<Nome do Projeto>

<Iteração/Mestre> Plano de Teste

Versão <1.0>

Histórico da Revisão

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versão** | **Descrição** | **Autor** |
| <dd/mmm/aa> | <1.0> | <detalhes> | <nome> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Índice Analítico

1. Introdução 4

1.1 Finalidade 4

1.2 Escopo 4

1.3 Público-alvo 4

1.4 Terminologia e Acrônimos do Documento 4

1.5 Referências 4

1.6 Estrutura do Documento 4

2. Missão de Avaliação e Motivação dos Testes 4

2.1 Informações Detalhadas 5

2.2 Missão de Avaliação 5

2.3 Motivadores dos Testes 5

3. Itens-alvo dos Testes 5

4. Resumo dos Testes Planejados 5

4.1 Resumo das Inclusões dos Testes 5

4.2 Resumo dos Outros Candidatos a Possível Inclusão 5

4.3 Resumo das Inclusões dos Testes 6

5. Abordagem dos Testes 6

5.1 Catálogos Iniciais de Idéias de Teste e Outras Fontes de Referência 6

5.2 Técnicas e Tipos de Teste 6

5.2.1 Teste de Função (Validação) 6

5.2.2 Teste de Unidade 7

5.2.3 Teste de Integração 7

<Iteração/Mestre> Plano de Teste

# Introdução

## Finalidade

A finalidade do Plano de Teste de Iteração é reunir todas as informações necessárias ao planejamento e ao controle do esforço de teste referente a uma iteração específica. Ele descreve a abordagem dada ao teste do software e é o plano de nível superior gerado e usado pelos gerentes para coordenar o esforço de teste.

Este *Plano de Teste* referente ao <Nome do Projeto> suporta os seguintes objetivos:

• [Identifica os itens que devem ser inspecionados pelos testes.

• Identifica a motivação e as idéias subjacentes às áreas de teste a serem abrangidas.

• Descreve a abordagem de teste que será usada.

• Identifica os recursos necessários e fornece uma estimativa dos esforços de teste.

• Lista os elementos liberados do projeto de teste.]

## Escopo

[Descreva os níveis de teste  por exemplo, Unidade, Integração ou Sistema  e os tipos de teste  como Funcionalidade, Usabilidade, Confiabilidade, Desempenho e Suportabilidade  que serão abordados por este **Plano de Teste**. Também é importante fornecer uma indicação geral das áreas importantes que serão **excluídas** do escopo, especialmente nos casos em que o público-alvo possa supor que elas serão incluídas.

**Observação**: Evite incluir detalhes aqui que serão repetidos nas seções 3, Itens-alvo dos Testes, e 4, Resumo dos Testes Planejados.]

## Público-alvo

[Forneça uma breve descrição do público para o qual o **Plano de Teste** está sendo escrito. Isso ajudará os leitores do documento a identificarem se ele realmente está destinado ao seu uso e também ajudará a evitar que o documento seja usado de forma inadequada.

**Observação**: Freqüentemente, o estilo e o conteúdo do documento são alterados em função do público-alvo.

Esta seção só deverá conter de três a cinco parágrafos.]

## Terminologia e Acrônimos do Documento

[Esta subseção fornece as definições de todos os termos, acrônimos e abreviações necessárias à adequada interpretação do **Plano de Teste**. Evite listar itens que geralmente se aplicam ao projeto como um todo e que já estão definidos no Glossário do projeto. Inclua uma referência ao Glossário do projeto na seção Referências.]

## Referências

[Esta subseção fornece uma lista dos documentos mencionados em qualquer outra parte do **Plano de Teste**. Identifique cada documento por título, número da versão (ou do relatório, se aplicável), data, organização de publicação ou autor original. Evite listar documentos que exercem influência no contexto, mas que não foram mencionados diretamente. Especifique as fontes a partir das quais as “versões oficiais” das referências podem ser obtidas como, por exemplo, nomes UNC de intranet ou códigos de referência de documento. Essas informações podem ser fornecidas por um anexo ou outro documento.]

## Estrutura do Documento

[Esta subseção descreve o que o restante do **Plano de Teste** contém e fornece uma introdução de como o restante do documento está organizado. Ela poderá ser eliminada se for usado um Índice Analítico.]

# Missão de Avaliação e Motivação dos Testes

[Forneça uma visão geral da missão e da motivação dos testes que serão conduzidos nesta iteração.]

## Informações Detalhadas

[Forneça uma breve descrição dos fundamentos que justificam o esforço de teste definido neste **Plano de Teste**. Inclua informações como, por exemplo, o problema principal que está sendo resolvido, os principais benefícios da solução, a arquitetura planejada da solução e um breve histórico do projeto. Quando essas informações estiverem definidas em outros documentos, você poderá incluir referências a esses documentos mais detalhados caso seja apropriado. Esta seção só deverá conter de três a cinco parágrafos.]

## Missão de Avaliação

[Forneça uma breve sentença que defina a missão do esforço de avaliação na iteração atual. Essa sentença poderá incorporar uma ou mais preocupações incluindo:

* localizar o maior número de erros possível
* localizar problemas importantes e avaliar os riscos da qualidade perceptível
* advertir sobre os riscos de projeto percebidos
* certificar um padrão
* verificar uma especificação (requisitos, design ou alegações)
* advertir sobre a qualidade do produto e satisfazer os envolvidos
* advertir sobre os testes
* cumprir as determinações do processo
* e assim por diante

Cada missão fornece um contexto diferente para o esforço de teste e altera a maneira como o teste deverá ser abordado.]

## Motivadores dos Testes

[Forneça um resumo dos principais elementos que motivarão o esforço de teste nesta iteração. Os testes poderão ser motivados por uma série de fatores como, por exemplo, riscos de qualidade, riscos técnicos, riscos do projeto, casos de uso, requisitos funcionais, requisitos não funcionais, elementos de design, falhas ou erros suspeitos, solicitações de mudança, etc.]

# Itens-alvo dos Testes

A listagem abaixo identifica os itens  software, hardware e elementos de suporte do produto  que foram identificados como alvo dos testes. Essa lista representa os itens que serão testados.

[Forneça uma lista de nível superior dos principais itens que estarão sujeitos a testes. Essa lista deve incluir itens produzidos diretamente pela equipe de desenvolvimento do projeto e itens de que dependem esses produtos; por exemplo, o hardware de processamento básico, dispositivos periféricos, sistemas operacionais, produtos ou componentes de terceiros, etc. É recomendável agrupar a lista por categoria e atribuir importância relativa a cada motivador.]

# Resumo dos Testes Planejados

[Esta seção fornece um resumo de nível superior dos testes que serão executados. O resumo fornecido aqui representa uma visão geral de nível superior dos testes que serão e dos que não serão executados.]

## Resumo das Inclusões dos Testes

[Forneça um resumo de nível superior dos principais testes planejados para a iteração atual. Observe o que será incluído no plano e registre o que **não** será incluído explicitamente na seção denominada Resumo das Inclusões dos Testes.]

## Resumo dos Outros Candidatos a Possível Inclusão

[Descreva separadamente as áreas de teste cuja avaliação e investigação você supõe que poderão ser úteis, mas que ainda não foram suficientemente pesquisadas para justificar com certeza a importância de examiná-las.]

## Resumo das Inclusões dos Testes

[Forneça um resumo de nível superior dos possíveis testes que poderiam ter sido conduzidos, mas que foram **explicitamente** **excluídos** deste plano. Se você não for implementar ou executar um tipo de teste, informe claramente que o teste não será executado ou implementado e justifique. A seguir, há exemplos de justificativas que poderão ser usadas:

* “Esses testes não contribuem para alcançar a missão de avaliação.”
* “Não há recursos suficientes para executar esses testes.”
* “Esses testes são desnecessários devido aos testes executados por xxxx.”

Segundo um prisma heurístico, se você achar que é perfeitamente concebível que um dos membros de seu público espere que um determinado aspecto de teste seja incluído e se você não pretender ou não puder incluí-lo, justifique sua exclusão. Se a equipe concordar que a exclusão é óbvia, você provavelmente não precisará listá-la.]

# Abordagem dos Testes

[Esta seção apresenta a estratégia recomendada para criar e implementar os testes necessários. As seções 3, Itens-alvo dos Testes, e 4, Resumo dos Testes Planejados, identificaram **que** itens serão testados e **que** tipos de testes serão executados. Esta seção descreve **como** os testes serão realizados.

Um aspecto a ser considerado na abordagem dos testes são as técnicas a serem usadas. Deverá ser incluído um resumo de como cada técnica poderá ser implementada, de uma perspectiva manual e/ou automatizada, e os critérios para comprovar que a técnica é útil e eficaz. Para cada técnica, forneça uma descrição a seu respeito e defina por que é uma parte importante da abordagem dos testes resumindo brevemente como ela ajuda a alcançar a Missão de Avaliação ou como aborda os Motivadores dos Testes.

Outro aspecto a ser discutido nesta seção são os modelos de Erro ou Falha aplicáveis e as maneiras de abordar como avaliá-los.

À medida que definir cada aspecto da abordagem, você deverá atualizar a seção Error: Reference source not found, Error: Reference source not found, para documentar a configuração do ambiente de teste e outros recursos que serão necessários para implementar cada aspecto.]

## Catálogos Iniciais de Idéias de Teste e Outras Fontes de Referência

[Forneça uma listagem dos recursos existentes que serão consultados para estimular a identificação e a seleção de testes específicos a serem conduzidos. É fornecido um Catálogo de Idéias de Teste de exemplo na seção de exemplos do RUP.]

## Técnicas e Tipos de Teste

## Teste de Unidade, Cadastro da Classe Edição

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar uma edição do evento SATI |
| Técnica: | Definição das Variáveis:  Os dados de uma edição são: Título, Tema, Data Inicial, Data Final, Dados Bancários. Para a inserção de dados de uma edição no banco de dados, é necessário que os dados do objeto não sejam nulos ou vazios. Além disso, a data inicial do evento DEVE ser menor que a data final do evento e somente UMA edição pode se iniciar em um ano. O teste sobre Dados bancários será realizado na integração dos objetos.  As variáveis título e tema não possuem restrições adicionais.  Sobre a registrar de data inicial, não será possível registrar uma edição tal que:   * A data inicial seja 11-11-11 e uma data final 10-11-11. * A data inicial seja 10-11-11 e uma data final 10-11-11.   Ainda sobre a data inicial e final, é possível registrar uma edição tal que:   * A data inicial seja 09-11-11 e uma data final 10-11-11.   Se dois eventos que comecem no mesmo ano, apenas o primeiro será registrado, enquanto o registro do segundo é negado, por exemplo:   * Evento\_1: Data Inicial: 11-11-10 e Data Final 25-11-10. * Evento\_2: Data Inicial: 14-05-10 e Data Final 25-11-10.   Mesmo tendo uma data inicial menor, o Event\_2 tentou ser cadastrado depois de Evento\_1 e, portanto, não será inserido no banco de dados.  Porém quando,   * Evento\_3: Data Inicial: 11-11-10 e Data Final 25-11-10. * Evento\_4: Data Inicial: 14-05-09 e Data Final 25-11-10.   Mesmo terminando na mesma data, os eventos não são restringidos. A única restrição é na data inicial.  Caminho Básico:  A função *Cadastrar* a edição possui duas saídas possíveis, **verdadeiro** e **falso**. A primeira dá-se quando for possível inserir o objeto no banco de dados, logo a segunda é quando o mesmo não é possível.  O método recebe um objeto do tipo Edição contendo os dados do mesmo.  Primeiramente o método *Cadastrar* invoca uma função *VerificarAtributos* para verificar a consistência dos dados.  Dentro da função *VerificarAtributos*, verifica-se se os dados são nulos ou vazios, se algum for, a função retorna **falso**. Caso contrário, é verificado se a Data Inicial do evento é anterior a Data Final, se não, é retornado **falso**. Em seguida, é verificado o ano da edição, se já houver alguma outra edição iniciando-se naquele ano, é retornado **falso**. Se a função não tiver retornado em nenhum ponto desta verificação, é retornado o verdadeiro para a função *Cadastrar*.  Se *VerificarAtributos* é **falso**, a função retorna falso, avisando que não foi possível o cadastro.  Caso *VerificarAtributos* seja **verdadeiro**, então é instanciado uma chamada da camada de persistência e o objeto é enviado para a inserção no banco de dados.  Dentro da classe da persistência, se for possível a conexão com o banco, o dado será inserido com sucesso e retorna verdadeiro, caso não seja possível é retornado falso.  Por fim, a função *Cadastrar* irá retornar o mesmo valor retornado da persistência.  A complexidade ciclomática será 6, determinado pelo total de caminhos possíveis. |
| Estratégias: | A estratégia utilizada para analisar a técnica foi o Teste de Unidades através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | Computador com máquina virtual JAVA instalado.  Acesso à o banco de dados MySQL. |
| Critérios de Êxito: | A técnica suporta o teste de:   * Unidade, testando o método de cadastro de uma nova edição no banco de dados. * Caixa Branca, que foi utilizado para descrever o algoritmo utilizado. |
| Considerações Especiais: | Não Aplica. |

### 

## Teste de Unidade, Cadastro da Classe Pessoa

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar uma pessoa do evento SATI |
| Técnica: | Definição das Variáveis:  Os dados de uma pessoa são: Nome, Tipo, RA, E-mail, Instituição, CPF e RG. Para a inserção de dados de uma pessoa no banco de dados, é necessário que os dados do objeto não sejam nulos ou vazios, com exceção do seguinte caso:   * Se uma pessoa não possuir um RA, isto indica que a pessoa não possui ligação com a UTFPR, logo, é necessário informar a instituição a qual pertence. * Caso a pessoa possua RA, então a UTFPR é a instituição determinada para aquela pessoa.   Logo, se uma pessoa não possuir um RA, o campo Instituição deve ser preenchido, caso contrário, não será possível realizar o cadastro.  Se o RA for preenchido, automaticamente o sistema identifica a instituição como **UTFPR**, antes de cadastrar no banco de dados. Se o RA for preenchido e a Instituição também, o sistema ignora e preenche **UTFPR** como instituição e cadastra a pessoa no banco de dados.  O RA deve ser uma sequência de 7 números quaisquer.  Logo, o RA **1234567** é válido e poderá ser cadastrado e **ABC1!@24234** ou **12345678** não serão aceitos e não serão cadastrados.  O tipo da pessoa pode ser:   * Aluno * Professor * Servidor * Outros   Qualquer outra entrada para o campo Tipo é inválida. Note que não há diferenciação entre letras maiúsculas e minúsculas.  Logo, se o Tipo da pessoa for: *Aluno*, *Professor*, *Servidor* ou *Outros* o cadastro será realizado. Para **qualquer outro dado** informado, o cadastro será negado.  O e-mail da pessoa deve ser único, ou seja, somente um e-mail por pessoa cadastrada. Além disso, um e-mail válido deve obedecer a seguinte expressão regular:  *^[\\w\\.-]+@([\\w\\-]+\\.)+[A-Z]{2,4}$*  Logo, o email deve é composto por:   * Uma sequência de caracteres, exceto por ponto (.) e espaço vazio, * Seguido pelo símbolo arroba (@), * Depois novamente uma sequência de caracteres, exceto por ponto (.) e espaço vazio, * E por fim, de dois a quatro letras.   Sendo assim, o seguinte e-mail deve ser considerado válido:  **exemplo@email.sati**  Enquanto, **exemploNãoValido** não será cadastrado.  O CPF deve ser válido. Para um CPF ser válido, este deve ser composto por 11 números e seguir as regras do Governo Brasileiro para conferir se o CPF é válido.  O CPF válido **14661485785** será aceito, enquanto **12345678901** não será.  Nome e RG não possuem restrições adicionais.  Caminho Básico:  A função *Cadastrar* a pessoa possui duas saídas possíveis, **verdadeiro** e **falso**. A primeira dá-se quando for possível inserir o objeto no banco de dados, logo a segunda é quando o mesmo não é possível.  O método recebe um objeto do tipo Pessoa contendo os dados do mesmo.  Primeiramente o método *Cadastrar* invoca uma função *VerificarAtributos* para verificar a consistência dos dados.  Dentro da função *VerificarAtributos*, verifica-se se os dados são nulos ou vazios, se algum dado for, a função retorna **falso**. Caso contrário, é verificado se o e-mail é válido através da função *validarEmail.*  Dentro *validarEmail* de é verificado se o email segue o padrão descrito acima, se sim retorna **verdadeiro**, caso contrário **falso**.  Se o de *validarEmail* retorno é falso, a função *VerificarAtributos* retorna **falso**. Caso contrário segue a verificação do RA.  Se o RA for válido (definido anteriormente), segue-se a verificação dos dados, caso contrário, retorna **falso**.  Na sequência é testado o tipo da pessoa, se for *Aluno*, *Professor*, *Servidor* ou *Outros*, segue a para verificação do CPF, caso contrário retorna **falso**.  O CPF é testado pelo método *isCPF* da classe validar, caso seja válido, retorna **verdadeiro** para a função *Cadastrar*, caso contrário, retorna **falso**.  Se *VerificarAtributos* é **falso**, a função retorna falso, avisando que não foi possível o cadastro.  Caso *VerificarAtributos* seja **verdadeiro**, então é instanciado uma chamada da camada de persistência e o objeto é enviado para a inserção no banco de dados.  Dentro da classe da persistência, se for possível a conexão com o banco, o dado será inserido com sucesso e retorna verdadeiro, caso não seja possível é retornado falso.  Por fim, a função *Cadastrar* irá retornar o mesmo valor retornado da persistência.  A complexidade ciclomática será 8, determinado pelo total de caminhos possíveis. |
| Estratégias: | A estratégia utilizada para analisar a técnica foi o Teste de Unidades através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | Computador com máquina virtual JAVA instalado.  Acesso à o banco de dados MySQL. |
| Critérios de Êxito: | A técnica suporta o teste de:   * Unidade, testando o método de cadastro de uma nova edição no banco de dados.   Caixa Branca, que foi utilizado para descrever o algoritmo utilizado. |
| Considerações Especiais: | Não Aplica. |

## Teste de Unidade, Cadastro da Classe Dados Bancários

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar dados bancários do evento SATI |
| Técnica: | Definição das Variáveis:  Os dados de uma edição são: *Carteira, Agência, Conta* e *Tipo*. Para a inserção de dados dos dados bancários no banco de dados, é necessário que os dados *Agência, Conta* e *Tipo* o do objeto não sejam nulos ou vazios.  A restrição da Carteira é que a carteira seja maior que 0.  Logo, se Carteira for **1** ou **1000**, o objeto será cadastrado.  Caso a carteira seja **0** ou **-10**, o objeto não será cadatrado.  Caminho Básico:  A função *Cadastrar* os dados bancários possui duas saídas possíveis, **verdadeiro** e **falso**. A primeira dá-se quando for possível inserir o objeto no banco de dados, logo a segunda é quando o mesmo não é possível.  O método recebe um objeto do tipo Dados Bancários contendo os dados do mesmo.  Primeiramente o método *Cadastrar* invoca uma função *VerificarAtributos* para verificar a consistência dos dados.  Dentro da função *VerificarAtributos*, verifica-se se os dados são nulos ou vazios, se *Agência, Conta* ou *Tipo* forem, a função retorna **falso**. Caso contrário, é verificado se *Carteira* é maior ou igual a 0. Se for, é retornado **verdadeiro**, caso contrário retorna **falso**.  Se *VerificarAtributos* é **falso**, a função retorna falso, avisando que não foi possível o cadastro.  Caso *VerificarAtributos* seja **verdadeiro**, então é instanciado uma chamada da camada de persistência e o objeto é enviado para a inserção no banco de dados.  Dentro da classe da persistência, se for possível a conexão com o banco, o dado será inserido com sucesso e retorna verdadeiro, caso não seja possível é retornado falso.  Por fim, a função *Cadastrar* irá retornar o mesmo valor retornado da persistência.  A complexidade ciclomática será 5, determinado pelo total de caminhos possíveis. |
| Estratégias: | A estratégia utilizada para analisar a técnica foi o Teste de Unidades através do método de Caixa Branca. Para descrever o algoritmo foi utilizado o método de teste de caminho básico. |
| Ferramentas Necessárias: | Computador com máquina virtual JAVA instalado.  Acesso à o banco de dados MySQL. |
| Critérios de Êxito: | A técnica suporta o teste de:   * Unidade, testando o método de cadastro de uma nova edição no banco de dados. * Caixa Branca, que foi utilizado para descrever o algoritmo utilizado. |
| Considerações Especiais: | Não Aplica. |

### Teste de Unidade , construindo o produto de forma correta, Código

|  |  |
| --- | --- |
| Objetivo da Técnica: | Cadastrar uma programação no evento da SATI |
| Técnica: | Os dados obrigatorios de uma programação são:   * Datas de inicio e fim, as datas de programação devem estar dentro do prazo de inicio e fechamento da edição no ano em que deseja-se efetuar o cadastro, o formato de uma data é completo, desta maneira quando mencionar data, neste documento, ela está composta da seguinte forma: YYYY-MM-DD HH:MM:SS; * Uma pessoa responsavel, é representada por um objeto da classe pessoa; * Uma edição da qual a programação faz parte, é representado por um objeto da classe edição; * Um custo variável que pode ser alterado durante a edição; * Um custo fixo por responsável; * Local, titulo e descrição, descrição diz respeito ao tipo de programação exemplo, palestra , minicurso, etc ; * Valor de inscrição.   A verificação dos campos ocorre em duas etapas, uma relacionada estritamente com os dados necessarios a camada de persistencia e outra que diz respeito as restrições do projeto.  Datas persistẽncia – Os campos referentes a data e hora de inicio e data e hora de encerramento, são verificados entre sí, de maneira que a data inicial da programação não seja posterior a data de encerrramento.  Local, datas e responsável- . Aṕos a verificação de restrição entre as datas mencionadas, há também a verificação de restrição das datas junto a edição, que não permite a uma programação ser cadastrada numa data que está fora do periodo de inicio e fim da edição. Um responsável não pode estar ocupando dois lugares ao mesmo tempo, para essa verificao o sistema faz uma checagem de onde o responsável está na hora e local em que deseja-se efetuar o cadastro. Para isso o sistema faz uma busca por data, sala e identificação do reponsável.  Caminho básico  A função *Cadastrar* a programação possui duas saídas possíveis, **verdadeiro** e **falso**. A primeira dá-se quando for possível inserir o objeto no banco de dados, logo a segunda é quando o mesmo não é possível.  O método recebe um objeto do tipo Programação contendo os dados do mesmo.Primeiramente o método *Cadastrar* invoca uma função *VerificarAtributos* para verificar a consistência dos dados. |
|  | Dentro da função *VerificarAtributos*, verifica-se se os dados são nulos ou vazios, se algum for, a função retorna **falso**. Caso contrário, é verificado se a Data Inicial da programação é anterior a Data Final, se não, é retornado **falso**. Em seguida, é verificado a data de inicio e fim que deseja-se cadastrar a programação, se a edição já tem agenda definida, é retornado **falso**. A restrição de pesssoa e responsavel é mais complexa e é realizada por último, se o responsável está relacionado a outra programação em outro local entre o horario de inicio e fim desta programação a função retorna **falso**. Quando estes casos não forem satisfeitos a verificação retorna **verdadeiro**, sinalizando que os dados informados podem ser levados a persistencia e a programação pode ser cadastrada. |
| Estratégias: |  |
| Ferramentas Necessárias: | A técnica exige as seguintes ferramentas:   * JUnit com ambiente de programação Java * Java Virtual Machine * Mysql com o schema definido |
| Critérios de Êxito: | A técnica suporta o teste de:   * Unidade, testando o método de cadastro de uma nova programação no banco de dados. * Caixa Branca, que foi utilizado para descrever o algoritmo e utilizado. |
| Considerações Especiais: | [Identifique ou descreva os itens ou problemas (internos ou externos) que exercem influência sobre a implementação e a execução do teste de funcionamento.] |

### Teste de Integração , ver se o o projeto tá funcionando

|  |  |
| --- | --- |
| Objetivo da Técnica: | [Experimentar a funcionalidade do objetivo do teste, incluindo a navegação, a entrada, o processamento e a recuperação de dados a fim de observar e registrar o comportamento-alvo.] |
| Técnica: | [Execute os recursos e os fluxos ou as funções de cada um dos cenários de caso de uso, utilizando dados válidos e inválidos para verificar se:   os resultados esperados ocorrerão quando forem usados dados válidos   as mensagens de erro ou de aviso apropriadas serão exibidas quandoforem usados dados inválidos   cada regra de negócio será aplicada de forma adequada] |
| Estratégias: | [Descreva uma ou mais estratégias que podem ser usadas pela técnica para observar, de forma precisa, os resultados do teste. A estratégia combina o método através do qual a observação pode ser feita e as características dos resultados específicos que indicam um provável êxito ou falha do teste. O ideal é que as estratégias sejam autoverificadas, permitindo que os testes automatizados façam uma avaliação inicial do êxito ou falha do teste. No entanto, tenha atenção para reduzir os riscos inerentes à determinação automática dos resultados.] |
| Ferramentas Necessárias: | [A técnica exige as seguintes ferramentas:   * Ferramenta de Automação de Scripts de Teste * restaurador e reprodutor de imagem da configuração básica * ferramentas de backup e de recuperação * ferramentas de monitoramento de instalação (registro, disco rígido, CPU, memória, etc.) * ferramentas de geração de dados] |
| Critérios de Êxito: | [A técnica suporta o teste de:   todos os principais cenários de caso de uso   todos os principais recursos] |
| Considerações Especiais: | [Identifique ou descreva os itens ou problemas (internos ou externos) que exercem influência sobre a implementação e a execução do teste de funcionamento.] |