Centro de Tecnologia da Informação Renato Archer, CTI

Ministério da Ciência e Tecnologia, MCT

Guia para Projetar o Teste de Software

Autores:

Adalberto Nobiato Crespo: Adalberto.crespo@cti.gov.br
Mario Jino: Jino@dca.fee.unicamp.br
Miguel Argollo: Miguel.argollo@cti.gov.br
Paulo M. S. Bueno: Paulo.bueno@cti.gov.br
Celso P. Barros: Celso.barros@cti.gov.br

Dezembro/2010

Guia para Projetar o Teste de Software

Apresentação

Este documento apresenta um guia para o projeto do teste de software. O guia contempla as diretrizes para o projeto do teste, quem deve projetar e utilizar os testes projetados, e quais os requisitos de inicio e término do projeto de teste. Além disso, apresenta detalhadamente as atividades para o projeto do teste de um software, identificando as informações necessárias para a execução de cada atividade e as fontes dessas informações. Apresenta também os resultados alcançados com a execução de cada atividade. Exemplos para uma melhor compreensão de algumas atividades são apresentados. Critérios estruturais e funcionais para a geração de casos de teste são introduzidos conceitualmente no documento **Teste de Software: Motivação e Conceitos Básicos**. Neste documento alguns critérios são utilizados para ilustrar o projeto de casos de teste.

Este documento é parte integrante da Estrutura Tecnológica de Teste de Software do SPB – Software Público Brasileiro. Destina-se às comunidades do SPB interessadas em teste com o objetivo de aprimorar a qualidade dos produtos de software.

O Projeto de Teste é um dos subprocessos do processo genérico de teste descrito no documento **Modelo de Processo Genérico de Teste de Software**.

Outros guias e tutoriais de teste complementam a Estrutura Tecnológica de Teste.

Sumário

1 Introdução

Como parte do processo de produção de software, o teste requer uma preparação cuidadosa que precede sua execução, bem como o registro adequado de seus resultados. A preparação deve incluir o planejamento e o projeto do teste; deverão estar definidos como serão registrados os resultados do teste e também como será gerenciado o processo do teste. Após a preparação ocorrem a execução do teste e o registro dos resultados, observando-se as concordâncias e discordâncias entre os resultados esperados e os obtidos, bem como eventuais incidentes observados.

Dificilmente um projetista de software passa dos requisitos de um sistema diretamente para a fase de codificação sem dedicar algum tempo organizando estes requisitos em conjuntos coerentes, priorizando estes requisitos segundo suas importâncias e definindo uma arquitetura que reflita coerentemente estes requisitos. O tempo e os recursos gastos nestas atividades, que compõem o projeto de software, são amplamente compensados na fase de implementação.

De forma semelhante, o responsável pelo teste de um sistema deve passar por uma fase de projeto de teste que tenha como objetivo organizar seu trabalho e definir os casos e procedimentos de teste antes de partir para sua execução.

2 Projeto do Teste de Software

O **projeto de teste** é um subprocesso que tem como propósito principal gerar **casos de teste** a partir dos requisitos do software[1] e do **objetivo do teste**, definido durante o planejamento do teste. O **objetivo do teste** identifica a razão ou o propósito para se projetar e executar um determinado teste.

O projeto de teste é fundamental para a obtenção de um conjunto de casos de teste que exercite adequadamente os diversos elementos do software em teste e, desta forma, forneça uma boa avaliação da qualidade deste software.

A importância da atividade de projeto do teste fica mais evidente quando se considera que o teste exaustivo é impossível de ser realizado na prática e, portanto deve-se buscar a obtenção de um subconjunto de casos de teste que seja o mais representativo possível e que tenha a capacidade de detectar o maior número possível de defeitos.

Uma característica desejável de um conjunto de casos de teste é uma alta **cobertura** – o grau, expresso em termos percentuais, em que um determinado tipo de elemento do software foi exercitado pelo conjunto de teste.

Exemplos de cobertura:

- Um conjunto de casos de teste tem uma cobertura de 80% dos comandos - 80% dos comandos executáveis foram exercitados pelo conjunto de teste.
- Um conjunto de teste tem uma cobertura de 75% das classes de equivalência – 75% das classes de equivalência foram exercitadas pelo conjunto de teste.

Uma cobertura adequada pode ser obtida pelo uso consistente de técnicas e critérios de teste, que direcionam o projetista de teste na definição dos casos de teste. As técnicas de teste podem ser divididas em duas grandes famílias – técnicas funcionais e técnicas estruturais, conforme apresentado no documento **Teste de Software: Motivação e Conceitos Básicos**.

Quando o projetista de teste emprega um critério da técnica estrutural ele se baseia no código fonte do software para gerar os casos de teste, o que torna esta técnica mais conveniente para ser aplicada por programadores na fase de teste de unidade, uma vez que são eles que detêm o conhecimento necessário do código fonte sendo testado.

Quando o projetista de teste emprega um critério da técnica funcional ele se baseia nos requisitos do software para gerar os casos de teste. Desta forma, os critérios da técnica funcional não precisam ser aplicados exclusivamente por programadores, uma vez que o conhecimento e o acesso ao código fonte não são necessários.

Cada técnica e cada critério de teste possuem foco e premissas próprios. Por exemplo, quando empregamos o critério Particionamento de Equivalência para a geração de casos de teste o foco é a divisão dos valores das variáveis de entrada de um software em classes tais que, de acordo com a especificação, os elementos de uma classe levem ao mesmo comportamento do software. A premissa é que o teste com um valor ou elemento de uma classe é equivalente ao teste de qualquer outro elemento da mesma classe. Adotar um determinado critério aumenta a possibilidade de que defeitos de determinado tipo sejam detectados.

Conhecer as técnicas e critérios existentes e saber identificar o critério mais adequado a uma situação específica são atribuições básicas de um bom projetista de teste.

1 Neste Guia, os requisitos do sistema também englobam as normas e procedimentos padrões da organização bem como a legislação pertinente.

O seguinte exemplo é utilizado para ilustrar conceitos gerais bem como a aplicação de diversos critérios de teste das técnicas funcional e estrutural, descritos a partir da seção 4. Embora simples, espera-se que ele caracterize situações típicas vivenciadas por um projetista de teste.

Exemplo: Sistema de Cálculo de Seguro de Automóveis

Um sistema de gerência de seguros de automóveis recebe como entrada o nome, a idade e gênero de um motorista e calcula o desconto ou acréscimo a ser aplicado ao valor base do seguro de seu automóvel da seguinte forma:

1. Motoristas do sexo feminino com idade menor do que ou igual a 65 anos têm um desconto de 10%.
2. Motoristas do sexo masculino com menos de 30 anos pagam um adicional de 15% sobre o valor do seguro base;
3. Motoristas com mais de 65 anos pagam um adicional de 5% sobre o valor do seguro base;
4. Motoristas com mais de 100 anos não são aceitos pela companhia de seguros.

Além disso, as seguintes restrições de preenchimento do formulário devem ser respeitadas:

1. O campo de entrada de dados para a variável “nome” deve aceitar no máximo 40 caracteres alfabéticos;
2. O campo de entrada de dados para a variável “idade” deve receber no máximo três caracteres;
3. O sistema deve desconsiderar espaços em branco digitados no início ou final dos campos de entrada de dados;
4. Os campos “nome”, “idade” e “sexo” são de preenchimento obrigatório;
5. O campo “sexo” deve ser preenchido apenas com os valores “M” (masculino) ou “F” (feminino).

Embora o cálculo do valor do seguro em um sistema real seja muito mais complexo e dependa de outros fatores, o nível de complexidade apresentado é suficiente para os propósitos deste documento.

Conforme descrito no documento **Teste de Software: Motivação e Conceitos Básicos**, um caso de teste é composto por um conjunto de dados de entrada a serem submetidos ao programa em teste e por um conjunto de resultados esperados; uma coleção de vários casos de teste é referida como um Conjunto de Casos de Teste.

Exemplos de casos de teste:

Dados de Entrada	Nome	José da Silva
	Sexo	M
	Idade	28
Resultado Esperado	Valor do seguro	Adicional de 15%

Os dados de entrada deste exemplo de caso de teste são o nome do motorista (José da Silva), seu sexo (masculino) e sua idade (28 anos); o resultado esperado deste caso de teste, ou seja, o resultado que o sistema deve gerar para os dados de entrada fornecidos, é uma taxa adicional de 15% a ser paga sobre o valor básico do seguro do carro deste motorista.

Este caso de teste foi gerado sem o conhecimento do correspondente código-fonte: casos de teste podem (na realidade, devem) ser gerados antes que o software esteja disponível para a execução dos testes; de fato, todo o conhecimento necessário para a definição do caso de teste estava contido nos requisitos apresentados no exemplo fornecido.

Outro exemplo de um caso de teste é apresentado a seguir:

Dados de Entrada	Nome	Maria Soares
	Sexo	F
	Idade	- 10
Resultado Esperado	Mensagem	Idade inválida

Este caso de teste avalia a capacidade de o software detectar e tratar corretamente um dado de entrada inválido – no caso, uma idade negativa. Este exemplo é interessante por dois motivos: em primeiro lugar nos lembra que os testes devem verificar o funcionamento do software tanto para dados de entrada válidos como para dados de entrada inválidos.

Em segundo lugar, é interessante perceber que nenhum requisito do software indica explicitamente que o valor das idades dos motoristas deve ser positivo – em princípio, um requisito óbvio, que muitos analistas de software não consideram necessário explicitar. Entretanto, a experiência nos mostra que por trás de requisitos óbvios pode existir uma implementação defeituosa, e que é função do projetista de teste avaliar tais situações. Este exemplo indica também a necessidade do projetista de teste conhecer a área de aplicação do software que está sendo testado, e não basear seu trabalho somente nos requisitos documentados.

2.1 Diretrizes para o Projeto do Teste

Os passos descritos neste documento estabelecem uma orientação geral para o projeto do teste. No entanto, caso haja necessidade, esses passos podem ser modificados para se adequarem às situações particulares de uma organização.

Além disso, dependendo da área de aplicação, do objetivo do teste ou da fase de teste, os passos podem ser adaptados, estendidos ou reduzidos de modo a produzir um maior ou menor detalhamento do teste.

É fundamental que o projeto do teste esteja alinhado à realidade da organização em relação à política de teste, aos recursos disponíveis para o teste e aos padrões da organização.

2.2 Quem Projeta o Teste do Software

Projetista de teste: tem como responsabilidade projetar os testes a partir dos requisitos do software (como visto anteriormente, normas e procedimentos da empresa e a legislação pertinente também devem ser levados em consideração pelo projetista) e das informações contidas no Plano de Teste, especialmente as contidas na abordagem de teste. O projetista deve conhecer as técnicas e critérios para a geração de casos de teste e escolher as mais adequadas em cada situação. É fortemente

aconselhável que ele não faça parte das equipes de análise ou de desenvolvimento do software, e que conheça a área de aplicação do software sendo testado. Entretanto, em muitas organizações pequenas, que estejam consolidando uma equipe de teste, este papel é desempenhado por um analista de software.

2.3 Quem Utiliza o Projeto do Teste de Software

Gerente de teste: tem como responsabilidade acompanhar o projeto dos testes; deve garantir a comunicação necessária entre o projetista dos testes e os demais envolvidos (equipe de análise, equipe de desenvolvimento, representantes dos usuários, etc). É responsável também pela aprovação dos testes projetados, devendo envolver todos os que possam contribuir nesta aprovação.

Testador: responsável por executar e registrar os testes projetados, avaliar os resultados e relatar os incidentes encontrados.

Analista de software: pode analisar os testes projetados para identificar situações de uso não previstas no projeto do software com o objetivo de aprimorá-lo.

2.4 Requisitos para Iniciar o Projeto do Teste

O projeto do teste depende basicamente da abordagem definida durante o planejamento dos testes e dos requisitos do software em teste.

Desta forma, para dar início ao projeto do teste é necessário que a abordagem do teste e os requisitos do software estejam definidos e disponíveis.

Caso as técnicas estruturais sejam empregadas para o projeto do teste de unidades, o correspondente código-fonte também deve estar disponível.

2.5 Atividades do Projeto do Teste

As atividades do projeto do teste são as seguintes:

1. Refinar a Abordagem do Teste;
2. Identificar e Priorizar Cenários de Teste;
3. Projetar e Priorizar Casos de Teste;
4. Documentar e Organizar Casos de Teste;
5. Elaborar Procedimentos de Teste;
6. Definir Base de Dados de Teste;
7. Refinar Critérios de Suspensão e Retomada do Teste;
8. Criar Associações entre Casos de Teste e Artefatos;
9. Refinar Ambiente de Teste;
10. Refinar Cronograma de Teste.

Observações:

- Conforme mencionado anteriormente, nem sempre será necessária a realização formal de todas essas atividades.
- Embora as atividades sejam apresentadas segundo uma forma que busque refletir sua ordem natural de realização, na prática elas podem ser realizadas de forma concorrente.

2.6 Critérios para Encerrar o Projeto do Teste

O principal critério para o encerramento do projeto de teste é que todos os casos de teste previstos segundo os critérios e técnicas de teste definidos no Plano de Teste tenham sido projetados.

Outros critérios que podem ser empregados para encerrar o projeto de teste são listados a seguir:

- Os resultados de todas as atividades do Projeto de Teste devem ter sido determinados;
- O Projeto de Teste deve atender aos padrões e às normas adotados pela empresa, avaliados por um processo de revisão;
- Aprovação do Projeto de Teste por quem de direito.

Da mesma forma que o planejamento de testes, o projeto de testes também é incremental, pois durante a fase de execução e controle podem ser identificados itens ou funcionalidades para os quais seja necessária a geração de novos casos de teste, que podem ser incorporados ao projeto inicial ou dar origem a uma nova versão do Projeto de Teste.

3 Descrição das Atividades

As seguintes atividades devem ser consideradas no projeto de teste:

3.1 Refinar a Abordagem do Teste

A abordagem do teste, definida durante o planejamento do teste, determina os passos gerais para a realização dos testes. Durante o projeto de teste, vários desses passos podem ser detalhados. Este detalhamento pode abranger, por exemplo, os seguintes aspectos:

- Refinamento dos ciclos de teste, com a identificação e distribuição de casos de teste pelos ciclos;
- Identificação da necessidade de outros critérios de geração de casos de teste – mais rigorosos ou menos rigorosos – como resultado da análise mais profunda dos requisitos, normas e legislação relativos ao sistema;
- Seleção dos casos de teste para realização dos testes de regressão, por exemplo, casos de teste que exercitam funções básicas e que devem ser re-executados periodicamente ou após alterações significativas no software;
- Reavaliação das métricas definidas no planejamento;
- Associação dos tipos e critérios de teste aos componentes ou funcionalidades do sistema;
- Etc ...

Entradas:

Informação	Fontes Possíveis
Abordagem de teste, riscos associados ao produto, cronograma e equipe de teste, requisitos do sistema	Plano de Teste, documento de requisitos do sistema

Saídas:

Abordagem de teste refinada

3.2 Identificar e Priorizar Cenários de Teste

Um cenário de teste estabelece condições de teste associadas aos itens ou eventos que devem ser verificadas por um ou mais casos de teste (uma função, transação, funcionalidade, atributo de qualidade ou elemento da arquitetura, do código ou da estrutura de dados do software). Um requisito pode dar origem a um ou mais cenários de teste, em função de seu risco associado e da experiência do projetista de teste. Por exemplo, um requisito que defina as regras associadas a uma

operação de cadastro pode dar origem a um cenário de teste associado ao preenchimento correto de todos os campos do cadastro e, para cada campo do cadastro, um cenário de teste associado ao preenchimento do campo com valores inválidos. O nível de detalhamento dos cenários de teste depende fundamentalmente do risco associado ao requisito sendo testado e à experiência do projetista de teste.

Exemplos de cenários de teste:

1. Cálculo do valor do seguro para motoristas do sexo feminino com menos de 65 anos;
2. Cálculo do valor do seguro para motoristas do sexo feminino com mais de 65 anos;
3. Cálculo do valor do seguro para motoristas do sexo masculino com menos de 30 anos;
4. ...

Entradas:

Informação	Fontes Possíveis
Requisitos do Sistema	Documento de requisitos do sistema

Saídas:

Cenários de teste

3.3 Projetar e Priorizar Casos de Teste

Para cada cenário de teste identificado na atividade anterior, um ou mais casos de teste devem ser gerados. Um caso de teste é composto por um conjunto de valores de entrada e resultados esperados, pré-condições e pós-condições de execução. Casos de teste mais complexos podem exigir a definição de passos para sua execução, com entradas e ações intermediárias bem como resultados intermediários.

Os casos de teste devem ser gerados a partir das técnicas e critérios definidos na abordagem de teste, descrita no Plano de Teste e refinada no Projeto de Teste. As técnicas e critérios guiam o analista de teste na geração dos dados de entrada dos casos de teste, ao passo que os requisitos do software devem ser empregados para a definição dos resultados esperados.

A melhor compreensão dos requisitos do software – por exemplo, pelo detalhamento dos fluxos principal, alternativos e de exceção de um caso de uso – pode levar o projetista de teste a perceber que algumas funcionalidades são mais complexas do que inicialmente previsto, levando-o a empregar um critério de teste mais rigoroso do que o planejado.

Por exemplo, se durante a definição dos casos de teste o projetista percebe que valores de saída dependem da combinação de valores e condições de entrada, ele pode optar por gerar casos de teste segundo o mecanismo conhecido como tabela de decisões, mesmo se a abordagem de teste tiver especificado somente o critério Particionamento de Equivalência.

Os casos de teste gerados devem ser documentados e podem ser revisados por outros grupos envolvidos no projeto do sistema: analistas de software, responsáveis pelo levantamento dos requisitos do sistema, representantes dos usuários, equipe de desenvolvimento, etc.

Entradas:

Informação	Fontes Possíveis
Abordagem do teste refinada	Projeto de Teste
Lista de cenários	Projeto de Teste

Riscos associados ao software	Plano de Teste

Saídas:

Casos de teste projetados

3.4 Documentar e organizar casos de teste

Os casos de teste projetados para uma versão do software podem ser reaproveitados no teste das versões futuras desde que estejam convenientemente documentados e organizados. Além disso, a documentação dos casos de teste permite a comunicação entre o projetista dos testes e o testador, registrando também evidências do projeto para futuras auditorias.

Os casos de teste devem ser organizados ou agrupados segundo critérios que facilitem sua busca e identificação no futuro.

Exemplos de organização:

1. Organizar casos de teste por funcionalidades ou casos de uso: esta forma de organização permite que os casos de teste associados a um determinado requisito modificado sejam identificados, alterados se necessário para refletir o novo requisito, e re-executados.
2. Organizar casos de teste por módulos: esta forma de organização permite que os casos de teste associados a um determinado módulo modificado sejam identificados, alterados se necessário para refletir as alterações realizadas, e re-executados.
3. Organizar casos de teste por fases de desenvolvimento: esta forma de organização permite, por exemplo, que casos de teste associados ao teste de integração sejam identificados e repetidos quando uma nova versão de um módulo for incorporada ao software.
4. Organizar casos de teste pelos riscos identificados: esta forma de organização facilita o acompanhamento do teste em termos das funcionalidades ou módulos que apresentam maiores riscos, e que desta forma são mais críticos para a aceitação do software.

Após a definição dos casos de teste é possível perceber que alguns deles só podem ser executados após a execução de outros casos de teste. Por exemplo, os testes de inclusão de registro podem ser necessários para a realização dos testes de busca, alteração e exclusão de registros do mesmo tipo. Os testes do cálculo do prêmio de um seguro que envolva várias combinações de dados de entrada só devem ser realizados após a realização dos testes do mesmo cálculo que não envolvam combinações de dados de entrada. Tal dependência entre casos de teste deve ser registrada, pois será importante para a elaboração do procedimento de teste.

Os casos de teste podem ser documentados por meio de ferramentas simples, como planilhas eletrônicas, ou por meio de ferramentas especializadas.

Entradas:

Informação	Fontes Possíveis
Casos de Teste	Projeto de Teste

Saídas:

Casos de teste documentados

3.5 Elaborar procedimento de teste

Esta atividade tem como objetivo descrever as ações que devem ser realizadas pelos testadores para a execução dos casos de teste. Normalmente, um procedimento de teste agrupa vários casos de teste que estão associados por alguma razão. No exemplo empregado neste guia todos os casos de teste gerados para a avaliação do cálculo do desconto ou acréscimo aplicado ao seguro de um automóvel podem estar agrupados em um único procedimento de teste.

Um procedimento de teste normalmente é organizado nas seguintes seções:

- Passos iniciais:

  Ações que o testador deve realizar para que o procedimento possa ser executado, tais como preparar o ambiente de teste e verificar as condições associadas ao estado do software, do ambiente de teste ou dos testes já realizados.

  Ex.: Executar o script “scr_carga_BD_077”, responsável por carregar a base de dados necessária ao teste.

- Passos do procedimento:

  Descrição detalhada das ações que o testador deve realizar para a execução dos casos de teste projetados, incluindo a verificação dos resultados obtidos.

- Passos finais:

  Ações que o testador deve realizar para finalizar a execução do procedimento.

  Ex.: Executar o script “scr_final_078”, responsável por assegurar que o estado final do ambiente de teste está consistente.

- Requisitos especiais de ambiente:

  Componentes do ambiente de teste necessários especificamente à realização do procedimento.

- Dependência com outros procedimentos:

  Relação de outros procedimentos que precisam ser executados completamente sem a ocorrência de nenhuma falha antes do início da execução do procedimento.

  A dependência de dados entre procedimentos, caracterizada pela necessidade de dados gerados por um procedimento para a execução de um segundo procedimento, deve ser evitada.

A descrição detalhada de um procedimento de teste facilita sua aplicação; entretanto, quanto mais detalhada for a definição de um procedimento de teste, maior é o esforço necessário à sua elaboração, maior a chance de se introduzir erros nesta descrição, maior o esforço necessário à sua manutenção, e, consequentemente, maior o risco desta descrição ficar desatualizada com o correr do tempo. Por outro lado, um procedimento de teste pouco detalhado dificulta sua aplicação por um testador que não conheça bem a operação do software.

Um procedimento especial pode ser necessário para a execução de testes iniciais (*smoke test* ou *intake test*), que engloba a execução de um subconjunto dos testes projetados que cobrem algumas funcionalidades principais; este procedimento pode ser realizado como uma etapa inicial para avaliar se o software está pronto para um esforço maior de teste.

Entradas:

Informação	Fontes Possíveis
Casos de teste documentados	Projeto de Teste
Modo de operação do software	Documentos de operação do software, protótipos de interface com o usuário

Saídas:

Procedimentos de teste documentados

3.6 Definir base de dados de teste

Para a execução de alguns casos de teste pode ser necessário que a base de dados do sistema contenha um conjunto específico de dados previamente cadastrados. Estes dados podem ser inseridos na base pela prévia execução de outros casos de teste, mas como anteriormente mencionado, isso pode criar uma indesejada e desnecessária dependência entre procedimentos de teste. Uma forma de evitar tal dependência é pela identificação dos dados necessários à execução dos testes e pela sua incorporação em uma base de dados de teste.

Caso uma base de dados histórica, que contenha dados reais, seja usada para a geração da base de dados de teste, cuidados devem ser tomados para a proteção de dados confidenciais.

Entradas:

Informação	Fontes Possíveis
Casos de teste	Projeto de Teste
Procedimentos de teste	Projeto de teste

Saídas:

Bases de dados de teste, scripts de carga de bases de teste

3.7 Refinar critérios de suspensão e retomada do teste

Tem como objetivo refinar os critérios de suspensão e retomada do teste definidos durante o planejamento dos testes. Possivelmente estes critérios foram definidos de forma genérica e em alto nível, pois o responsável pelo planejamento dificilmente possuía o entendimento necessário do software ou mesmo a necessidade de detalhar muito tais critérios. Esta situação fica alterada após a definição dos casos de teste, quando o projetista tem todas as informações necessárias para definir estes critérios, por exemplo, em termos de grupos de casos de teste associados a funcionalidades ou módulos críticos ou de alto risco.

Entradas:

Informação	Fontes Possíveis
Critérios de suspensão e retomada do teste	Plano de Teste
Casos de teste documentados	Projeto de Teste

Saídas:

Critérios de suspensão e retomada do teste refinados

3.8 Criar associações entre casos de teste e artefatos

Uma associação explícita deve ser mantida entre os casos de teste gerados e os requisitos do software que lhes deram origem. Esta ligação de rastreabilidade oferece diversas vantagens na fase de execução dos testes:

- Permite identificar requisitos associados a testes que ainda não foram realizados ou que detectaram falhas ainda não sanadas.
- Permite identificar casos de teste associados a requisitos que sofreram modificações;
- Permite verificar a quantidade de casos de teste projetados por requisito;
- Permite verificar quantos casos de teste foram executados por requisito;

O conceito de rastreabilidade pode ser estendido para outros artefatos gerados ao longo do projeto do software, tais como documentos associados ao projeto do software ou arquivos com o correspondente código-fonte. Embora importante, a manutenção destas informações consome um considerável esforço e fica facilitada com o uso de ferramentas adequadas.

Entradas:

Informação	Fontes Possíveis
Casos de teste documentados	Projeto de Teste
Requisitos de software	Documento de requisitos do software

Saídas:

Ligações de rastreabilidade

3.9 Refinar ambiente de teste

Durante o planejamento dos testes foi estabelecido o ambiente necessário à sua execução. Entretanto, novas necessidades do ambiente podem ser identificadas durante o projeto dos testes. Caso isso ocorra, as ações necessárias à obtenção dos novos componentes do ambiente devem ser tomadas.

Caso seja prevista alguma dificuldade na obtenção dos novos componentes, a análise de risco do projeto deve ser revisada.

Entradas:

Informação	Fontes Possíveis
Ambiente de teste	Plano de Teste

Saídas:

Ambiente de teste refinado

3.10 Refinar cronograma de teste

O cronograma de teste definido na fase de planejamento pode ser detalhado pela incorporação dos procedimentos de teste definidos, caso maiores visibilidade e controle do andamento dos testes **executados sejam necessários. Para que isso ocorra o projetista de teste deve ter uma noção do esforço necessário à execução dos procedimentos de teste.**

Se o refinamento do cronograma indicar que o tempo inicialmente previsto para a execução dos testes é insuficiente deve-se buscar o prolongamento do prazo alocado para a realização dos testes,

a incorporação de novos membros na equipe de teste ou a alteração do escopo do teste. Qualquer que seja a decisão, a análise de risco do projeto deve ser revisada.

Caso necessário, o planejamento dos ciclos de teste previstos deve ser revisado ou detalhado.

Entradas:

Informação	Fontes Possíveis
Cronograma de teste	Plano de Teste

Saídas:

Cronograma de teste refinado

4 Critérios para a Geração de Casos de Teste

Nesta seção são apresentados e descritos os conceitos básicos dos principais critérios associados à técnica funcional e à técnica estrutural; detalhes adicionais e exemplos mais completos estão contemplados em tutoriais específicos a serem elaborados.

A aplicação dos critérios considerados dá origem a casos de teste com dados de entrada válidos e inválidos. Os dados válidos estão associados a situações normais do software e visam avaliar a correta implementação de uma funcionalidade. Os dados inválidos estão associados a situações não permitidas para o software e visam a avaliar se a implementação trata adequadamente essas situações. A postura de um projetista de teste é considerada muitas vezes como destrutiva, uma vez que ele deve buscar situações em que o sistema possa reagir incorretamente aos dados fornecidos. Entretanto, esta postura deve ser considerada como positiva e ser utilizada para o desenvolvimento de produtos de melhor qualidade. Se o projetista de teste não identificar essas situações de erro os usuários do software conseguirão colocar o software nestas situações.

A atividade de teste pode ser uma atividade de prevenção de defeitos, e não simplesmente de detecção de defeitos. Por exemplo, os casos de teste da fase de teste de sistema podem ser projetados em paralelo ao desenvolvimento do software e repassados à equipe de desenvolvimento. Esta comunicação permite que a equipe de desenvolvimento possa identificar situações de uso não inicialmente previstas ou requisitos inconsistentes ou ambíguos e, desta forma, construa um software mais robusto e de melhor qualidade.

Critérios Funcionais

Os seguintes critérios da técnica funcional são abordados:

1. Erros Sintáticos;
2. Particionamento de Equivalência;
3. Análise de Valores Limites;
4. Testes combinatórios;
5. Testes Baseados no Ciclo de Vida das Entidades;
6. Teste Baseado em Casos de Uso.

A apresentação destes critérios será complementada com a apresentação dos conceitos de testes exploratórios, que pode ser considerado como um método informal para a geração de casos de teste onde o conhecimento adquirido pelo projetista de teste durante a execução de um teste é utilizado para o projeto dos demais casos de teste.

Os critérios da técnica funcional procuram cobrir, de alguma forma, todos os requisitos associados a um software; seu uso não considera os elementos do código fonte do software.

Os casos de teste gerados pelos critérios funcionais podem ser complementados por casos de teste gerados pela aplicação de critérios da técnica estrutural, que são baseados no código fonte do software em teste e que garantem, de alguma forma, que determinados elementos do código fonte sejam exercitados.

Os seguintes critérios da técnica estrutural são apresentados:

1. Teste de Comandos;
2. Teste de Ramos.

Observações:

- Esta seção contemplará somente critérios para o teste de funcionalidade do software; outros tipos de teste para testar outras características do software, como os testes de usabilidade, desempenho ou segurança, não serão abordados.
- A aplicação dos critérios está baseada na especificação apresentada como exemplo na seção 2.

4.1 Erros Sintáticos

Este critério direciona o testador a selecionar casos de teste que exercitem valores de entrada sintaticamente inválidos.

Esta técnica é útil para detectar falhas provocadas pelo processamento incorreto de dados de entrada que sejam sintaticamente inválidos.

Exemplo:

Para o exemplo do sistema empregado neste guia foram identificadas as seguintes situações de teste para as variáveis “nome”, “idade” e “sexo” pela aplicação desta técnica:

Variável	Valores válidos	Valores inválidos
Nome	Campo obrigatório Comprimento < 40 caracteres Caracteres alfa-numéricos	Comprimento > 40 caracteres Caracteres inválidos Campo nulo
Idade	Campo obrigatório Comprimento < 3 caracteres Caracteres numéricos Valor inteiro	Comprimento > 3 caracteres Caracteres não numéricos Valor não inteiro Campo nulo
Sexo	Campo obrigatório M, F	Campo nulo Outras letras

Um caso de teste gerado com o emprego deste critério é apresentado a seguir:

Nome	Antônio da Silva
Idade	45
Sexo	M
Resultado	Valor base inalterado
Mensagem de erro	--

Um conjunto de casos de teste gerado por este critério para o campo “idade” é apresentado a seguir:

Dados de Entrada			Resultado Esperado	
Nome	**Idade**	**Sexo**	**Resultado**	**Msg**
Renato	54	M	Valor base inalterado	-----
Bernadete	0027	F	-----	Campo idade com tamanho inválido
Carlos	$%	M	-----	Campo idade com caracteres inválidos

Os dois últimos casos de teste apresentados são bastante semelhantes – no 2º caso de teste o campo “idade” foi preenchido com quatro caracteres – 0027 – um número de caracteres maior do que o permitido, ao passo que no 3º caso de teste o mesmo campo recebe caracteres inválidos – $%. Uma boa prática para o projeto de teste é que cada caso de teste exercite no máximo uma situação que viole um requisito ou regra de negócio do software em teste, como apresentado no exemplo. Se o projetista de teste tivesse agrupado os casos de teste 02 e 03, criando um caso de teste único, com quatro caracteres inválidos, o sistema poderia detectar corretamente uma situação de erro (por exemplo, o emprego de caracteres inválidos) sem que fosse possível afirmar algo sobre a identificação da outra situação de erro (número inválido de caracteres).

4.2 Particionamento de Equivalência

Direciona o testador a selecionar casos de teste com valores oriundos das classes de dados do domínio (ou conjunto de valores) das variáveis do software. Esta técnica está baseada no pressuposto de que se as classes forem corretamente identificadas e se o software tratar corretamente um elemento de cada classe ele também tratará corretamente os demais elementos da classe.

Exemplo:

Para cada uma das três variáveis (nome, idade e sexo) do exemplo utilizado por este guia foram identificadas as seguintes classes de equivalência:

Nome:

Foram identificadas duas classes de equivalência, a 1ª com nomes válidos e a 2ª com nomes inválidos (com caracteres não alfabéticos);

Idade:

- 1ª classe: valores negativos;
- 2ª classe: $0 \leq idade < 18$
- 3ª classe: $18 \leq idade < 30$
- 4ª classe: $30 \leq idade \leq 65$
- 5ª classe: $65 < idade \leq 100$;
- 6ª classe: valores superiores a 100 anos.

Sexo:

Foram identificadas três classes, duas com valores válidos (“M” e “F”) e uma com valores inválidos.

Para cada variável foram identificadas classes com valores válidos e inválidos.
Um conjunto de casos de teste gerado por este critério para o campo “idade” é apresentado a seguir:

Dados de Entrada			Resultado Esperado	
Nome	Idade	Sexo	Resultado	Msg
Antônio	-8	M	-----	Valor inválido para o campo “idade”
Bernadete	15	F	Desconto de 10%	Valor inválido para o campo “idade”
Carlos	22	M	Adicional de 15%	
Débora	47	F	Desconto de 10%	-----
Estevam	78	M	Adicional de 5%	-----
Fernanda	113	F	-----	Valor inválido para o campo “idade”

Casos de teste adicionais devem ser gerados para as variáveis “nome” e “sexo”.

O tutorial específico sobre este critério de teste apresentará como as classes de equivalência são identificadas e agrupadas para a obtenção de casos de teste.

Notas:

Os casos de teste obtidos, que foram gerados para exercitar o software com valores de todas as classes identificadas para a variável “idade”, utilizam dados de clientes tanto do sexo masculino como feminino; embora esta seja uma decisão sensata, não é exigida pelo critério.

Como este critério é aplicado a campos ou variáveis individuais, ele não trata situações nas quais os resultados do software dependam de combinações de valores de dados de entrada. Por exemplo, os casos de teste gerados não avaliam se o software trata corretamente um motorista do sexo masculino com idade entre 30 e 65 anos que, segundo a especificação apresentada, não deve receber descontos. Compare a aplicação deste critério com, por exemplo, a aplicação da análise de combinações de dados de entrada.

Alguns requisitos de teste definidos pelo critério anterior – Erros Sintáticos – não foram exercitados pela aplicação deste critério (por exemplo, tamanhos inválidos para os campos de entrada de dados). Técnicas e critérios de teste são complementares e devem ser usados de forma combinada.

As variáveis de saída também devem ser analisadas para a aplicação deste critério. Suponhamos que no exemplo dado seja acrescentada uma nova regra de negócio que limite o valor do desconto a R$ 500,00. Para a correta aplicação deste critério deveriam ser gerados casos de teste que dessem origem a descontos superiores e inferiores a R$ 500,00.

4.3 Análise de Valores Limites

Direciona o testador a selecionar casos de teste com valores oriundos das fronteiras das classes de equivalência identificadas para o software; a experiência mostra que justamente nestes pontos os defeitos ocorrem com maior frequência.

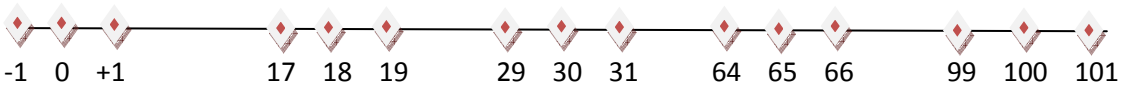
Exemplo:

Empregando-se as classes de equivalência levantadas na seção anterior foram identificadas as seguintes situações de teste para a variável de entrada “idade”:

- 1ª classe: valores negativos;
- 2ª classe: $0 \leq idade < 18$
- 3ª classe: $18 \leq idade < 30$
- 4ª classe: $30 \leq idade \leq 65$
- 5ª classe: $65 < idade \leq 100$;

- 6ª classe: valores superiores a 100 anos.

Estas classes e seus valores limites podem ser representadas graficamente pela seguinte figura:



Variável: Idade	Comportamento esperado do sistema
-1, 0, 1	Situação de erro identificada
17	Situação de erro identificada
18, 19	Cálculo de prêmio de seguro efetuado
29, 30, 31	Cálculo de prêmio de seguro efetuado
64, 65, 66	Cálculo de prêmio de seguro efetuado
99, 100	Cálculo de prêmio de seguro efetuado
101	Situação de erro identificada

A partir das situações identificadas casos de teste são gerados com a atribuição de valores para as demais variáveis de entrada e a especificação dos respectivos resultados esperados.

Um conjunto de casos de teste gerado por este critério para o campo “idade” é apresentado a seguir:

Dados de Entrada			Resultados Esperados	
Nome	**Idade**	**Sexo**	**Resultado**	**Msg**
Antônio	-1	M	-----	Valor inválido para o campo “idade”
Bernadete	0	F	-----	Valor inválido para o campo “idade”
Carlos	1	M	-----	Valor inválido para o campo “idade”
Débora	17	F	-----	Valor inválido para o campo “idade”
Estevam	18	M	Adicional de 15%	-----
Fernanda	19	F	Desconto de 10%	-----
Geraldo	29	M	Adicional de 15%	-----
Helena	30	F	Desconto de 10%	-----
Ivan	31	M	Valor base inalterado	-----
Joana	64	F	Desconto de 10%	-----
Kleberson	65	M	Valor base inalterado	-----
Luciana	66	F	Adicional de 5%	-----
Marcelo	99	M	Adicional de 5%	-----
Nair	100	F	Adicional de 5%	-----
Orlando	101	M	-----	Valor inválido para o campo “idade”

Este critério, como o anterior, é aplicado a campos ou variáveis individuais, e não garante que todas as situações nas quais os resultados do software dependem de combinações de valores de dados de entrada sejam tratadas. Neste caso específico, a aplicação do critério não gerou casos de teste que avaliem o comportamento do software para um motorista do sexo feminino com mais de 100 anos.

4.4 Testes combinatórios

Em diversas situações o comportamento de um software depende da combinação de valores de duas ou mais variáveis, como é o caso do exemplo apresentado neste Guia, onde o preço final do seguro de um automóvel depende de uma combinação entre a idade e o sexo do motorista.

Testes combinatórios direcionam o testador a exercitar o software utilizando combinações de valores específicos ou de classes para as variáveis do software.

Uma forma de obter estas combinações é pelo uso da estrutura conhecida como **tabela de decisões**, que direciona o testador a selecionar dados de teste que exercitem todas as combinações possíveis entre classes ou valores específicos.

Exemplo:

Uma tabela de decisões gerada para o exemplo utilizado neste guia é apresentada a seguir; as seguintes características devem ser observadas:

1. A linha inicial está relacionada à variável “Sexo”;
2. A linha seguinte está relacionada à variável “Idade”;
3. O número de colunas – 12 – é tal que permite acomodar todas as combinações possíveis entre os valores da variável “sexo” (“F” e “M”) e todas as faixas de valores relevantes para a variável “idade” (negativa, menor que 18 anos, entre 18 e 30 anos, entre 31 e 65 anos, entre 66 e 100 anos, maior que 100 anos); observe que o número de colunas (12) é o produto entre as 2 possibilidades de valores da variável “Sexo” e as 6 faixas definidas para a variável “Idade”;
4. Cada uma das colunas numeradas pode dar origem a um caso de teste, cujo resultado depende da correspondente associação de valores entre as variáveis “Sexo” e “Idade”.

	01	02	03	04	05	06	07	08	09	10	11	12
Sexo	F	F	F	F	F	F	M	M	M	M	M	M
Idade	< 0	1..17	18..30	31..65	66..100	>100	< 0	1..17	18..29	30..65	66..100	>100
Ação												
Resultado	erro	erro	-10%	- 10%	+ 5%	erro	erro	erro	+ 15%	V.B.[2]	+ 5 %	erro

A tabela gerada contém todas as combinações possíveis entre valores das variáveis de entrada do software.

Um conjunto de casos de teste gerado a partir da tabela anterior é apresentado a seguir:

[2] V.B.: Valor Base

Dados de Entrada			Resultados Esperados	
Nome	**Idade**	**Sexo**	**Resultado**	**Msg**
Luciana	-10	F	-----	Valor inválido para o campo “idade”
Bernadete	15	F	-----	Valor inválido para o campo “idade”
Débora	28	F	Desconto de 10 %	-----
Maria	40	F	Desconto de 10 %	-----
Regina	70	F	Adicional de 5%	-----
Teresa	110	F	-----	Valor inválido para o campo “idade”
Geraldo	-8	M	-----	Valor inválido para o campo “idade”
José	13	M	-----	Valor inválido para o campo “idade”
Antônio	20	M	Adicional de 15%	-----
Carlos	55	M	Valor base inalterado	-----
Guilherme	88	M	Adicional de 5%	-----
José	102	M	-----	Valor inválido para o campo “idade”

No exemplo anterior havia apenas duas variáveis, sendo viável gerar todas as combinações de classes de valores; em situações em que várias variáveis precisarem ser consideradas a quantidade de combinações pode tornar proibitiva a geração de todos os casos de teste. Nesses casos, outras formas de geração de casos de teste são mais adequadas, conforme apresentado a seguir.

Suponha que o sistema usado neste guia deva ser instalado em ambientes operacionais que suportem as seguintes configurações:

Componentes	Opções
Navegador	Mozilla Firefox, Google Chrome, Opera
SGBD	MySQL, PostgreSQL, Oracle
Sistema operacional	Windows 7, Linux-Ubuntu, Linux- Red Hat
Servidor de aplicações	JBoss, GlassFish, Weblogic

Para a verificação completa do correto funcionamento do software em todos os ambientes é necessária a configuração de 81 ambientes de teste (3 opções de navegador * 3 opções de gerenciadores de base de dados * 3 opções de sistemas operacionais * 3 opções de servidores de aplicações).

Entretanto, a experiência mostra que erros são mais comuns para combinações de valores de pares de variáveis. Por exemplo, o software falha em uma determinada situação somente quando estiver instalado em um ambiente com certa combinação de navegador e sistema operacional, não importando os valores das demais variáveis.

Se desejarmos aproveitar esta situação escolhendo de forma criteriosa uma combinação de ambientes de teste, obtemos a seguinte tabela, que exercita todos os possíveis pares de componentes.

Navegador	SGBD	Sistema operacional	Servidor de aplicações
Firefox	MySql	Windows 7	Jboss
Firefox	PostgreSQL	Ubuntu	GlassFish
Firefox	Oracle	Red Hat	Weblogic
Chrome	MySql	Ubuntu	Weblogic
Chrome	PostgreSQL	Red Hat	Jboss
Chrome	Oracle	Windows 7	GlassFish
Opera	MySql	Red Hat	GlassFish
Opera	PostgreSQL	Windows 7	Weblogic
Opera	Oracle	Ubuntu	Jboss

Como pode ser observado, com um esforço significativamente menor do que o original, é possível garantir a avaliação do todos os pares de componentes.

A tabela apresentada com os casos de teste foi obtida com o emprego de uma estrutura conhecida como **array ortogonal**, que possui a seguinte característica: dadas duas colunas quaisquer, garante-se que todas as combinações possíveis entre valores das variáveis associadas são gerados. No tutorial a ser elaborado será apresentado como se obter arrays ortogonais para a geração de casos de teste bem como o uso de outras formas equivalentes de se combinar dados de entrada.

Vale a pena ressaltar que este critério pode ser utilizado para a combinação de componentes, como mostrado, mas também para combinações entre condições das variáveis de entrada do software.

Exemplo: Gerar dados de teste usando arrays ortogonais considerando as seguintes condições das variáveis de entrada:

Nome: valor válido, valor inválido

Idade: 17, 18, 19, 29, 30, 31.

Sexo: F, M

Nome	Idade	Sexo
1	17	F
***&¨%	19	F
((%% 21212	30	F
!!! da Silva	18	M
***#&&	29	M
123 %% 90	31	M
Eliane Ribeiro	18	F
Marina Araújo	29	F
Solange Ribeiro	31	F
Ricardo Barbosa	17	M
Joaquim Tavares	19	M
Sérgio da Silva	30	M

Os 24 casos de teste necessários para testar todas as combinações foram reduzidos para 12 casos de teste que exercitam as combinações entre todos os pares de valores das variáveis.

As técnicas combinatórias, bem como as demais técnicas funcionais apresentadas a seguir, são comumente empregadas na fase de teste de sistema, após a realização dos testes de unidade e integração, quando o software em teste deve ter atingido um nível básico de estabilidade. Como estas técnicas estão voltadas para regras de negócios e exercitam situações de teste mais complexas, devem ser utilizadas após o emprego das técnicas anteriormente apresentadas, que exercitam situações mais simples.

4.5 Testes Baseados no Ciclo de Vida das Entidades

Consiste em exercitar as operações de criação, busca, alteração e remoção de objetos ou instâncias de todas as entidades gerenciadas pelo software.

Exemplo:

> Aplicando este critério de teste no exemplo deste guia foram identificados os seguintes cenários de teste:

1.1. Criação de uma apólice de seguro;
1.2. Busca de uma apólice de seguro previamente criada;
1.3. Alteração dos dados de uma apólice de seguro;
1.4. Remoção de uma apólice de seguro.

Estas quatro operações representam uma instância de ciclo de vida para a entidade “Apólice”.

4.6 Testes Baseados em Casos de Uso

Casos de uso descrevem o funcionamento de um software sob o ponto de vista de seus usuários, e contém uma série de informações importantes para a geração de casos de teste; este critério define um conjunto de casos de teste que exercita o fluxo principal do caso de uso bem como seus fluxos alternativos e de exceção.

Exemplo:

Para demonstrar a aplicação deste critério o seguinte requisito será acrescentado aos requisitos definidos inicialmente para o sistema de gerência de seguros:

Uso do CPF para identificar clientes previamente cadastrados:

> O sistema oferece a possibilidade de o operador indicar o número do CPF de um cliente. Caso este cliente esteja cadastrado o sistema recupera sua idade e sexo e realiza os cálculos correspondentes; caso contrário o sistema emite uma mensagem de erro: Cliente não cadastrado.

O caso de uso para a operação de cálculo de seguro utilizado nesta neste guia é apresentado a seguir:

Fluxo principal:

1. O caso de uso se inicia quando o sistema solicita a identificação do cliente;
2. O operador do sistema fornece o nome do cliente [exceção: valor nulo ou inválido];
3. O sistema solicita a idade do cliente;
4. O operador do sistema fornece essa informação [exceção: valor nulo ou inválido];
5. O sistema solicita o sexo do cliente;
6. O operador do sistema fornece essa informação [exceção: valor nulo ou inválido];
7. O sistema realiza o cálculo e informa o resultado ao operador.

Fluxo alternativo: Cliente já cadastrado.

2.A.1 O operador do sistema fornece o CPF do cliente [exceção: CPF não válido]

[exceção: usuário não cadastrado no sistema]

2.A.2 O sistema recupera as informações sobre o cliente.

2.A.3 O sistema realiza o cálculo e informa o resultado ao operador.

Para este caso de uso foram identificados os seguintes cenários de teste:

- Cenário 01: valores corretos para os campos “nome”, “idade” e “sexo”;
- Cenário 02: nome do cliente inválido;
- Cenário 03: nome do cliente nulo;
- Cenário 04: idade do cliente inválida;
- Cenário 05: idade do cliente nula;
- Cenário 06: valor do campo “sexo” não fornecido;
- Cenário 07: valor do campo CPF correto de um cliente cadastrado;
- Cenário 08: valor do campo CPF correto de um cliente não cadastrado;
- Cenário 09: valor do campo CPF incorreto.

Os casos de teste da tabela apresentada a seguir exercitam os cenários identificados:

Dados de Entrada					Resultados Esperados	
Cenário	Nome	Idade	Sexo	CPF	Resultado	Mensagem
01	Antônio	20	M	---	+ 15%	---
02	123#$	40	M	---	---	Valor inválido para campo “Nome”
03	---	50	F	---	---	Campo “Nome” não preenchido
04	Beatriz	$%	F	---	---	Valor inválido para campo “Idade”
05	Carlos	---	M	---	---	Campo “Idade” não preenchido
06	Kalesh	40		---	---	Campo “Sexo” não preenchido
07	---	----	---	185.773.495-51	+ 10%	---
08	---	----	---	535.273.846-93	---	Cliente não cadastrado
09	---	----	---	783.828.125-xx	---	Valor inválido para campo “CPF”

Observação: o CPF “185.773.495-51” corresponde a um cliente de sexo masculino com 28 anos previamente cadastrado na base de dados de teste do sistema.

4.7 Testes Exploratórios

É reconhecido que o simples fato de um testador experiente se sentar para executar testes em um software faz com que ele imagine situações de utilização do software distintas das obtidas pelo emprego dos critérios de geração de casos de teste. Desta forma, é uma boa prática o emprego conjunto de abordagens sistemáticas, baseada na aplicação de critérios como os apresentados para a geração de casos de teste, com abordagens não sistemáticas, que permitem que o testador reserve um período de tempo para explorar situações incomuns de uso do software, e descubra novos cenários de teste ao mesmo tempo em que projeta e executa casos de teste. A abordagem conhecida como teste exploratório, que vem ganhando adeptos, é um exemplo típico de uma abordagem não sistemática para a geração de casos de teste.

Teste Exploratório pode ser considerado como um método informal para o projeto de casos de teste no qual o projetista usa o conhecimento adquirido durante a execução de um teste para direcionar o projeto dos próximos testes. Sua filosofia é que a experiência adquirida durante o teste deve ser utilizada para determinar o foco que os demais testes devem ter. Os testes devem ser acompanhados por um projetista de teste experiente ou pelo próprio gerente de teste, que deve garantir que todos os requisitos ou funcionalidades do software foram adequadamente avaliados.

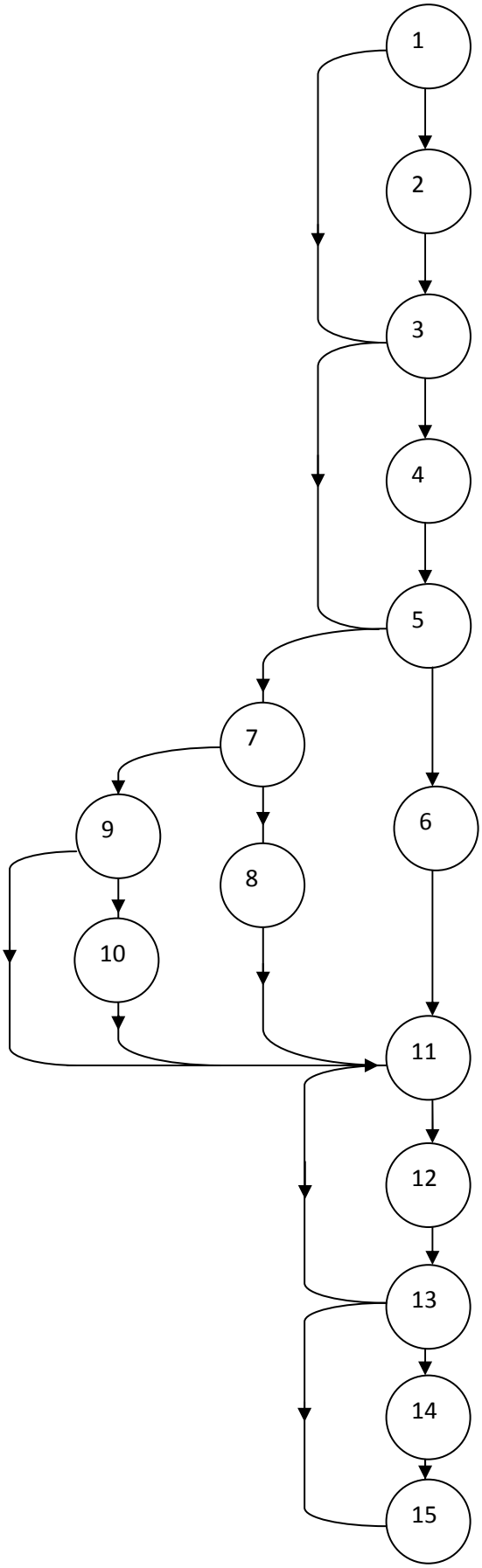
Critérios estruturais

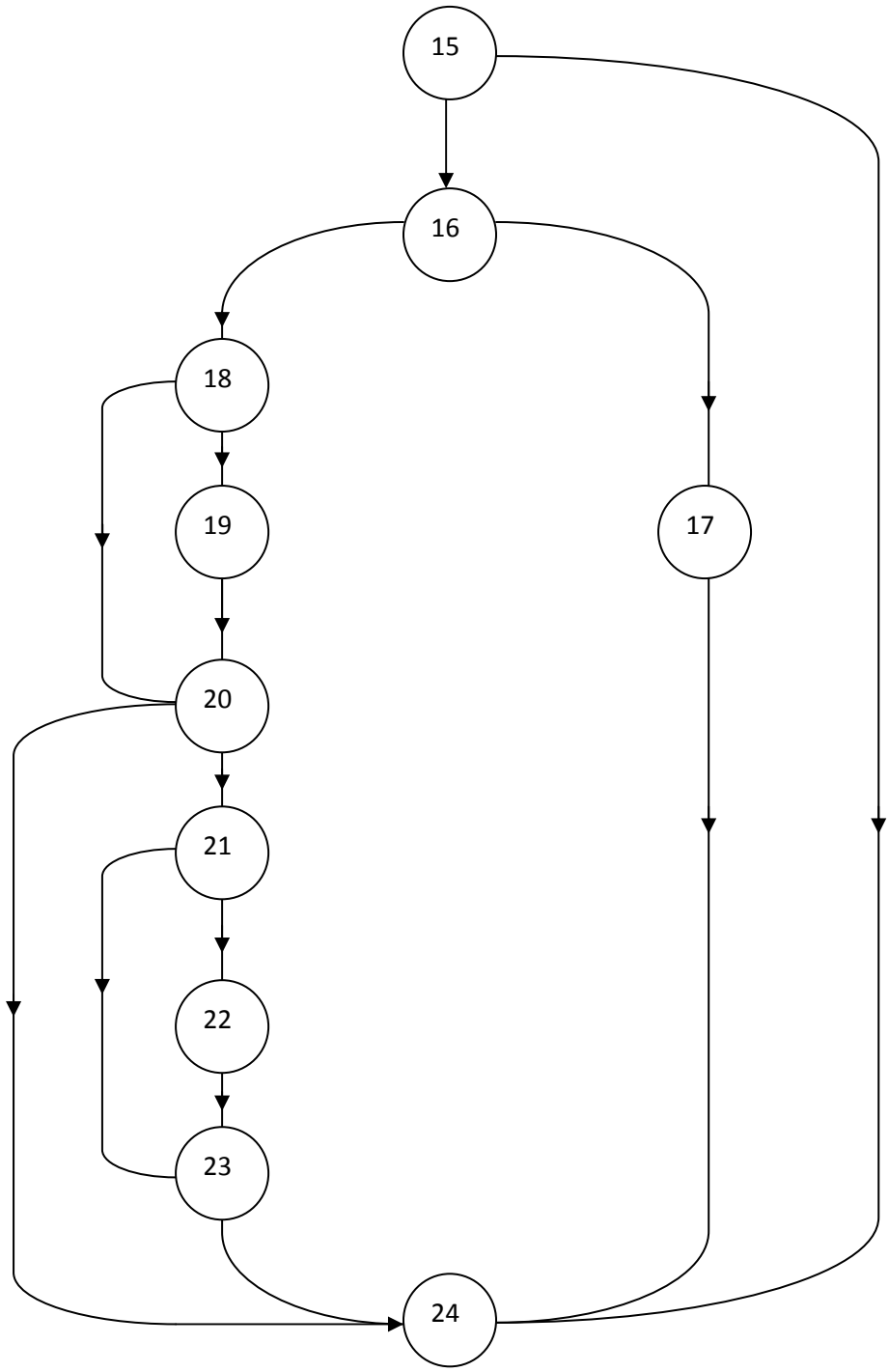
A aplicação dos critérios estruturais depende do conhecimento do código fonte do software sendo testado. Para facilitar a compreensão dos próximos critérios, os requisitos do sistema empregado como exemplo neste guia foram implementados por uma rotina escrita na linguagem Ruby; o código fonte desta rotina, bem como seu correspondente grafo de controle, são apresentados nas páginas seguintes.

Pessoas com experiência em programação não devem ter dificuldades em entender a estrutura do grafo de controle apresentado e seu mapeamento com o código fonte da rotina, cuja listagem traz, em sua terceira coluna, o número do correspondente bloco de cada comando.

Note que a rotina não implementa o requisito associado à situação de um cliente do sexo masculino com idade inferior a 30 anos; segundo os requisitos do sistema, este cliente deve ter o valor base acrescido de 15%, mas a rotina apresentada não realiza nenhum cálculo nesta situação, mantendo o valor base do seguro inalterado.

Linha	Código	Bloco
01	def CalculaPremio(vbase,nome, idade,sexo)	Nó: 1
02	nome.strip!	Nó: 1
03	idade.strip!	Nó: 1
04	sexo.strip!	Nó: 1
05	erro=false	Nó: 1
06	if (nome.size == 0) or (idade.size == 0) or (sexo.size==0)	Nó: 1
07	puts "Erro - Campos com valores nulos"	Nó: 2
08	erro=true; vbase=nil	Nó: 2
09	end	Nó: 2
10	if (nome.size > 40) or (idade.size>3)	Nó: 3
11	puts "Erro - Campos com tamanhos inválidos"	Nó: 4
12	erro=true; vbase=nil	Nó: 4
13	end	Nó: 4
14	if !(idade =~ /^-?[\d]+(\.[\d]+){0,1}$/)	Nó: 5
15	puts " Erro - Campo Valor Base ou Idade não numérico"	Nó: 6
16	erro=true; vbase = nil	Nó: 6
17	else	Nó: 6
18	idade=idade.to_i; vbase=vbase.to_i	Nó: 7
19	if idade<0	Nó: 7
20	puts "Erro - Idade inválida - negativa"	Nó: 8
21	erro=true; vbase = nil	Nó: 8
22	elsif idade<18 or idade>100	Nó: 9
23	puts "Erro - Idade inválida - fora da faixa aceita"	Nó: 10
24	erro=true; vbase = nil	Nó: 10
25	end	Nó: 10
26	end	Nó: 10
27	if !(nome =~ /^[a-zA-Z]+$/)	Nó: 11
28	puts "Erro - Nome invalido"	Nó: 12
29	erro=true; vbase = nil	Nó: 12
30	end	Nó: 12
31	if (sexo != "M" and sexo != "F")	Nó: 13
32	puts "Erro - Valor inválido para campo sexo"	Nó: 14
33	erro=true; vbase = nil	Nó: 14
34	end	Nó: 14
35	if erro == false	Nó: 15
36	if idade > 66	Nó: 16
37	valorBase=valorBase*1.05 # Adicional de 5%	Nó: 17
38	else	Nó: 17
39	if sexo == "F"	Nó: 18
40	valorBase=valorBase * 0.90 # Desconto de 10%	Nó: 19
41	end	Nó: 19
42	if sexo == "M"	Nó: 20
43	if id >= 30	Nó: 21
44	puts " Valor base inalterado"	Nó: 22
45	end	Nó: 22
46	end	Nó: 23
47	end	Nó: 24
48	End	Nó: 24
49	return vbase	Nó: 24
50	end # CalculaPremio	Nó: 24


1
2
3
4
5
7
9
6
8
10
11
12
13
14
15


15
16
18
19
17
20
21
22
23
24

4.8 Todos os comandos

Um conjunto de casos de teste que atenda a este critério é um conjunto cuja execução exercite todos os nós do grafo de controle do software em teste.

O seguinte conjunto de casos de teste atende a este critério em relação ao código apresentado:

Caso de teste 1:

Dados de Entrada	Valor base	100
	Nome	---
	Idade	0025
	Sexo	1
Resultado Esperado	Mensagens de erro	Campos com valores nulos
		Campos com tamanhos inválidos
		Nome inválido
		Valor inválido para campo “Sexo”
	Valor calculado	---

Nós executados: 1,2,3,4,5,7,9,11,12,13,14,15,24

Caso de teste 2:

Dados de Entrada	Valor base	100
	Nome	Antonio
	Idade	-20
	Sexo	M
Resultado Esperado	Mensagens de erro	Idade inválida - negativa
	Valor calculado	---

Nós executados: 1,3,5,7,8,11,13,15,24

Caso de teste 3:

Dados de Entrada	Valor base	100
	Nome	Beatriz
	Idade	120
	Sexo	F
Resultado Esperado	Mensagens de erro	Idade inválida – fora da faixa aceita
	Valor calculado	---

Nós executados: 1,3,5,7,9,10,11,13,15,24

Caso de teste 4:

Dados de Entrada	Valor base	100
	Nome	Carlos
	Idade	4#
	Sexo	M
Resultado Esperado	Mensagens de erro	Campo Valor Base ou Idade não numérico
	Valor calculado	---

Nós executados: 1,3,5,6,11,13,15,24

Caso de teste 5:

Dados de Entrada	Valor base	100
	Nome	Débora
	Idade	70
	Sexo	F
Resultado Esperado	Mensagens de erro	---
	Valor calculado	105

Nós executados: 1,3,5,7,9,11,13,15,16,17,23

Caso de teste 6:

Dados de Entrada	Valor base	100
	Nome	Estevam
	Idade	40
	Sexo	M
Resultado Esperado	Mensagens de erro	---
	Valor calculado	100

Nós executados: 1,3,5,7,9,11,13,15,16,18,20,21,22,23,24

Caso de teste 7:

Dados de Entrada	Valor base	100
	Nome	Fernando
	Idade	35
	Sexo	M
Resultado Esperado	Mensagens de erro	---
	Valor calculado	100

Nós executados: 1,3,5,7,9,11,13,15,16,18,19, 20,24

4.9 Todos os ramos

Um conjunto de casos de teste que atenda a este critério é um conjunto cuja execução exercite todos os ramos do grafo de controle do software em teste.

O seguinte conjunto de casos de teste atende a este critério em relação ao código apresentado:

Caso de teste 1:

Dados de Entrada	Valor base	100
	Nome	---
	Idade	0025
	Sexo	1
Resultado Esperado	Mensagens de erro	Campos com valores nulos
		Campos com tamanhos inválidos
		Valor base ou idade não numérica
		Nome inválido
	Valor calculado	---

Ramos Percorridos: 1-2, 2-3, 3-4, 4-5, 5-7, 7-9, 9-11, 11-12, 12-13, 13-14, 14-15, 15-24

Caso de teste 2:

Dados de Entrada	Valor base	100
	Nome	Antonio
	Idade	-20
	Sexo	M
Resultado Esperado	Mensagens de erro	Idade inválida - negativa
	Valor calculado	---

Ramos Percorridos: 1-3, 3-5, 5-7, 7-8, 8-11, 11-13, 13-15, 15-24

Caso de teste 3:

Dados de Entrada	Valor base	100
	Nome	Beatriz
	Idade	120
	Sexo	F
Resultado Esperado	Mensagens de erro	Idade inválida – fora da faixa aceita
	Valor calculado	---

Ramos Percorridos: 1-3, 3-5, 5-7, 7-9, 9-10, 10-11, 11-13, 13-15, 15-24

Caso de teste 4:

Dados de Entrada	Valor base	100
	Nome	Carlos
	Idade	4#
	Sexo	M
Resultado Esperado	Mensagens de erro	Campo Valor Base ou Idade não numérico
	Valor calculado	Nil

Ramos Percorridos: 1-3, 3-5, 5-6, 6-11, 11-13, 13-15, 15-24

Caso de teste 5:

Dados de Entrada	Valor base	100
	Nome	Débora
	Idade	70
	Sexo	F
Resultado Esperado	Mensagens de erro	---
	Valor calculado	105

Ramos Percorridos: 1-3, 3-5, 5-7, 7-9, 9-11, 11-13, 13-15, 15-16, 16-17, 17-24

Caso de teste 6:

Dados de Entrada	Valor base	100
	Nome	Estevam
	Idade	40
	Sexo	M
Resultado Esperado	Mensagens de erro	---
	Valor calculado	100

Ramos Percorridos: 1-3, 3-5, 5-7, 7-9, 9-11, 11-13, 13-15, 15-16, 16-18, 18-20, 20-21, 21-22, 22-23, 23-24

Caso de teste 7:

Dados de Entrada	Valor base	100
	Nome	Fernanda
	Idade	35
	Sexo	F
Resultado Esperado	Mensagens de erro	---
	Valor calculado	100

Ramos Percorridos: 1-3, 3-5, 5-7, 7-9, 9-11, 11-13, 13-15, 15-16, 16-18, 18-19, 19-20, 20-24

Caso de teste 8:

Dados de Entrada	Valor base	100
	Nome	Rogério
	Idade	20
	Sexo	M
Resultado Esperado	Mensagens de erro	---
	Valor calculado	115

Ramos Percorridos: 1-3, 3-5, 5-7, 7-9, 9-11, 11-13, 13-15, 15-16, 16-18, 18-20, 20-21, 21-23, 23-24

Como pode ser observado, este conjunto de casos de teste é idêntico ao conjunto anterior acrescido do último caso de teste. Este caso de teste é necessário, pois sem ele o arco 21-23 não é exercitado. Note que é justamente este caso de teste adicional que detecta o defeito existente na rotina – a não realização do calculo da alteração de seguro para clientes do sexo masculino com idade entre 18 e 30 anos.

Um exercício interessante é gerar um conjunto de casos de teste para uma rotina semelhante a esta na qual as linhas 42 a 46 não existam; neste caso é possível gerar um conjunto de caso de teste menor que atenda aos dois critérios, mas que não detecte o defeito caracterizado pela falta de comandos.

Observações:

1. A aplicação dos critérios estruturais em rotinas com lógica mais elaborada, que misturem, por exemplo, estruturas de repetição com estruturas de controle, certamente é mais complexa do que no exemplo apresentado. Esta complexidade pode ser tratada, parcialmente, com o emprego de ferramentas que apresentem o grafo de controle do software em teste de forma gráfica e que controlem os nós ou ramos exercitados pelos casos de teste. Estas ferramentas são dependentes da linguagem em que o software foi desenvolvido, e este é um dos motivos pelos quais elas não são amplamente empregadas. Deve-se destacar, também, que muitas vezes os programadores podem aplicar estes critérios baseados unicamente no conhecimento que têm do código, mesmo sem o emprego explícito do grafo de fluxo de controle.
2. Os critérios da técnica estrutural também podem ser empregados para avaliar a cobertura do código atingida pela execução de um conjunto de casos de teste.

Critérios - Conclusões

Nesta seção do Guia foram apresentados alguns dos principais critérios das técnicas funcional e estrutural para a geração de casos de teste.

Cada critério possui um foco bastante objetivo – por exemplo, a aplicação do critério Valores Limites busca avaliar o comportamento de um sistema quando suas variáveis assumem valores oriundos das fronteiras entre comportamentos funcionais distintos deste sistema, mas não exercita todas as combinações possíveis entre os valores de entrada destas variáveis. Por sua vez, o uso de tabelas de decisão busca avaliar o comportamento de um sistema com todas as combinações possíveis entre os valores de suas diversas variáveis, mas não exercita necessariamente o sistema com os valores limites que estas variáveis podem assumir. A aplicação dos diversos critérios em um único exemplo

teve como objetivo demonstrar que os critérios são complementares e devem ser utilizados em conjunto.

A escolha do critério mais adequado em uma determinada situação depende dos requisitos do sistema (e dos riscos associados aos mesmos) e da experiência do projetista de teste.

Uma grande vantagem da utilização de critérios funcionais é a possibilidade de gerar casos de teste sem a necessidade de o código fonte correspondente estar disponível. Isso permite a identificação de requisitos ambíguos ou incorretos no início do projeto de um novo sistema, quando a correção necessária é menos dispendiosa do que seria se fosse realizada nas etapas finais do desenvolvimento. Os critérios estruturais consideram a cobertura dos elementos do código fonte do software (o que não é possível pelos critérios funcionais) e só podem ser aplicados no final do projeto, quando o código fonte estiver disponível.