Gestão de Testes

Ferramentas Open Source e melhores práticas na gestão de testes

Cristiano Caetano
c_caetano@hotmail.com
É certificado CBTS pela ALATS. Consultor de teste de software sênior com mais de 10 anos de experiência, já trabalhou na área de qualidade e teste de software para grandes empresas como Zero G, DELL e HP Invent. É colunista na área de Teste e Qualidade de software do site linhadecodigo.com.br e autor dos livros "CVS: Controle de Versões e Desenvolvimento Colaborativo de Software" e "Automação e Gerenciamento de Testes: Aumentando a Produtividade com as Principais Soluções Open Source e Gratuitas". Criador e mantenedor do portal TestExpert: A sua comunidade gratuita de teste e qualidade de software (www.testexpert.com.br).

Estima-se que o custo decorrente da correção de um bug cresce bastante à medida que ele é descoberto em fases mais avançadas no processo de desenvolvimento de software. No entanto, ainda existe uma forte tendência nas empresas em negligenciar essa realidade e não dedicar o tempo mínimo necessário para a realização das atividades de teste de software.

As atividades de teste são muitas vezes realizadas de maneira pouco estruturada ao final do projeto, quando não existe mais solução para os problemas. Segundo Pressman, a atividade de teste seria um dos elementos críticos da garantia da qualidade de software e pode assumir até 40% do esforço gasto em seu desenvolvimento.

A qualidade é um atributo do software que deve ser introduzida ao longo do processo de desenvolvimento do software, haja vista que ela não pode ser imposta depois que o produto tenha sido finalizado.

Glenford Myers, no seu livro "The Art of Software Testing", destaca que teste de software é o processo de executar um sistema com o objetivo de revelar falhas. No entanto, as atividades de teste de software não se resumem apenas a isso. Teste de software é uma atividade estruturada e sistemática baseada em técnicas, ferramentas e processos formais, como veremos ao longo deste artigo.

Validação e Verificação

A validação e verificação são atividades de apoio de um processo de garantia de qualidade de software. A motivação principal dessas atividades é prevenir e detectar os defeitos e minimizar os riscos do projeto.

Os defeitos podem ser introduzidos ao longo do processo de desenvolvimento do software. É necessário que eles sejam identificados o quanto antes dentro do processo de desenvolvimento, de preferência na própria fase onde foram inseridos, mas nem sempre isso acontece.

As atividades de validação e verificação são baseadas em técnicas de análise estática ou dinâmica dos artefatos (documentos, código fonte, código executável, etc) com o intuito de detectar os defeitos ou revelar falhas na própria fase onde eles foram inseridos ou em fases posteriores.

A verificação tem o objetivo de avaliar se o software está sendo desenvolvido conforme os padrões e metodologia estabelecidos no projeto. A verificação normalmente é realizada por meio da análise estática (revisões, inspeções, etc) dos artefatos (documentos, código fonte, etc) produzidos ao longo do processo de desenvolvimento do software.

A validação, por outro lado, tem o objetivo de avaliar a aderência, ou conformidade, do software implementado em relação ao comportamento descrito nos requisitos A validação normalmente é realizada por meio da análise dinâmica (execução de testes contra o código executável).

Níveis de teste

As atividades de testes são normalmente divididas em níveis. O nível de teste define, de certa forma, a fase do processo de desenvolvimento do software na qual os testes serão realizados. Existem quatro níveis de testes, como pode ser observado na Tabela 1.

Tipos de teste

Os tipos de teste normalmente são definidos em função das características ou dimensões da qualidade que serão avaliadas no software. A escolha da característica da qualidade de um software é, às vezes, um processo subjetivo. No entanto, essa escolha normalmente é realizada com base nos riscos associados a um problema causado por uma falha em uma dessas características.

A norma internacional ISO/IEC 9126, publicada em 1991 e que na versão brasileira de agosto de 1996 recebeu o número NBR 13596, define qualidade de software como "A totalidade de características de um produto de software que lhe confere a capacidade de satisfazer necessidades explícitas e implícitas".

Necessidades explícitas são as condições e objetivos propostos por aqueles que produzem o software. São, portanto fatores relativos à qualidade do processo de desenvolvimento do produto e são percebidos somente pelas pessoas que trabalharam no seu desenvolvimento.

As necessidades implícitas são necessidades subjetivas dos usuários (inclusive operadores, destinatários dos resultados do software e os mantenedores do produto). As necessidades implícitas são também chamadas de qualidade em uso e devem permitir a usuários atingir metas com efetividade, produtividade, segurança e satisfação em um contexto de uso especificado.

A ISO/IEC 9126 (NBR 13596) fornece um modelo de propósito geral que define seis amplas categorias de características de qualidade de software que são, por sua vez, subdivididas em sub-características, como pode ser observado na Tabela 2.

Conforme o que foi exposto anteriormente, a escolha do tipo de teste dependerá do grau de importância de cada uma das características de qualidade que serão avaliadas no software. Os tipos de testes mais comuns conforme o Guide to the CSTE Common Body of Knowledge do QAI são descritos resumidamente na Tabela 3. É importante destacar que não foram esgotadas todas as opções disponíveis, você poderá encontrar outros tipos de testes em outros artigos ou livros.

Nível de Teste	Descrição
Testes de unidade	Nesta fase são testadas as menores unidades de software desenvolvidas (por exemplo: métodos de uma classe).
Testes de integração	Nesta fase é testada a integração entre os componentes do sistema (por exemplo: classes, módulos, sub-sistemas, etc).
Testes de sistema	Nesta fase o sistema é testado como um todo com o objetivo de encontrar discordâncias entre o que foi implementado e o comportamento descrito nos requisitos.
Testes de aceitação	Nesta fase o sistema é testado como um todo com o objetivo de encontrar discordâncias entre o que foi implementado e o comportamento descrito nos requisitos, sob o ponto de vista das necessidades do usuário final.

Tabela 1. Níveis de teste.

Característica	Sub-características
Funcionalidade O conjunto de funções satisfaz as necessidades explícitas e implícitas para a finalidade a que se destina o produto?	Adequação Acurácia Interoperabilidade Segurança de acesso Conformidade
Confiabilidade O desempenho se mantém ao longo do tempo e em condições estabelecidas?	Maturidade Tolerância a falhas Recuperabilidade
Usabilidade É fácil utilizar o software?	Inteligibilidade Apreensibilidade Operacionalidade
Eficiência Os recursos e os tempos utilizados são compatíveis com o nível de desempenho requerido para o produto?	Comportamento em relação ao tempo Comportamento em relação aos recursos
Manutenibilidade Há facilidade para correções, atualizações e alterações?	Analisabilidade Modificabilidade Estabilidade Testabilidade
Portabilidade É possível utilizar o produto em diversas plataformas com pequeno esforço de adaptação?	Adaptabilidade Capacidade para ser instalado Capacidade para substituir Conformidade

Tabela 2. Categorias de características de qualidade de software da ISO/IEC 9126 (NBR 13596).

Técnicas de teste

Considerando o tamanho e a complexidade dos sistemas desenvolvidos na atualidade, podemos afirmar que é muito difícil executar testes que garantam 100% de cobertura de todos os requisitos ou linhas de código existentes. Somado a isso, prazos curtíssimos e orçamentos pequenos podem inviabilizar qualquer tentativa de alcançar 100% de cobertura.

Nessa condição, as técnicas de teste têm o objetivo de auxiliar os analistas de teste a identificar os casos de teste mais importantes. Ou seja, os casos de teste que exercitem a maior quantidade de linhas de código e garantam a maior cobertura possível, como pode ser observado na listagem:

- Teste Estrutural: nesta técnica, também conhecida como "Teste de Caixa Branca", são usados critérios para a geração de casos de teste com o objetivo de identificar defeitos nas estruturas internas do software.
- Teste Funcional: nesta técnica, também conhecida como "Teste de Caixa Preta", são usados critérios para a geração de casos de teste com o objetivo de avaliar a aderência, ou conformidade do software implementado em relação ao comportamento descrito nos requisitos.

Como você deve ter notado, as técnicas de teste avaliam o software sob pontos de vista diferentes. Dessa forma, podemos afirmar que as técnicas de teste são complementares e devem ser usadas em conjunto, em virtude de que elas identificam classes distintas de defeitos.

Modelo em "V" de teste de software

O modelo em "V" de teste de software é um dos modelos mais aceitos da atualidade. Este modelo enfatiza as atividades de validação e verificação com o intuito de prevenir/detectar defeitos e minimizar os riscos do projeto.

Para cada fase do processo de desenvolvimento do software, o modelo em "V" introduz uma fase, ou nível de teste correspondente. Neste modelo, o planejamento e a especificação dos testes ocorrem de cima para baixo, ou seja, ao longo das fases de desenvolvimento de software os testes são planejados e especificados. A execução dos testes ocorre no sentido inverso, como pode ser visto na (Figura 1).

Além disso, este modelo reforça o entendimento de que o teste não é uma fase que deve ser executada ao final do projeto, mas uma atividade que deve ser exercida ao longo do processo de desenvolvimento do software.

Estratégia de Testes

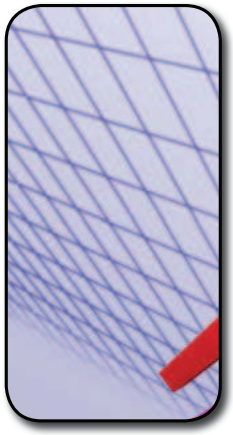
Os conceitos apresentados anteriormente são os pilares fundamentais para a elaboração de uma estratégia de testes. No livro "Teste de Software", Emerson Rios e Trayahu Moreira afirmam que durante a formulação da estratégia de testes devem ser levados em consideração diversos fatores, tais como: o porte e a importância do software, os seus requisitos, os prazos estabelecidos, o risco para o negócio, entre outros.

O diagrama da Figura 2 apresenta a relação entre as técnicas, tipos e níveis de testes que devem ser considerados durante a elaboração de uma estratégia de testes. Na Tabela 4 é apresentado um exemplo de uma estratégia de testes para uma aplicação web.

Perceba que neste exemplo hipotético houve uma atenção redobrada nos tipos de testes de segurança, usabilidade e desempenho em virtude de que essas características de qualidade são mais importantes em se tratando de aplicações web.

Tipo de teste	Descrição
Teste de Estresse	Avalia o desempenho do sistema com um volume de acesso/transações acima da média esperada e em condições extremas de uso.
Teste de Execução	Avalia se o sistema atende os requisitos de performance (proficiência) com um volume de acesso/transações dentro do esperado.
Teste de Contingência	Avalia se o sistema retorna a um status operacional após uma falha.
Teste de Operação	Avalia se o sistema (aplicação, pessoal, procedimentos e manuais) pode ser executado corretamente em ambiente de pré-produção.
Teste de Conformidade	Avalia se o sistema foi desenvolvido em consonância com os padrões e metodologia estabelecidos no projeto.
Teste de Segurança	Avalia se o sistema foi desenvolvido em consonância com os padrões de segurança da organização.
Teste de Regressão	Avalia por meio do re-teste se uma funcionalidade que estava funcionando ainda funciona após uma modificação no sistema.
Teste de Integração	Avalia se a interconexão entre as aplicações funciona corretamente.

Tabela 3. Tipos de teste.



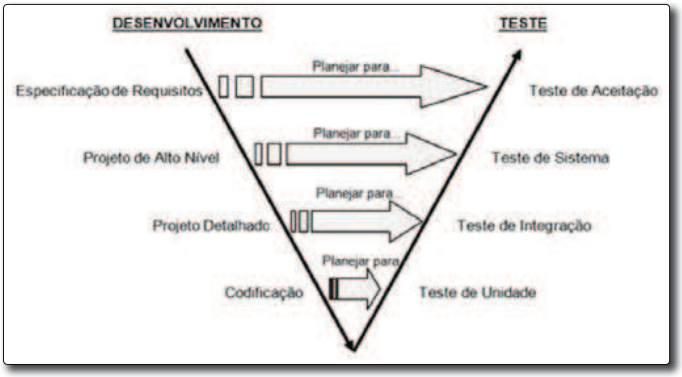


Figura 1. Modelo V descrevendo o paralelismo entre as atividades de desenvolvimento e teste de software (CRAIG e JASKIEL, 2002)

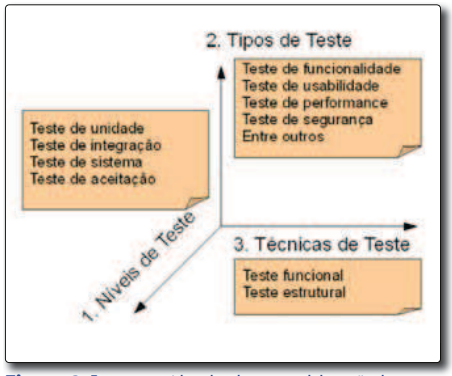


Figura 2. Fatores considerados durante a elaboração de uma estratégia de testes.

Processo estruturado de teste de software

Uma metodologia ou processo estruturado de teste de software tem a finalidade de formalizar as fases, atividades, papéis, artefatos e responsabilidades necessárias para o planejamento e a execução dos testes sistematicamente.

O TMAP (Testing Management Approach), ou abordagem estruturada de gestão de testes, é uma das metodologias mais populares da atualidade. Em virtude da sua simplicidade e foco no resultado, o TMAP é amplamente aceito pelo mercado.

O lema do TMAP é: "Entregar mais com menos. Mais rápido e melhor". O processo de gestão recomendado pelo TMAP é baseado em diversas práticas fundamentais e em um ciclo de vida estruturado de testes.

O ciclo de vida do TMAP (ver Figura 3) é bastante genérico e pode ser utilizado em diferentes tipos de projeto de teste de software. O ciclo de vida é dividido em sete fases distintas, como pode ser observado a seguir:

- Planejamento: Nesta fase é realizado o planejamento e a definição geral da estratégia e planos de testes.
- Controle: Nesta fase são realizados o controle e a monitoração das atividades planejadas.
- Configuração e manutenção da infra-estrutura: Nesta fase é preparada e mantida a infra-estrutura (software e hardware) necessária para a plena realização dos testes.
- Preparação: Nesta fase é realizado o refinamento da estratégia de testes e plano de testes criados na fase de Planejamento.
- Especificação: Nesta fase é realizada a especificação dos casos de testes e demais documentos.
- Execução: Nesta fase é realizada a execução dos testes, reporte do progresso e indicadores de qualidade.
- Conclusão: Nesta fase o processo de teste é avaliado a fim de promover as melhorias para os próximos projetos.

Documentação das atividades de testes

Uma metodologia ou processo estruturado de teste exige um conjunto mínimo de documentos padronizados para documentar os planos e estratégias, assim como, o relato do progresso das atividades de testes.

Níveis de Teste	Teste Funcional	Teste Estrutural
	Tipos de Teste	
Teste de unidade		Teste Unitário Teste de Conformidade
Teste de integração	Teste de Integração	Teste de Estresse Teste de Execução Teste de Segurança
Teste de sistema	Teste de Requisitos Teste de Regressão Teste de Usabilidade	Teste de Estresse Teste de Segurança Teste de Execução
Teste de aceitação	Teste de Aceitação Teste de Usabilidade	Teste de Contingência Teste de Operação

Tabela 4. Exemplo da formulação de uma estratégia de testes.

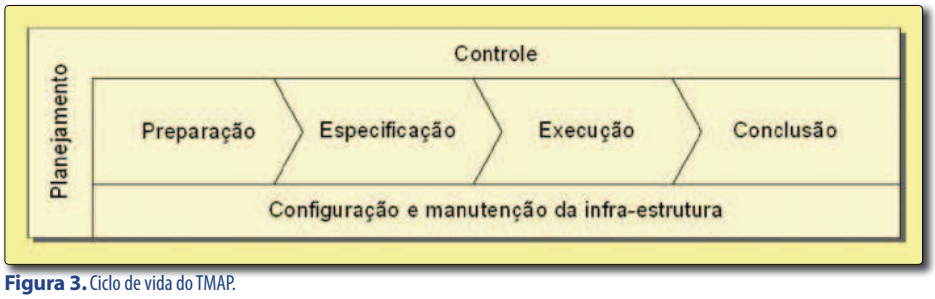
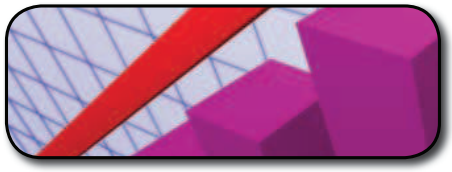
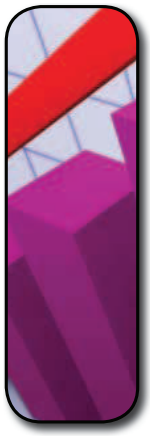


Figura 3. Ciclo de vida do TMAP.

O padrão IEEE Std 829-1998 (IEEE Standard for Software Test Documentation) define a documentação e relatórios necessários para a execução de um projeto de teste de software. Os oito documentos essenciais deste padrão são:

- Plano de Teste: Define o planejamento para execução do teste, incluindo a abrangência, abordagem, recursos e cronograma das atividades de teste. Identifica os itens e funcionalidades a serem testados, as características dos itens a serem testados, as tarefas a serem realizadas e os riscos associados com a atividade de teste.
- Especificação do Projeto de Teste: Refina a abordagem apresentada no Plano de Teste e identifica as funcionalidades e características a serem testadas pelo projeto e pelos seus testes associados. Também identifica os casos e os procedimentos de teste, se existirem, e apresenta os critérios de aprovação para esses elementos.
- Especificação dos Casos de Teste: Define os casos de teste, incluindo as pré-condições, passos, resultados esperados e condições gerais para a execução do teste.
- Especificação dos Procedimentos de Teste: Define a seqüência de ações necessárias para executar um conjunto de casos de teste.
- Relatório de Encaminhamento de Item de Teste: Identifica os itens encaminhados para teste, no caso da existência de times distintos responsáveis pelas tarefas de desenvolvimento e de teste.
- Log de Testes: Documenta os registros cronológicos dos fatos relevantes durante a execução dos testes.
- Relatório de Incidente de Testes: Documenta qualquer evento que ocorra durante a atividade de teste e que necessite de alguma análise posterior.
- Relatório Sumário de Testes: Documenta de forma resumida os resultados das atividades de teste associadas com uma ou mais especificações de projeto de teste e fornece avaliações baseadas nesses resultados.





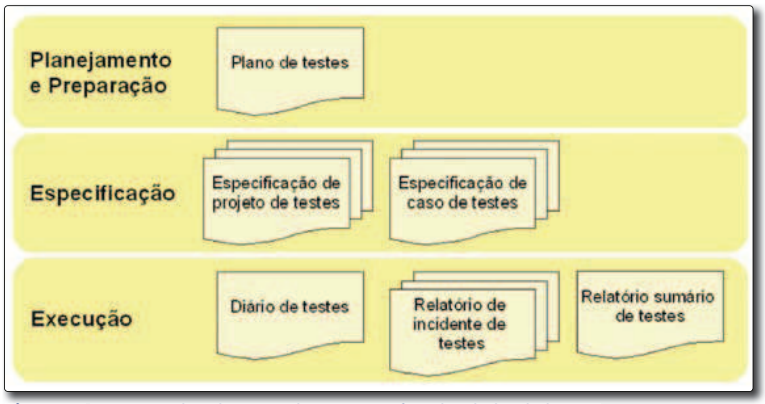


Figura 4. Documentos do padrão IEEE Std 829-1998 nas fases do ciclo de vida do TMAP.

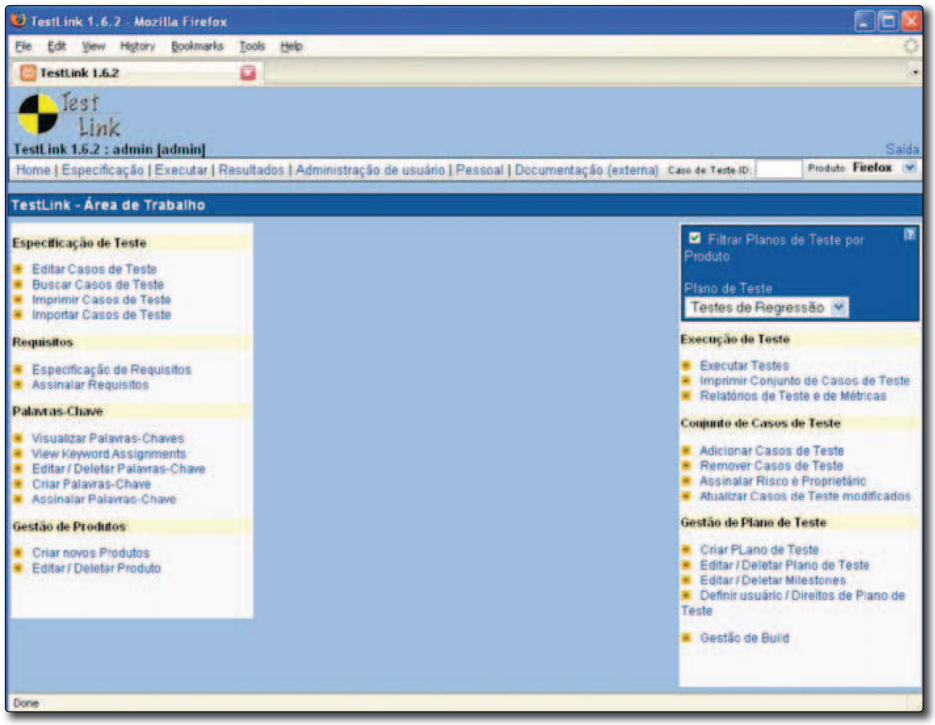


Figura 5. Página principal do TestLink.

Na Figura 4 é apresentada uma sugestão de utilização dos documentos do padrão IEEE Std 829-1998 nas fases do ciclo de vida do TMAP (Testing Management Approach).

TestLink

TestLink é uma ferramenta open source automatizada escrita em PHP cujo principal objetivo é dar suporte às atividades de gestão de testes, como pode ser observado no exemplo apresentado na Figura 5.

Você poderá criar planos de testes e gerar relatórios com diversas métricas para o acompanhamento da execução dos testes. TestLink oferece um recurso para que você possa registrar e organizar os requisitos do projeto, assim como, associar os casos de teste aos requisitos. Dessa forma, você poderá garantir o rastreamento entre os requisitos e os casos de teste utilizando uma matriz de rastreabilidade.

O endereço do site da ferramenta TestLink está disponível na seção Links. Caso você tenha interesse em conhecê-la com maior profundidade, os passos para a instalação são:

1. As pré-condições para a instalação são (PHP 4.0.6 ou superior, MySQL 3.23.2 ou superior, Apache);
2. Faça o download do TestLink no site disponível na seção Links;
   - Descompacte o arquivo zip na pasta www ou htdocs do servidor WEB (Apache).
3. Abra o seu navegador e acesse o seguinte endereço: (http://localhost/testlink-1.6.2/install/index.php);
4. Na janela de instalação, selecione a opção New installation;
5. Na janela TestLink Setup, preencha o campo login com "root" e o campo password com a senha de acesso ao seu banco de dados MYSQL;
6. Deixe os valores default nos demais campos;
7. Pressione o botão "Setup TestLink";
8. Abra o seu navegador e acesse o seguinte endereço: (http://localhost/testlink-1.6.2/login.php);
9. Faça o login com o usuário padrão (admin/admin). Lembre-se de mudar a senha deste usuário.

TestLink é instalado em inglês por padrão. Caso você prefira utilizar TestLink com os textos traduzidos para a língua portuguesa, então siga os passos:

1. Faça o login normalmente com o seu usuário e senha;
2. Clique no menu "Personal";
3. No campo Locale, selecione a opção "Portuguese (Brazil)";
4. Pressione o Botão "Update".

Entre as diversas funcionalidades oferecidas por TestLink, devemos destacar:

- Controle de acesso e níveis de permissões por papéis (líder, testador, etc);
- Os casos de testes são organizados hierarquicamente em suítes;
- Os casos de testes podem ser classificados por palavras-chave "keywords" para facilitar a pesquisa e organização;
- Criação ilimitada de projetos e casos de teste;
- Os planos de testes podem ser priorizados e atribuídos aos testadores;
- Gerador interno de relatórios e gráficos (possibilidade para exportar os dados nos formatos CSV, Excel e Word);
- Integração com ferramentas de gestão de defeitos (Bugzilla, Mantis, Jira).

A fim de dar ao leitor uma exposição sobre as principais funcionalidades da ferramenta TestLink e como elas auxiliam no planejamento execução dos testes, vamos simular na prática o cadastro de um caso de teste e acompanhar todos os passos até a sua execução.

No nosso cenário, vamos simular o cadastro e execução de um caso de teste para o gerenciador de temas do

navegador Firefox. Este cenário é baseado em um dos casos de testes disponibilizados pelo time de Quality Assurance do Firefox. Para saber mais, visite o site: http://litmus.mozilla.org/.

Primeiro, devemos criar um projeto em TestLink. No nosso exemplo, foi criado um projeto hipotético chamado "Firefox". Tão logo o projeto seja criado, devemos criar os requisitos. Para realizar tal tarefa, acesse o menu "Requisitos → Especificação de Requisitos".

Nesta janela você poderá criar os requisitos. No nosso exemplo, foi criada uma especificação de requisito para o recurso "Add-On". Para este recurso foram criados os requisitos "Extensions" e "Themes", como pode ser visto no exemplo apresentado na Figura 6.

Assim que os requisitos forem criados, você estará apto para criar os casos de teste. Antes, no entanto, você deverá criar um "Componente" e uma "Categoria". A arquitetura de TestLink exige que os testes sejam organizados em componentes e categorias. Para o nosso exemplo, foi criado um "Componente" chamado "Add-Ons" e uma "Categoria" chamada "Smoke Tests".

Satisfeitas as pré-condições de TestLink, o próximo passo é a criação do caso de teste. Para isto, você deverá acessar o menu "Especificação", selecionar a "Categoria" desejada e então clicar no botão "Criar Casos de Teste".

Na janela de cadastro de caso de teste, você deverá informar o sumário, os passos e os resultados esperados, como pode ser observado na Figura 7. Assim que o teste for cadastrado, você poderá criar uma matriz de rastreabilidade bidirecional entre os requisitos através da opção de menu "Requisitos → Assinalar Requisitos".

TestLink também oferece uma funcionalidade para imprimir um documento com todos os dados fornecidos durante a criação do "Componente", "Categoria" e caso de teste. Este documento é muito parecido com os documentos propostos pelo padrão IEEE Std 829-1998 (IEEE Standard for Software Test Documentation), como pode ser visto no exemplo da Figura 8.

TestLink permite que os casos de teste sejam agrupados em Planos de Teste (ou Teste Suítes). Para criar um Plano de

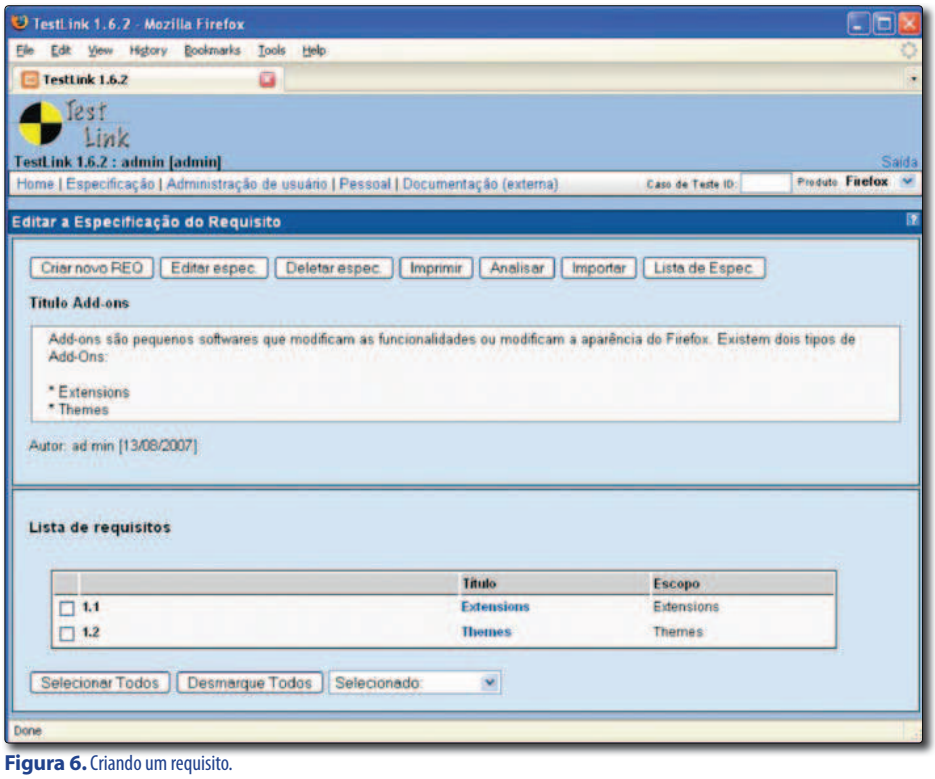


Figura 6. Criando um requisito.

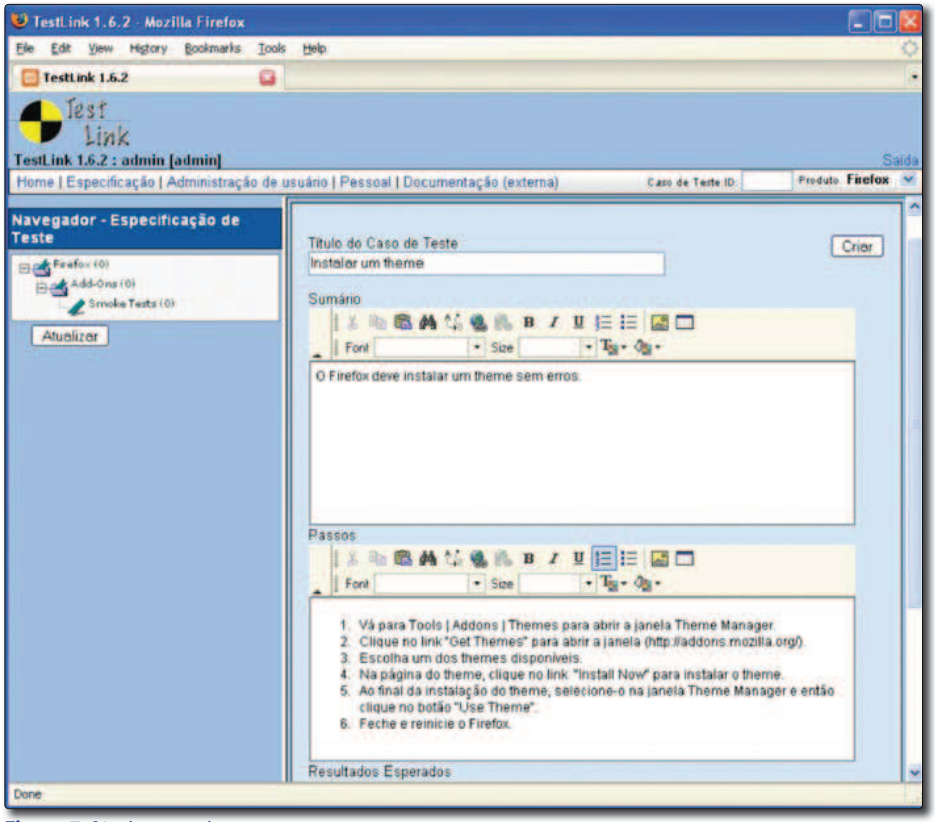


Figura 7. Criando um caso de teste.

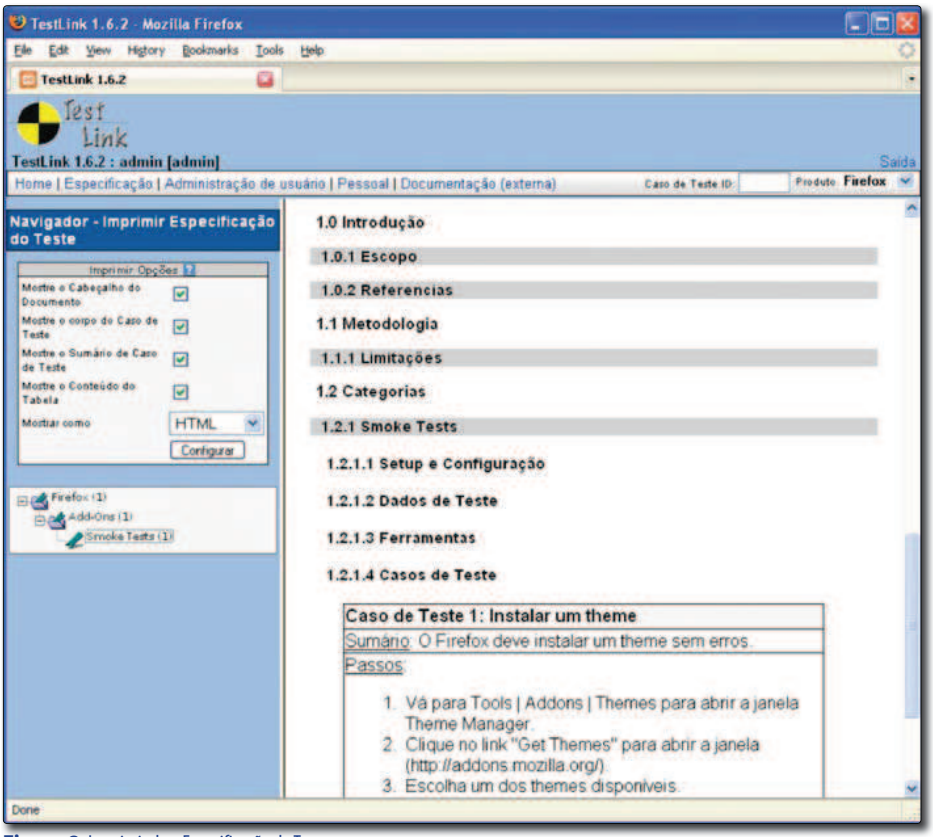


Figura 8. Imprimindo a Especificação do Teste.

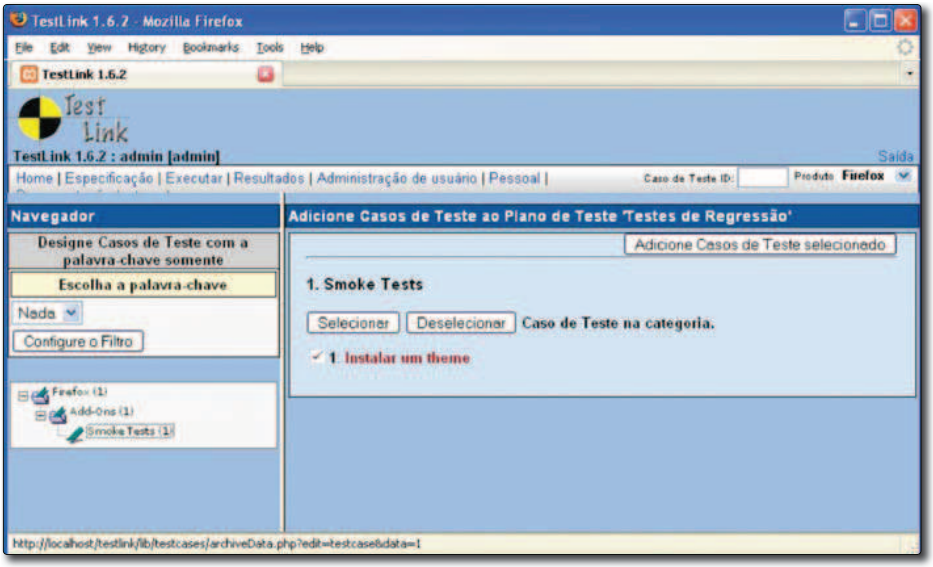
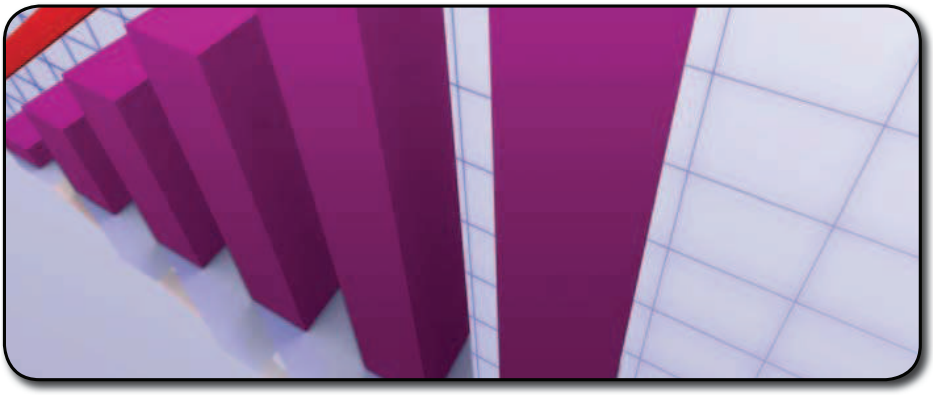


Figura 9. Associando um caso de teste a um Plano de Teste.



Teste, você deverá acessar o menu "Gestão de Plano de Teste → Criar Plano de Teste". Para o nosso exemplo, foi criado o Plano de Teste chamado "Testes de Regressão".

Tão logo o Plano de Teste seja criado, você poderá associá-lo aos casos de testes na janela "Adicione Casos de Teste ao Plano de Teste" por meio da opção de menu "Conjunto de Casos de Teste → Adicionar Casos de Teste", como pode ser observado na Figura 9.

Antes de começar a execução dos testes, você deverá associar os Planos de Testes aos testadores responsáveis por cada área ou funcionalidade do sistema. Para realizar tal tarefa, você deverá abrir a janela "Assinalar usuários para o Plano de Teste" acessando o menu "Gestão de Plano de Teste → Definir usuário/Direitos de Plano de Teste" e associar o Plano de Teste aos testadores, como pode ser visto no exemplo apresentado na Figura 10.

Para iniciar a execução dos testes, o testador deverá selecionar o Plano de Teste associado a ele e então ir para a janela "Execução dos casos de teste" por meio do menu "Execução de Teste > Executar Testes". Nesta janela, o testador poderá executar todos os testes existentes no Plano de Teste.

Para cada caso de teste, são exibidos os passos a serem executados e os resultados esperados. Ao final da execução do teste, o testador poderá relatar alguma nota ou comentário, assim como, o resultado do teste (Não executado, Passado/Verificado, Com falha ou Bloqueado), como pode ser observado na Figura 11.

É importante destacar que a arquitetura de TestLink exige que a execução dos testes seja associada a um Build. Um Build representa uma versão ou liberação do sistema. Para criar um Build, você deverá abrir a janela "Gestão de Build" acessando o menu "Gestão de Plano de Teste → Gestão de Build".

Durante ou ao final da execução dos testes, você poderá acompanhar o progresso através da geração de relatórios gerenciais. Estes relatórios apresentam a situação da execução dos testes, assim como, métricas importantes para determinar a qualidade do sistema, ou melhor, do Build atual do sistema. Entre os relatórios existentes, podemos destacar:

- Métricas gerais do Plano de Teste: Apresenta as métricas dos testes agrupados por prioridade, componente, testador, entre outros;
- Status Geral do Build: Neste relatório é apresentado o sumário dos testes planejados em relação aos executados, os testes que falharam, os que passaram, etc;
- Relatórios de Testes baseados nos requisitos: Apresenta os requisitos cobertos pelos testes e o sumário da execução agrupado por Especificação de Requisitos.

Para visualizar os relatórios, você deverá abrir a janela "Relatórios de Teste e de Métricas" clicando em "Execução de Teste > Relatórios de Teste e de Métricas", como pode ser visto na Figura 12.

Conclusão

Neste artigo foram apresentados os conceitos e algumas das melhores práticas na gestão de testes. O objetivo principal do artigo era reforçar o entendimento de que o teste não é uma fase que deve ser executada ao final do projeto, mas uma atividade que deve ser exercida ao longo do processo de desenvolvimento do software.

Teste de software é uma atividade cara que exige tempo, planejamento, conhecimento técnico, infra-estrutura e comprometimento. Por maior e mais organizado que seja o esforço para a realização das atividades de testes, é impossível garantir 100% de cobertura de todos os requisitos ou linhas de código existentes no sistema.

Edsger Dijkstra sabiamente resumiu este fato com a seguinte frase: "Testes podem mostrar a presença de erros, mas não a sua ausência". Dessa forma, está em suas mãos a responsabilidade de conhecer e aplicar as melhores práticas na gestão de testes para garantir a maior cobertura com o mínimo de esforço.

Ao longo desse artigo foram apresentadas as principais funcionalidades de TestLink, ferramenta Open Source para gestão de testes. No entanto, caso TestLink não atenda suas necessidades, são apresentados na Tabela 5 algumas alternativas comerciais e open source. Não foram esgotadas as opções disponíveis, mas já é um bom ponto de partida para auxiliar o leitor a encontrar a solução ideal para a sua necessidade. ●

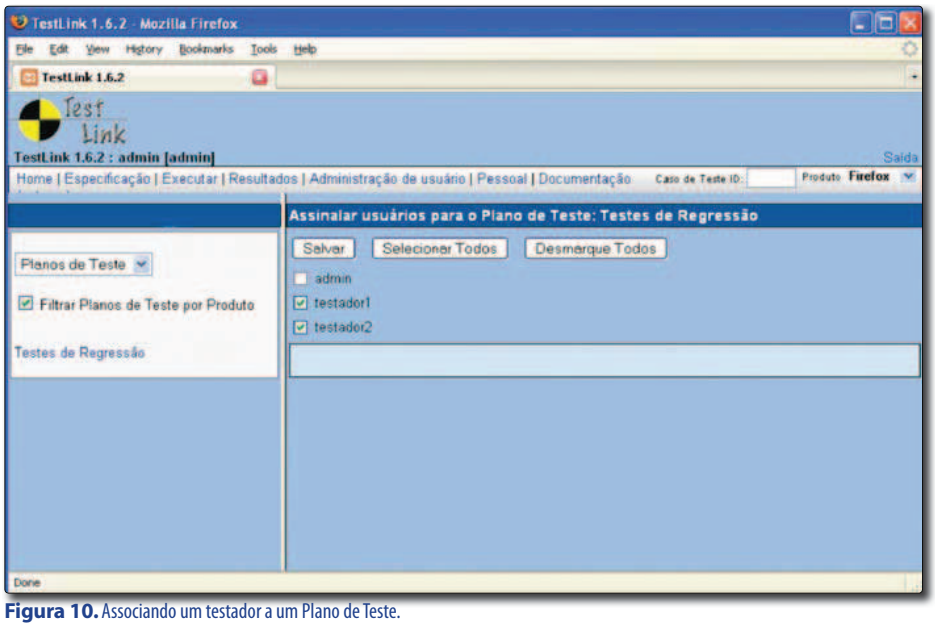


Figura 10. Associando um testador a um Plano de Teste.

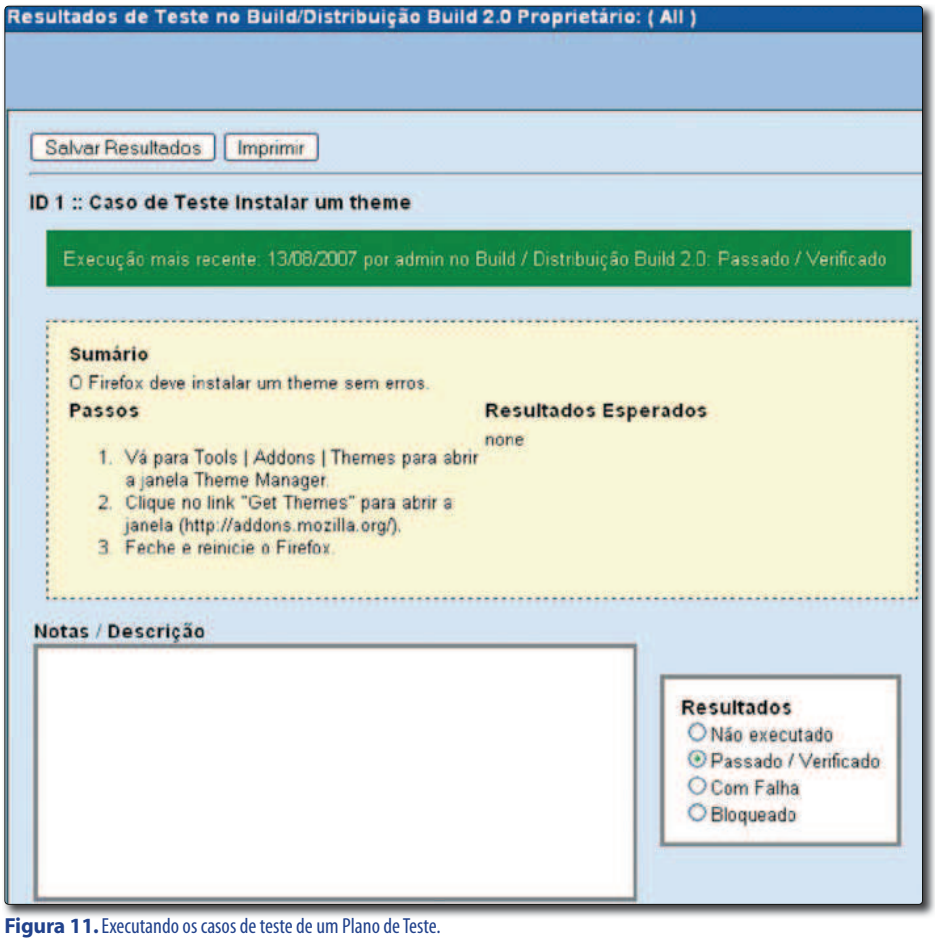


Figura 11. Executando os casos de teste de um Plano de Teste.

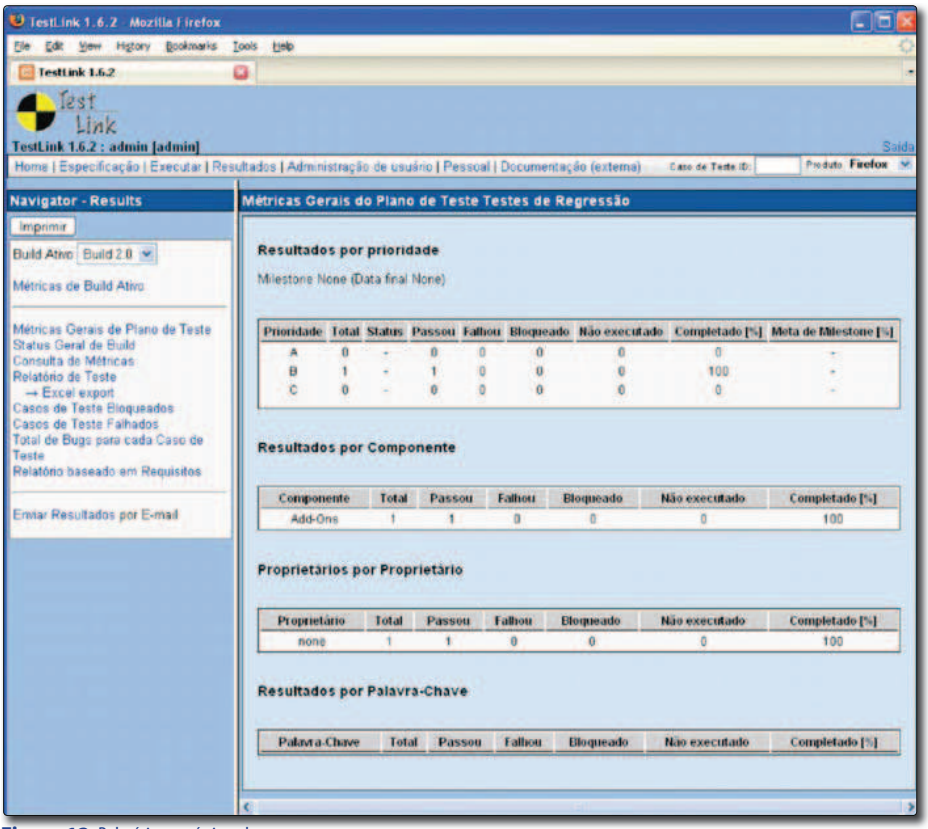


Figura 12. Relatórios e métricas de teste.

Dê seu feedback sobre esta edição!

A Engenharia de Software Magazine tem que ser feita ao seu gosto. Para isso, precisamos saber o que você, leitor, acha da revista!

Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/esmag/feedback

Links

Site do TestLink
http://www.teamst.org/

Norma ISO/IEC 9126
http://pt.wikipedia.org/wiki/ISO_9126

Modelo FURPS
http://en.wikipedia.org/wiki/Furps

IEEE Std 829-1998 - Standard for Software Test Documentation
http://en.wikipedia.org/wiki/IEEE_829

RUP – Rational Unified Process
http://pt.wikipedia.org/wiki/Rational_Unified_Process

Qualidade de Software – Uma Necessidade
http://www.fazenda.gov.br/ucp/pnafe/cst/arquivos/Qualidade_de_Soft.pdf

Open Source	Comercial
rth http://www.rth-is-quality.com	**RSI/QA-Teste** http://www.rsinet.com.br/modules/content/index.php?id=9
TestMaster http://testmaster.sourceforge.net/	**TestLog** http://www.testlog.com/
Testitool http://majordojo.com/testitool/	**Rational** TestManager http://www-306.ibm.com/software/awdtools/test/manager/
Testopia http://www.mozilla.org/projects/testopia/	**Mercury Quality Center** http://www.mercury.com/us/products/quality-center/

Tabela 5. Ferramentas de gestão de testes alternativas.