



Artigo



Casos de Teste: Importância da rastreabilidade entre requisitos

Veja nesse artigo como estabelecer mecanismos de rastreabilidade para garantir que os requisitos sejam atendidos pelo sistema e permitir avaliar o impacto de alterações nos requisitos e os casos de teste.

Por que eu devo ler este artigo: A rastreabilidade dos requisitos de software é uma atividade da gerência de requisitos e está associada à qualidade do desenvolvimento de software. Uma decisão importante

[Ver mais](#)

Marcar como lido



Anotar



Artigos



Casos de Teste: Importância da rastreabilidade entre requisitos

Teste pode ser definido como uma atividade que tem por objetivo verificar se o software produzido está de acordo com sua especificação e se satisfaz as expectativas do cliente. Teste de software é parte integrante do processo de Validação e Verificação (V&V) da Engenharia de Software.



tarefas que asseguram que o software foi criado e pode ser rastreado segundo os requisitos do cliente. A definição de V&V abrange muitas atividades de garantia da qualidade de



Das atividades de verificação e validação, a atividade de teste é considerada um elemento crítico para garantia da qualidade. Um problema na atividade de teste de software esta relacionado a como determinar se um programa P foi suficientemente testado e pode ser liberado para os usuários com razoável confiança de que funcionará de modo aceitável.

O significado de aceitável varia para uma aplicação específica em função da criticidade das funções, consequências antecipadas de um mau funcionamento e da frequência de uso esperada. Apesar dos esforços na utilização de modelos de melhoria de processos em organizações de desenvolvimento de software, os padrões para determinar a adequação da atividade de teste são praticamente inexistentes. Um conjunto de problemas no processo de teste que devem ser evitados são:

- Cronogramas agressivos deixando a equipe de teste impossibilitada de completar os testes planejados devido à falta de recursos, equipe não qualificada e falta de tempo;
- Falta de rastreabilidade de casos de teste entre diferentes versões do sistema, dificultando o reuso e repetição dos testes após modificações nos requisitos;
- Testes manuais, pois há um grande esforço da equipe de testes a cada início de uma nova atividade de teste, consequentemente quando o software sofre manutenção e a documentação não é atualizada;
- Ambiente de teste diferente do ambiente de produção;
- Ausência de critérios para seleção dos casos de teste, definição da sua completude e estabelecimento de um ponto de parada.

Este artigo aborda a rastreabilidade de requisitos, enfatizando aspectos relacionados às informações de rastreabilidade entre requisitos e casos de teste. Nosso principal propósito é apresentar ao leitor alguns aspectos que envolvem a rastreabilidade visando conscientizá-

Uma estratégia para teste de software pode ser realizada numa série de quatro passos executados sequencialmente, conforme apresentado na **Figura 1** e explicado a seguir:



funções operam adequadamente. O teste de unidade focaliza o esforço de verificação na menor unidade de projeto do software, componente ou módulo de software.

2. Em seguida os componentes devem ser integrados (teste de integração) para formarem um pacote completo de software.
3. Depois que o software passou pelo processo de teste de unidade e integração testes de validação são conduzidos. Os requisitos estabelecidos com parte da análise de requisitos do software são validados contra a implementação do sistema.
4. Finalmente o software deve ser combinado com outros elementos do sistema e testado como um todo. Testes de sistema verificam se todos os elementos estão integrados de maneira adequada e se todas as funcionalidades do sistema são atendidas [5].

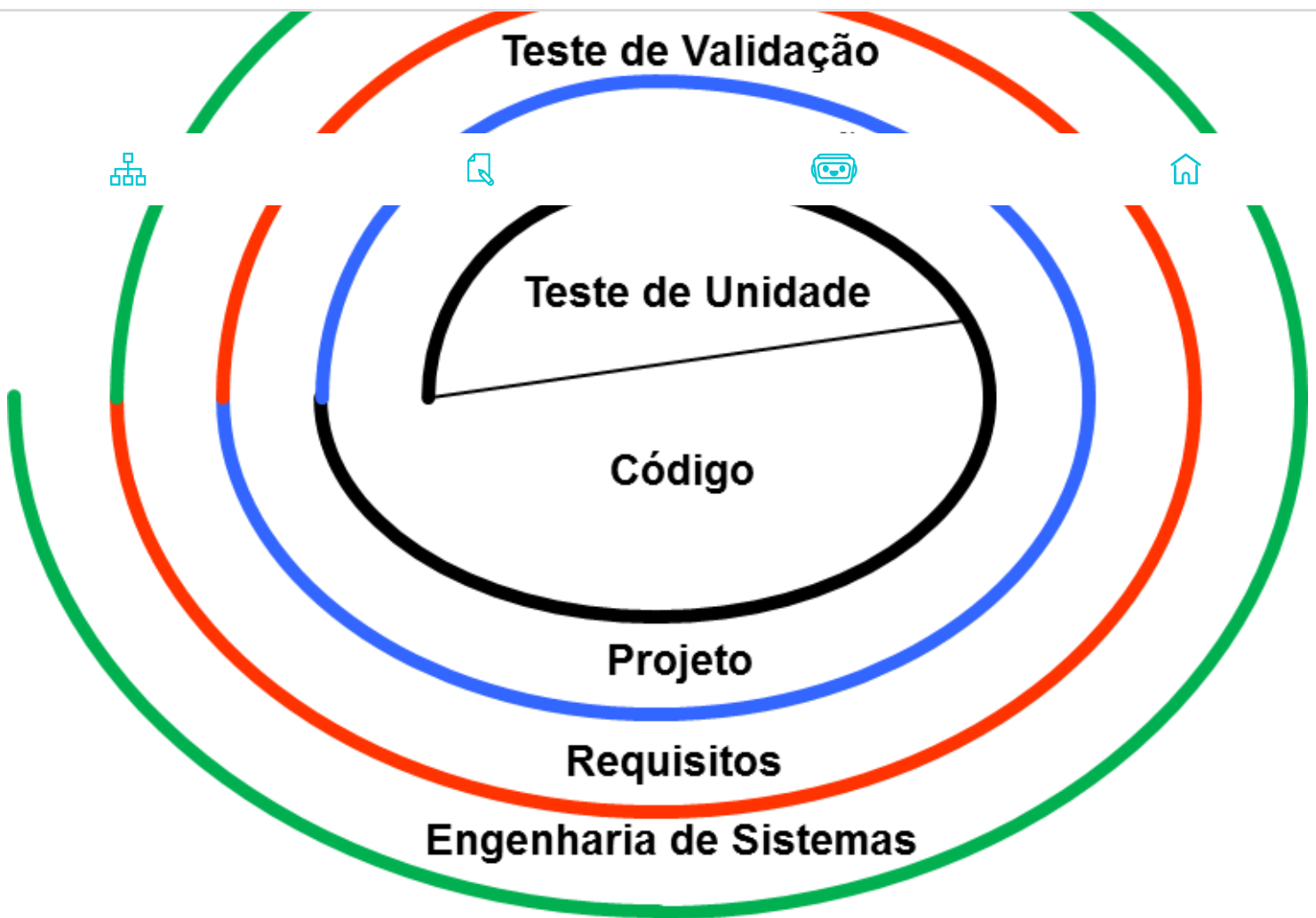


Figura 1. Estratégia de teste [5]

A aplicação de uma estratégia de teste é uma aliada na garantia de qualidade, mas somente terá sucesso se os testadores de software realizarem as seguintes atividades [5]:

- Especificarem o software de maneira quantificável, muito antes de começarem os testes: além de encontrar erros, uma boa estratégia de teste também avalia outras características de qualidade, como: portabilidade, manutenção, confiabilidade, usabilidade, entre outros. O objetivo de mensurar esses requisitos é que não ocorra ambiguidade nos resultados dos testes;
- Definirem explicitamente os objetivos do teste: Os objetivos específicos dos testes deverão ser definidos em termos mensuráveis no plano de testes. Por exemplo: eficiência e abrangência do teste, tempo médio de falhas e custo para corrigir

de funcionalidades. O retorno gerado por esses testes de ciclo rápido pode ser usado para controlar os níveis e qualidade e as correspondentes estratégias de teste;



Criação de software que diagnostique certas classes de erros, além de testes automatizados;

- Utilizarem revisões técnicas eficazes com filtro antes do teste: As revisões técnicas podem ser tão eficazes como testes para descobrir erros, pois podem antecipar erros, reduzir o trabalho e produzir software de alta qualidade;
- Executarem revisões técnicas para avaliar a estratégia de teste e os próprios casos de testes: As revisões técnicas podem descobrir inconsistências, omissões e erros na abordagem de testes;
- Desenvolverem abordagem de melhoria contínua para o processo de teste: A estratégia de teste deverá ser medida, para que haja um controle estatístico de processo para o teste de software.

No cenário atual de desenvolvimento de software, além da estratégia de testes, há a necessidade de automatizar os testes, pois, quando desenvolvidos de forma adequada, serão facilmente executados.

Para agregar valor ao negócio e atender a pressão por prazos em projetos de desenvolvimento de software é necessário estabelecer um processo de automação de teste de software robusto [1].

A Programação eXtrema, em particular, recomenda explicitamente testes automatizados para ajudar na garantia da qualidade dos sistemas de software [4]. Em sistemas complexos o volume de scripts de teste pode crescer muito e é importante nestes casos uma estratégia de organização dos scripts.

Modelo de rastreabilidade



sendo solucionado muda constantemente. Esses requisitos devem evoluir para refletir essas novas visões do problema.



dos requisitos de sistema. Existem vários relacionamentos entre os requisitos e o projeto do sistema. Quando as mudanças são propostas, deve-se estabelecer meios para rastrear seu impacto em outros requisitos e no projeto do sistema, a chamada hierarquia de rastreabilidade.

A rastreabilidade é a propriedade de uma especificação de requisitos que reflete a facilidade de encontrar os requisitos relacionados. Os elementos do projeto envolvidos em rastreabilidade são chamados de itens de rastreabilidade. Os itens típicos de rastreabilidade incluem diferentes tipos de requisitos - funcionais e não funcionais, elementos de modelo de projeto e de análise, artefatos de testes (conjuntos de testes, casos de teste, etc.), material de treinamento e documentação de suporte a usuário final.

Em qualquer processo de desenvolvimento de software existe algum mecanismo de rastreabilidade implícito. Geralmente a rastreabilidade é estabelecida pelas relações formais entre os artefatos do processo.

Um exemplo deste tipo de rastreabilidade implícita é a convenção de nomenclatura, ou seja, a classe no modelo de projeto chamada “Cliente” é implementada pela classe no modelo de implementação chamada “Cliente”.

Três tipos de informações de rastreabilidade podem ser mantidos:

1. Informações de rastreabilidade da origem: elos de rastreamento (link) ligam requisitos aos envolvidos que propuseram os requisitos e a necessidade desses requisitos. Quando uma mudança é proposta pode-se utilizar essas informações para encontrar e consultar os envolvidos sobre a mudança.
2. Informações de rastreabilidade de requisitos: elos de rastreamento (link) ligam os



requisitos aos módulos de projeto, nos quais esses requisitos são implementados.

Permite avaliar o impacto das mudanças de requisitos no projeto e na implementação



As informações de rastreabilidade são frequentemente representadas por meio de matrizes de rastreabilidade que relacionam os requisitos aos envolvidos, a outros requisitos ou módulos do projeto. Para a criação de tabelas de rastreabilidade para auxiliar o processo de gestão de requisito é necessário identificar os requisitos.

A **Figura 2** apresenta uma tabela de rastreabilidade genérica, onde as linhas representam os requisitos e as colunas os atributos de requisitos que podem ser aspectos específicos do sistema ou do ambiente, inclusive casos de teste associados ao requisito.

Requisitos	Aspecto específico do sistema ou do seu ambiente								
	A01	A02	A03	A04	A05	A06	A07	A08	Aii
R01			✓		✓				
R02	✓		✓						
R03	✓			✓					✓
R04		✓			✓				
R05	✓	✓		✓					✓
Rnn	✓		✓						

Figura 2. Tabela de rastreabilidade genérica

Cada tabela de rastreabilidade relaciona os requisitos identificados a um ou mais aspectos do sistema ou de seu ambiente. Entre as muitas tabelas de rastreamento possíveis estão as seguintes:

- Tabela de rastreabilidade de característica: mostra como os requisitos se relacionam a características importantes do sistema/produto.

- Tabela de rastreabilidade de subsistemas: caracteriza os requisitos pelo subsistema que ele governa.



as interfaces internas e externas do sistema.

A estratégia de rastreabilidade apresentada na **Figura 3** relaciona os requisitos de usuário (Requisitos do Produto - Req A), os requisitos de sistema (Requisitos detalhados (Casos de Uso) - Req B), com artefatos de projeto, casos de teste e documentos de usuário.

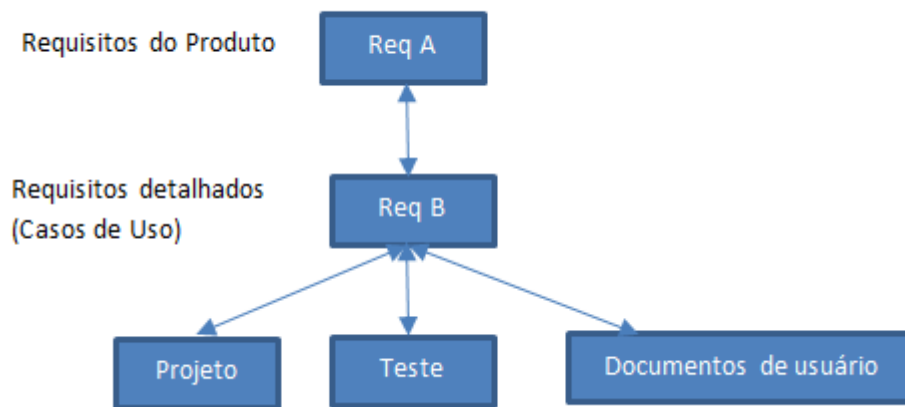


Figura 3. Estratégia de rastreabilidade de requisitos

O Rational Unified Process (RUP) é um processo de engenharia de software que oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento. Sua meta é garantir a produção de software de alta qualidade que atenda às necessidades dos usuários dentro de um cronograma e de um orçamento previsíveis.

O RUP considera entre as melhores práticas de desenvolvimento de software o gerenciamento de requisitos (ênfase na rastreabilidade de requisitos em todo o processo de desenvolvimento de software, conforme apresentado na **Figura 4**).

■ Compreender a origem dos requisitos;

■ Gerenciar o escopo do projeto;

■ Gerenciar mudanças nos requisitos;



■ Avaliar o impacto da falha de um teste nos requisitos (isto é, se o teste falhar, talvez o requisito não seja atendido);

■ Verificar se todos os requisitos do sistema são atendidos pela implementação;

■ Verificar se o aplicativo faz apenas o que era esperado que ele fizesse.

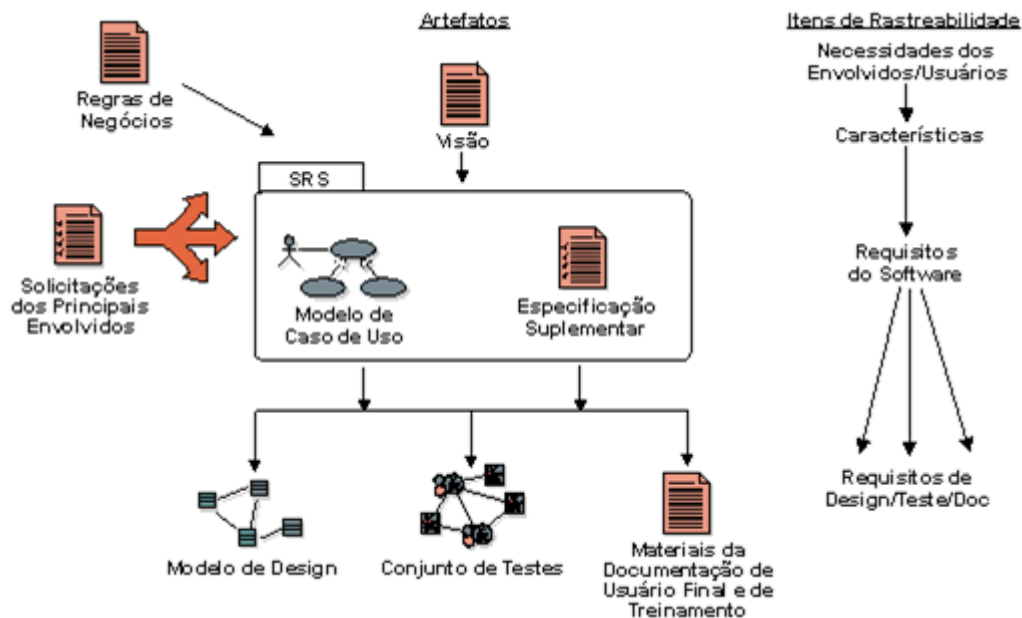


Figura 4. Hierarquia de rastreabilidade

De acordo com os artefatos do RUP, as informações fornecidas sobre os requisitos como Regras de Negócios e Solicitações dos Principais Envolvidos, são convertidas em um conjunto de necessidades chave dos envolvidos/usuários e características do sistema, conforme especificado no documento de Visão.

O modelo de caso de uso, por sua vez, descreve como essas características são convertidas na funcionalidade do sistema. Os detalhes de como o sistema interage com o mundo externo são capturados nos casos de uso, com outros requisitos importantes (como

usuário.

No caso de sistemas complexos, os casos de uso e as especificações suplementares podem



Requirements Specification) para uma "característica" particular ou outros agrupamentos de subsistemas.

O modelo de caso de uso e a especificação suplementar são artefatos de entrada continuamente para o modelo de design, conjunto de testes (onde serão especificados os casos de teste) e documentação adicional.

Um conceito-chave para ajudar a gerenciar mudanças nos requisitos é o vínculo de rastreabilidade "suspeito". Quando um requisito (ou outro item de rastreabilidade) muda em qualquer extremidade do vínculo de rastreabilidade, todos os vínculos associados àquele requisito são marcados como "suspeitos". Isso é uma marca para que seja realizada uma análise de mudança que determina se os itens associados precisarão mudar também. Esse conceito também ajuda a analisar o impacto de mudanças potenciais.

A rastreabilidade pode ajudar a responder ao seguinte conjunto de questões:

- Necessidades dos usuários que não foram vinculadas a características do produto;
- Status dos testes em todos os casos de uso na interação #n;
- Requisitos suplementares vinculados a testes que possuem status não testado;
- Resultados de testes que falharam, em ordem de importância;
- Características programadas para este lançamento, quais necessidades de usuários elas satisfazem e o status delas.

Uma consideração, é que o RUP pode ser utilizado para todos os tipos de projeto de software, de simples a complexo. Recentemente, um número cada vez maior de modelos ágeis, como eXtreme Programming (XP), SCRUM, Feature-Driven Development (FDD) e a metodologia Crystal Clear, têm obtido reconhecimento como modelos eficazes para a



O conceito de rastreabilidade de requisitos pode ser aplicado em qualquer modelo ágil. Por exemplo, no XP podem-se definir requisitos com histórias escritas em “cartões”



Os requisitos não funcionais podem ser mapeados em restrições, e os casos de testes podem ser representados pelos testes de aceitação, testes unitários, dados do teste e resultado do teste.

Rastreabilidade entre requisitos e casos de teste

O software deve ser testado para garantir que se comporta de acordo com os requisitos estabelecidos e que irá funcionar como esperado em seu ambiente pretendido. O teste deve ser eficaz para encontrar defeitos, mas também deve ser eficiente e rápido de executar a um baixo custo.

A automação da execução do teste de software pode reduzir significativamente o esforço necessário de teste, ou aumentar significativamente as condições a serem testadas em um tempo limitado. Alguns estudos mostram economia na ordem de 80% do esforço de teste manual com a utilização de teste automatizado.

Algumas organizações não obtiveram uma economia financeira ou de esforço diretamente com a automação do processo, mas isto, tem permitido produzir software de melhor qualidade mais rapidamente do que seria possível por meio de testes manuais isoladamente.

Implementar uma solução para automação de testes de software, mesmo com a disponibilidade de excelentes ferramentas de teste é uma tarefa não trivial, não evidente, com alta chance de insucesso e frustração. A elaboração de scripts deve ser precedida por uma análise da automação que determine qual a melhor abordagem a ser seguida de maneira a maximizar o retorno da atividade de teste.



pouca flexibilidade e produtividade, limitando os benefícios que poderiam ser obtidos com o processo de automação.



níveis diferentes, como mostrado na **Figura 5** “pirâmide de automação de teste”. Na base da pirâmide de teste estão os testes de unidade e componente.

No desenvolvimento ágil o maior esforço de teste é direcionado para esta camada. O meio da pirâmide inclui os testes automatizados da frente de negócios escritos para oferecer suporte à equipe.

Na camada do topo são mantidos os testes que manipulam e operam a camada de apresentação tradicionalmente os testes de maior custo para escrever.

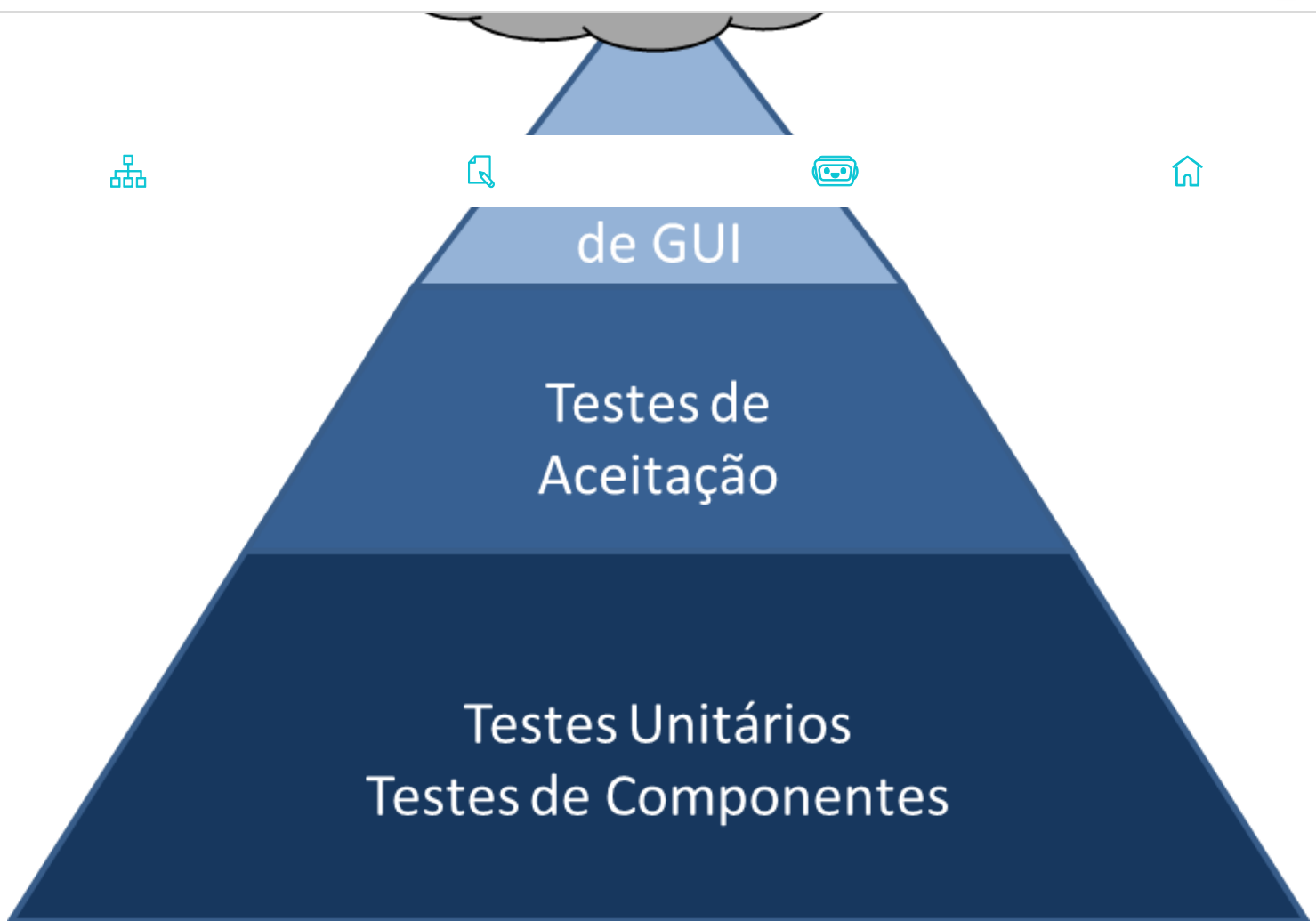


Figura 5. Pirâmide de automação de teste

Uma dificuldade na estratégia de automação de teste frequentemente está relacionada a como organizar os scripts de teste. É necessário planejar esta organização, pois na medida em que os scripts de teste aumentam, pode se tornar muito complicado localizar um caso de teste.

Considerando uma tecnologia orientada a objeto a unidade básica da organização do código de teste é o método de teste. Decidir o que colocar e onde é o foco central desta organização. Existem quatro possibilidades de organização, sendo que não existe uma solução ótima deve-se analisar e decidir a melhor estratégia:

1. Uma classe de teste para toda aplicação: o número de casos de teste pode crescer e será necessário dividir em mais classes de teste reduzindo o número de métodos de

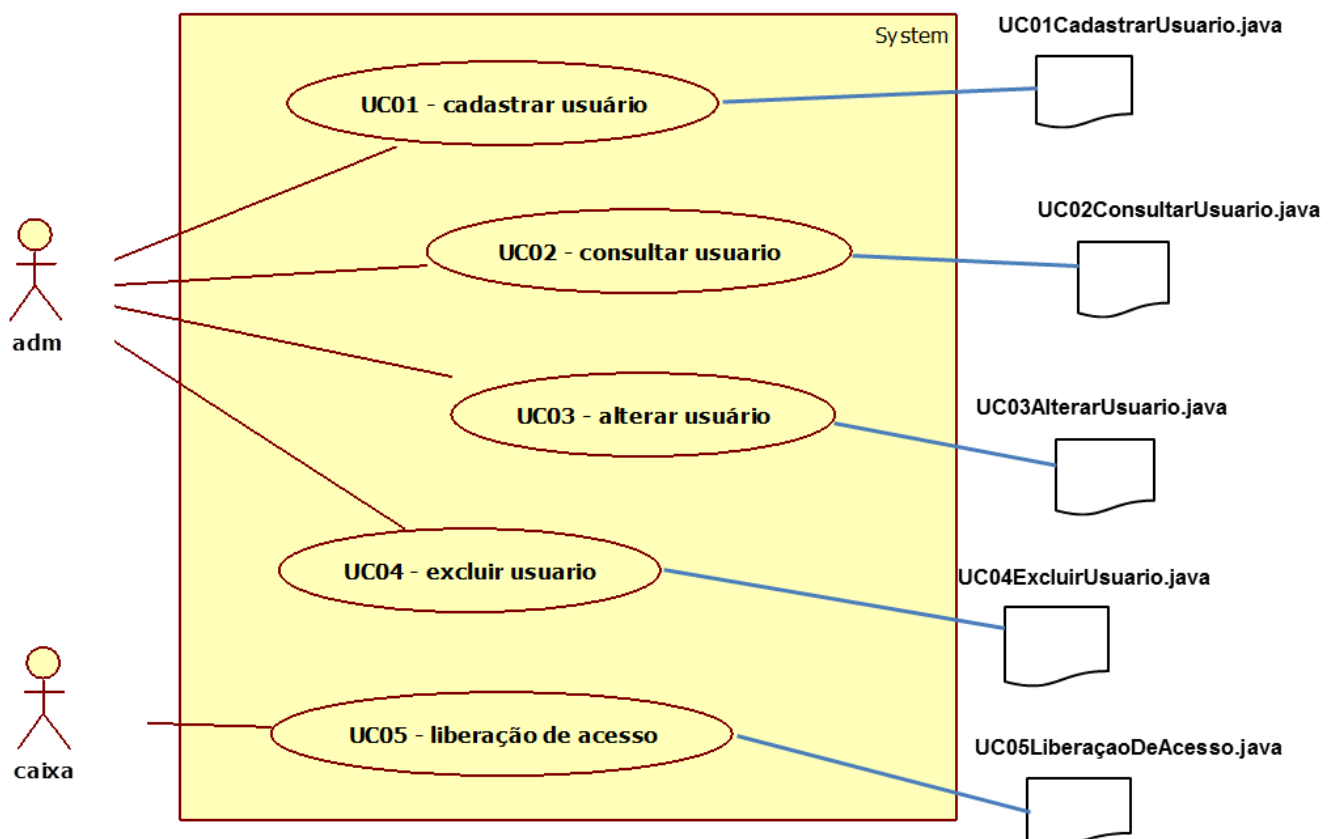
4. Uma classe de teste por pré-condições: um grupo de testes que requerem as mesmas pré-condições (Por exemplo: sistemas de processamento de cartão de crédito



bateria de teste, justifica o compartilhamento das pré-condições independentemente das funções a serem testadas).

Outro aspecto importante está relacionado à convenção de nomes. Os nomes estabelecidos para uma classe de teste ou um método de teste podem ajudar a encontrar e compreender um caso de teste.

A estratégia de organização dos scripts de teste associada a uma convenção de nomes adequada permite estabelecer os elos de rastreabilidade de acordo com as necessidades de cada projeto. No exemplo da **Figura 6**, a estratégia de um script de teste por caso de uso foi estabelecida e uma convenção de nomes permite identificar o caso de uso e o script de teste correspondente.





Req	Script deTeste	Casos de teste(métodos de teste)
		
UC05	UC05LiberacaoDeAcesso.java	CT02UC05A1LiberacaoDeAcesso_senha_invalida()
...

Tabela 1. Matriz de Rastreabilidade

Cada script de teste mantém os casos de teste de acordo com as condições de teste estabelecidas para validar o comportamento do sistema. A rastreabilidade é mantida por seção de caso de uso (fluxo básico e alternativo), conforme apresentado na **Figura 7**. Desta forma, de acordo com o critério de aceitação estabelecido, pode-se avaliar se o resultado da execução dos testes está aderente aos objetivos de qualidade planejados.

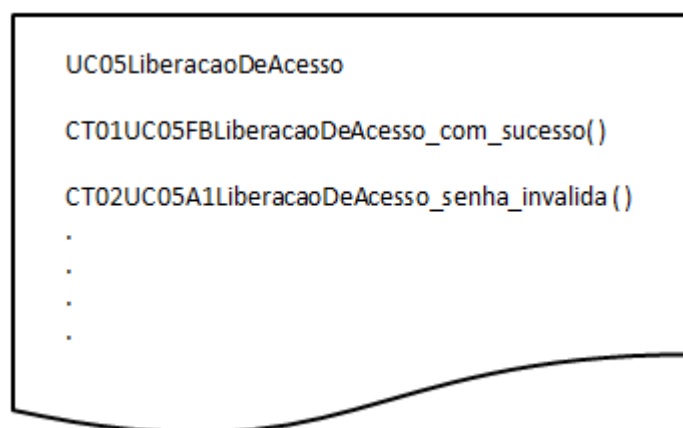


Figura 7. Casos de teste associados a casos de uso

A estratégia pode ser adaptada de acordo com as necessidades do projeto. O modelo pode ser aplicado para os testes de unidade, integração ou de sistema.

A existência dos mecanismos de rastreabilidade permite identificar as origens de cada

Mecanismos de rastreabilidade entre requisitos e casos de testes permitem, entre outras coisas, visualizar a quantidade de casos de teste alocada a um requisito e se algum requisito



aplicação de um modelo de rastreabilidade podem envolver:

1. A captura e uso dos elos (links) de rastreabilidade devem ser adaptados às necessidades específicas de cada projeto;
2. Definir no início do projeto um modelo de rastreabilidade, considerando a aplicação a ser desenvolvida, os objetos e artefatos que serão alvo de registro da rastreabilidade;
3. Identificar ferramentas que apoiarão o processo de rastreabilidade;
4. Conscientizar a equipe da importância do processo de rastreabilidade;
5. Estabelecer os momentos de registro e controle da rastreabilidade;
6. Avaliar um mecanismo de extração de elos (agilidade na recuperação);
7. Analisar criticamente com a equipe, após a liberação do software a efetividade do modelo de rastreabilidade adotado.

A rastreabilidade de requisitos tem sido identificada na literatura como fator de qualidade do processo de desenvolvimento de software, apesar disto, ainda existe um desconhecimento sobre os benefícios da utilização de modelos de rastreabilidade em organizações de desenvolvimento de software.

A efetiva aplicação da rastreabilidade no processo de desenvolvimento de software depende da definição de um modelo de rastreabilidade. Neste artigo abordamos a rastreabilidade de requisitos considerando aspectos de rastreabilidade entre requisitos e casos de teste.

O planejamento da rastreabilidade deve adotar uma abordagem que permita facilitar o processo de desenvolvimento de software e não restringi-lo. A existência dos mecanismos de rastreabilidade permite identificar as origens de cada funcionalidade de um sistema.

usuários. Mecanismos de rastreabilidade entre requisitos e casos de testes permite, entre outras coisas, visualizar a quantidade de casos de teste alocada a um requisito e se algum



Referências:

1. CRISPIN, Lisa; GREGORY, Janet. Agile testing: A practical guide for testers and agile teams. Pearson Education, 2009.
2. FEWSTER, Mark; GRAHAM, Dorothy. Software test automation: effective use of test execution tools. ACM Press/Addison-Wesley Publishing Co., 1999.
3. MESZAROS, Gerard. xUnit test patterns: Refactoring test code. Pearson Education, 2007.
4. BECK, Kent; ANDRES, Cynthia. Extreme programming explained: embrace change. Addison-Wesley Professional, 2004.
5. RUP. Rational Unified Process, 2001.
6. SAYÃO, Miriam; DO PRADO LEITE, Julio Cesar Sampaio. Rastreabilidade de requisitos. RITA, v. 13, n. 1, p. 57-86, 2006.
7. SPENCE, Ian; PROBASCO, Leslee. Traceability strategies for managing requirements with use cases. White Paper, Rational Software Corporation, 1998.

Tecnologias:

Engenharia de Software

Teste de Software

UML

Marcar como lido



Anotar



Por Devmedia
Em 2014

