casos de uso é uma excelente descrição de testes funcionais, pois já contém um levantamento de todas as condições que podem causar falhas e os respectivos tratamentos. Pelo menos um caso de teste é necessário para cada extensão. (Cockburn 2005)

Os casos de uso também podem ser usados como base de casos de testes e em testes de regressão. (Booch; Rumbaugh; Jacobson, 2005)

Atividade

1. Baixe o software TestLink e faça um teste para alguns casos de teste que você deverá desenvolver. Faça também uma pesquisa sobre os temas relacionados com o assunto para saber que abordagens estão sendo utilizadas no cenário atual do desenvolvimento de software.

2. No caso de teste através do método de análise de documentos, caso estejamos utilizando a orientação a objeto em conjunto com a linguagem UML como padrão de documentação, quais as principais fontes para extrair os casos de testes?

a) Diagrama de atividades e de estado
b) Diagrama de estado e o código-fonte
c) Somente o código-fonte
d) Diagrama de atividades e o código-fonte
e) Caso de uso e diagrama de condição

3. Na empresa, seu chefe solicitou que você elaborasse a documentação da abordagem da equipe de software para os testes a serem realizados em uma importante aplicação web da sua empresa.

Essa documentação deve conter a definição do plano que descreve a estratégia global e o procedimento, designando as etapas específicas do teste, assim como os tipos de testes que serão aplicados.

Nesse caso, qual documento você deverá elaborar?

a) Massa de teste
b) Caso de uso
c) Script de teste
d) Caso de teste
e) Especificação de teste

---

4. Existem determinadas ferramentas que possibilitam o gerenciamento e o controle do processo de execução, reexecução e medição dos testes automatizados e a integração entre as demais fases.

O objetivo dessas ferramentas é executar os testes selecionados no planejamento, tendo como principais características: a análise de cobertura, a execução de scripts, simuladores de performance e testadores de memória.

Neste caso, são classificadas como ferramentas:

a) De suporte aos testes
b) De modelagem e automação
c) De revisões e inspeções
d) De planejamento de testes
e) De execução e conferência

---

5. Existem diferentes papéis com diferentes reponsabilidades dentro de uma equipe de teste independente. Marque a opção **INCORRETA**:

a) Gerente de teste: responsável pela liderança de um projeto de teste específico.
b) Analista de teste: responsável pela modelagem e elaboração dos casos de testes e scripts de teste.
c) Arquiteto de teste: responsável pela montagem do ambiente de teste (infraestrutura) e escolha de ferramentas.
d) Testador: responsável pela execução dos casos de teste e scripts de teste.
e) Product Owner: responsável pela análise dos dados de teste.

Referências

BARTIÉ, A. **Garantia de qualidade de software**. 1. ed. Rio de Janeiro: Campus, 2002.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. 2.ed. **UML**: Guia do usuário. Rio de Janeiro: Campus, 2005.

LINHA DE CÓDIGO. CAETANO, C. **Automação e gerenciamento de testes: aumentando a produtividade com as principais soluções open source e gratuitas**. Disponível em: http ://www .linhadecodigo .com .br /artigo /1566 /automacao -e -gerenciamento -de -testes -aumentando -a -produtividade -com -as -principais -solucoes -open -source -e -gratuitas -2a -edicao .aspx. Acesso em: 31 maio 2019.

COCKBURN, A. **Escrevendo casos de uso eficazes**. Porto Alegre: Bookman, 2005.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. São Paulo: Makron Books, 1995.

SOMMERVILLE, Ian. **Engenharia de Software**. 9. ed. São Paulo: Pearson, 2011.

Próxima aula

- Desenvolvimento de rotinas de teste com base no framework Cucumber e automação com Selenium WebDriver;
- Metodologias utilizadas para testes de aceitação;
- Behavior Driven Development (BDD).

Explore mais

Assista aos vídeos:

- Tutorial Testlink - disciplina teste e qualidade de software
- Laboratório de testes com TestLink

Leia os textos:

- Teamgeist online
- Testlink - uma ferramenta de gerenciamento de testes de software
- Como fazer um plano de testes baseado em casos de uso
- Criação e geração de planos de teste de software