Pontifícia Universidade Católica
do Rio de Janeiro

Fernando Alberto Correia dos Santos Junior

Geração Automática de Exemplos de Uso a Partir da Descrição Textual de Casos de Uso

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós-graduação em Informática da PUC-Rio como requisito parcial para obtenção do grau de Mestre em Informática.

Orientador: Prof. Arndt von Staa

Rio de Janeiro
Abril de 2017

Pontifícia Universidade Católica
do Rio de Janeiro

Fernando Alberto Correia dos Santos Junior

Geração Automática de Exemplos de Uso a Partir da Descrição Textual de Casos de Uso

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Arndt von Staa
Orientador
Departamento de informática – PUC-Rio

Prof.ª Simone Diniz Junqueira Barbosa
Departamento de informática – PUC-Rio

Prof. Alessandro Fabrício Garcia
Departamento de informática – PUC-Rio

Prof. Marcio da Silveira Carvalho
Coordenador do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 24 de abril de 2017

Fernando Alberto Correia dos Santos Junior

É bacharel em Engenharia de Computação (Universidade Estadual de Feira de Santana) em 2014. Cursou parte da sua graduação no Instituto Superior Técnico da Universidade Técnica de Lisboa. Trabalhou por mais de 4 anos no desenvolvimento de software, atuando como analista, desenvolvedor e arquiteto de software. É engenheiro de Software do Projeto Supremo em Números na Escola de Direito do Rio de Janeiro da Fundação Getúlio Vargas.

Ficha Catalográfica

Santos Junior, Fernando Alberto Correia dos

Geração automática de exemplos de uso a partir da descrição textual de casos de uso / Fernando Alberto Correia dos Santos Junior; orientador: Arndt von Staa. – 2017.

128 f. : il. color. ; 30 cm

1. Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2017.

Inclui bibliografia

1. Informática – Teses. 2. Engenharia de software. 3. Exemplos de uso. 4. Casos de uso. 5. Geração de comportamentos. 6. Geração automática de exemplos de uso. I. Staa, Arndt von. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Para meus pais e para minha irmã, pelo apoio e confiança.

Agradecimentos

Aos meus pais e a minha irmã, pela atenção, compreensão e suporte que me deram forças para concluir mais essa etapa.

Ao meu orientador Professor Arndt, pelas longas conversas, pela atenção e direcionamento que me motivaram e permitiram a realização deste trabalho.

À CAPES e à PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

Aos meus amigos por todo apoio, paciência e compreensão.

Aos meus colegas da PUC-Rio. Aos professores que participaram da Comissão examinadora.

A todos os professores e funcionários do Departamento de Informática pelos ensinamentos e pela ajuda.

A todos os familiares que de uma forma ou de outra me estimularam ou me ajudaram.

Resumo

Santos Junior, Fernando Alberto Correia; Staa, Arndt von (Orientador). **Geração Automática de Exemplos de Uso a Partir da Descrição Textual de Casos de Uso.** Rio de Janeiro, 2017. 128p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Esta dissertação apresenta uma solução que permite a geração automática de exemplos de uso a partir da descrição textual de casos de uso. Os casos de uso descrevem especificações em um nível de formalização suficiente para a geração dos exemplos. Um exemplo gerado é um texto em linguagem natural que é o resultado da paráfrase de um possível comportamento do software, extraído de um caso de uso e aplicado a um contexto real, em que atores são convertidos em personagens fictícios e os atributos são valorados de acordo com as regras de negócios especificadas no caso de uso. O formato proposto para a construção de exemplos tem como objetivo permitir que clientes possam ler, entender e julgar se o comportamento que está sendo proposto é o desejado. Com isso é esperado que o próprio cliente possa validar as especificações e que, quando defeitos forem encontrados, a especificação possa logo ser corrigida e refletida de volta nos exemplos. Ao mesmo tempo a especificação formalizada na forma de um caso de uso auxiliará desenvolvedores a criar soluções mais próximas do correto por construção, quando comparado com especificações textuais convencionais.

Palavras-chave

Engenharia de software; Exemplos de uso; Casos de uso; Geração de comportamentos; Geração automática de exemplos de uso.

Abstract

Santos Junior, Fernando Alberto Correia; Staa, Arndt von (Advisor). **Automatic Generation of Examples of Use from the Textual Description of Use Cases.** Rio de Janeiro, 2017. 128p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

This master's dissertation presents a solution for the automatic generation of examples of use from the textual description of use cases. Use cases describe specifications in a sufficiently formal way that is enough to automatically generate usage examples. A generated example is a text in a natural language which is the paraphrase of one possible manner to use the software, extracted from the use case and applied to a real context where actors are converted into fictitious personas and attributes are valued according to the business rules specified in the use case. The proposed format to present the example aims to allow clients to read, to understand and to judge whether the expressed behavior is in fact what he wants. With this approach, it is expected that the customer himself can approve the specifications and when defects are found, so the specification can quickly be corrected and reflected in the examples. At the same time, the formalized specification in the form of a use case will help developers create solutions that are by construction closer to the correct one when compared to conventional textual specifications.

Keywords

Software engineering; Examples of use; Use cases; Behavior generation; Automatic generation of examples of use.

Sumário

Lista de Figuras

Lista de Tabelas

Listagens

Lista de Abreviações

API	*Application Programming Interface*
ATDD	*Acceptance Test Driven Development*
BDD	*Behavior-Driven Development*
FA	Fluxo Alternativo
FP	Fluxo Principal
IHC	Interação Humano-Computador
MBT	*Model-Based Test*
NLP	*Natural Language Processing*
STAL	*Structured Test Automation Language*
TDD	*Test Driven Development*
UML	*Unified Modeling Language*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*

1
Introdução

A insatisfação de um cliente nunca é uma coisa boa e, como diz Adzic (2009), a diferença entre a insatisfação e a satisfação não é uma questão de entrega do software, mas sim o quanto que a entrega satisfaz o que o cliente espera. Para simplificar a redação, usaremos "cliente" para denotar os interessados (*stakeholders*) cujos interesses tenham a ver com aspectos funcionais. Portanto, envolve os usuários e os clientes propriamente ditos, além de possíveis outros interessados.

No modelo tradicional, o levantamento de requisitos e especificações de um software são baseados em várias formalizações, traduções e entregas. Considerando os papéis desempenhados ao desenvolver, primeiro, analistas junto com os clientes extraem conhecimento sobre os requisitos, os formalizam em especificações e, em seguida, os entregam para os desenvolvedores e testadores. A partir das especificações, os desenvolvedores extraem conhecimento, adicionam conhecimento sobre a solução, e os traduzem em código executável, que é entregue aos testadores. Por sua vez, os testadores extraem conhecimento a partir das especificações, e traduzem esse conhecimento em scripts de verificação, que são aplicados ao código entregue pelos desenvolvedores, e então é verificada a conformidade do programa com a interpretação dos testadores sobre a especificação. O resultado é um sistema cujo nível de qualidade assegurada depende do rigor e da abrangência dos testes. Além disso, se a especificação for inadequada, o sistema implementado também o será.

Como a participação de clientes na construção das especificações é frequentemente pequena, há uma elevada tendência que a especificação seja inadequada. Processos ágeis procuram resolver esse problema através da disponibilidade de um representante dos clientes que seja capaz de responder corretamente a todas as dúvidas que desenvolvedores e testadores possam ter. Na realidade dilui-se sobre o processo de desenvolvimento o esforço de verificação da

adequação. Em teoria, segundo Adzic (2009), esse processo de verificação da adequação funciona bem, mas na prática não é bem assim, porque ele é intrinsecamente imperfeito e o resultado geralmente é uma grande diferença entre o que foi originalmente desejado e o que é efetivamente entregue. Pode haver grandes lacunas de comunicação a cada passo no processo de desenvolvimento e, a cada tradução, informações podem ser distorcidas ou mal compreendidas, ampliando o grau de não conformidade do software com o que era esperado pelo cliente. Uma interpretação de um testador pode ajudar a corrigir uma falsa interpretação do desenvolvedor, mas também pode ser completamente inadequada e, se usada para remover a causa do "defeito" imaginário, tornar o software inadequado. Finalmente, as duas interpretações podem ser equivocadas. Nesse ponto, o problema reside na comunicação, na interpretação e no entendimento do que é dito, documentado e programado (Adzic, 2009).

1.1. Definição do problema

Em um processo de desenvolvimento ágil, os curtos ciclos de desenvolvimento propiciam um retorno rápido do cliente sobre a adequação do que está sendo desenvolvido, o que, em princípio, permite que problemas sejam descobertos rapidamente (Dybå & Dingsøyr, 2008). De fato, esta é uma boa forma de identificar problemas e mal-entendidos, só que não representa uma ação efetiva e suficiente para evitá-los, até porque essa verificação da adequação será feita somente após o fim de um ciclo. Momento em que qualquer alteração na especificação implicará a refatoração ou remoção de uma funcionalidade implementada e testada[1]. No entanto, se antes do desenvolvimento fosse feita uma verificação da especificação? Essa verificação, potencialmente, reduziria o número de defeitos na especificação e, consequentemente, reduziria o número de problemas

[1] Uma especificação inadequada dará origem a testes também inadequados.

durante o desenvolvimento, reduzindo o risco de retrabalho inútil, além do tempo e o custo final do desenvolvimento do sistema.

Só que o cliente, geralmente, não tem suficiente conhecimento técnico, e muito menos habilidade para interpretar o formalismo presente nas especificações. Na literatura, experiências mostram que clientes entendem melhor uma especificação quando ela é expressa em uma linguagem natural do que quando expressa de forma mais formal, como, por exemplo, diagramas de sequências, diagrama de atividades ou casos de uso (Mason & Supsrisupachai, 2009). Com a evolução dos métodos ágeis, surgiram práticas que buscam encontrar um meio termo na forma como uma especificação é representada. Uma dessas práticas é o Desenvolvimento Dirigido por Comportamento (*Behavior-Driven Development,* BDD), que surgiu com a proposta de permitir aos clientes se expressarem, utilizando uma linguagem natural e semiestruturada, o comportamento que esperam para os seus softwares (North, 2006), onde um comportamento é especificado em um nível de detalhes que permita a sua tradução em um teste de aceitação. Como cita Solis & Wang (2011), o objetivo principal do BDD é a identificação de comportamentos e a construção de especificações executáveis de um sistema. No BDD um comportamento é descrito com uso de cenários, que são exemplos reais de uso daquela funcionalidade descrita no comportamento.

Segundo Adzic (2009), a utilização de exemplos de uso do software como demonstrações práticas de possíveis comportamentos, descritos de uma forma que permita a leitura e compreensão por pessoas sem conhecimento técnico, não somente permite um melhor entendimento ao cliente sobre o que está sendo proposto, mas também é uma forma de compartilhar conhecimento sobre um domínio entre todos envolvidos no processo de construção de um software.

No entanto, no BDD, algumas questões relacionadas à comunicação podem ficar em aberto (Li et al, 2016). Não existe um processo bem definido para a descrição de comportamento, não é possível julgar se a descrição do comportamento realmente descreve por completo aquilo que é desejado. Isso ocorre porque os cenários estarão limitados à experiência e ao imaginativo de quem os escrevem. Heumann (2001) afirma que um dos grandes problemas na construção de testes é a ausência de uma metodologia clara de construção de casos de teste, o que torna difícil assegurar a cobertura e a qualidade dos testes construídos. Visto

que no BDD comportamentos darão origem aos testes. O cenário descrito por Heumann pode ser estendido à descrição de comportamentos no BDD. Como proposta, Heumann (2001) apresenta uma metodologia que se baseia no uso de uma formalização das especificações na forma de descrição textual de casos de uso, uma formalização mínima e suficiente que permite a implementação de um processo claro de construção de casos de teste.

Seguindo a ideia apresentada por Heumann (2001), a formalização de uma especificação na descrição textual de casos de uso pode ser utilizada para, ao invés de uma geração sistemática de casos de teste, a geração dos exemplos reais de uso que compõem um comportamento no BDD. Casos de teste podem ser parafraseados, traduzidos, em exemplos reais de uso que descrevem um comportamento no estilo BDD. Só que quando o tema é teste eficaz, a geração de casos de testes tem um caráter exploratório e torna necessária a geração de uma grande coleção de casos de teste, cada qual exercitando caminhos e condições específicas e diferentes dos demais casos de teste. Segundo relata Pinto[2] o uso de uma grande quantidade de exemplos de uso para a descrição de um comportamento não é interessante. Segundo ele apurou, torna-se muito cansativa a leitura por um cliente ou desenvolvedor e, consequentemente, torna ineficiente a verificação do comportamento. Então, somente gerar não é suficiente; é preciso também selecionar casos que gerem exemplos significativos em uma quantidade mínima e suficiente para que seja possível uma validação da especificação.

1.2. Objetivo geral

Diante do que foi exposto, esta dissertação tem como objetivo apresentar um processo de geração automática de exemplos de uso de um software para a descrição de comportamentos no estilo BDD a partir da descrição textual de casos de uso. Os exemplos gerados devem ser legíveis e inteligíveis tanto pelos clientes

[2] Pinto, T. D., (Pontifícia Universidade Católica do Rio de Janeiro) Comunicação Pessoal, 2016.

como pelos desenvolvedores. Pinto (2013) apresenta uma ferramenta que é capaz de, a partir da descrição textual de casos de uso, fazer uma geração automática de testes funcionais, o FunTester. Em complemento a essa ferramenta, foi desenvolvida uma ferramenta para a geração de exemplos a partir dos casos de uso construídos utilizando o FunTester. Com a geração de exemplos é esperado que o próprio cliente possa validar as especificações e que, quando defeitos forem encontrados, a especificação possa logo ser corrigida e refletida de volta nos exemplos. Ao mesmo tempo a especificação formalizada na forma de um caso de uso auxiliará desenvolvedores a criar soluções mais próximas do correto por construção quando comparado com especificações textuais convencionais, além de permitir a geração de suítes de teste de aceitação criados para uma ferramenta de teste automatizado selecionada pelos desenvolvedores. Essa solução também facilita a manutenção, pois no caso de ser necessária uma alteração pode-se alterar a especificação, gerar novos exemplos, um *Diff* aplicado aos exemplos pode identificar quais exemplos precisam ser revistos e verificados pelo cliente. Além disso esses exemplos "alterados" guiaram os desenvolvedores e, com o uso do FunTester, uma nova suíte de teste poderá ser gerada.

1.3. Objetivos específicos

Este trabalho tem objetivos específicos:

1. Apresentar um processo de geração e seleção de cenários abstratos de uso a partir da exploração de um modelo de comportamento gerado a partir da descrição textual de casos de uso.
2. Gerar uma coleção de exemplos para cada caso de uso.
3. Converter um cenário abstrato de uso em um cenário concreto de uso.
4. Fazer a paráfrase de um cenário concreto de uso em um exemplo de uso escrito usando texto semiestruturado.
5. Avaliar a possibilidade do uso de exemplos como um canal efetivo de comunicação e compartilhamento de conhecimento de domínio.

6. Avaliar o uso de exemplos como uma ferramenta de validação de comportamento.

Os objetivos 5 e 6 serão verificados com o auxílio de um caso de estudo apresentado no Capítulo 6.

1.4. Questões de Pesquisa

Baseado no que foi apresentado foram definidas as questões de pesquisa apresentadas a seguir.

Questão de Pesquisa (QP1) - Como projetar e implementar uma abordagem para a geração de exemplos a partir da descrição textual de casos de uso?

i. Como representar os elementos necessários para a construção de um modelo intermediário para a geração de cenários?
ii. Como utilizar os exemplos como ferramenta de verificação, validação e documentação de comportamentos?

1.5. Organização dos capítulos seguintes

Esta tese é estruturada como segue:

- No Capítulo 2 são apresentados fundamentos que serão abordados nos capítulos seguintes.
- O Capítulo 3 são apresentados conceitos e alguns trabalhos relacionados à presente dissertação.
- No Capítulo 4 é apresentado o processo utilizado pela solução proposta para a geração de exemplos a partir da descrição textual de casos de uso.

- O Capítulo 5 apresenta uma demonstração da viabilidade da ferramenta construída.
- No Capítulo 6 apresenta-se um estudo de caso com análise qualitativa sobre o uso da ferramenta.
- O Capítulo 7 apresenta as conclusões finais e os trabalhos futuros.

2
Contexto

Neste capítulo são apresentados, de forma breve, os seguintes conceitos e temas importantes para contextualizar a proposta apresentada:

- Definição de um comportamento no contexto do desenvolvimento dirigido por comportamento (BDD sigla em inglês)
- Descrição textual de casos de uso
- Descrição de fluxos em casos de uso
- Geração de cenários a partir de um caso de uso
- Exemplo de uso do software

2.1.
Definição de um comportamento no BDD

No intuito de aproximar clientes e desenvolvedores e com o objetivo de tornar as práticas ágeis mais acessíveis aos iniciantes, North (2006) propôs o Desenvolvimento Dirigido por Comportamento (*Behavior-Driven Development,* BDD) que visa aproveitar ao máximo possível da disponibilidade do cliente, aproximando-o ao desenvolvimento e incentivando sua participação ao longo dos ciclos de desenvolvimento do software. O BDD é o resultado de uma combinação de algumas práticas ágeis, dentre elas, destaca-se a influência do Desenvolvimento Dirigido por Teste (*Test Driven Development,* TDD) e o Desenvolvimento Dirigido por Teste de Aceitação (*Acceptance Test Driven Development,* ATDD), só que, diferente delas, o foco do BDD é voltado para o comportamento esperado ao invés do teste.

No BDD o comportamento do sistema é obtido a partir de uma comunicação ativa entre time de desenvolvimento e clientes, incorporando clientes ao processo de desenvolvimento, e tornando mais eficiente o desenvolvimento dirigido por teste de aceitação. Diferente do ATDD, no BDD os testes são escritos de uma forma que visa facilitar o entendimento de uma pessoa sem conhecimento técnico, porque a abordagem prevê o uso de uma linguagem de domínio ubíqua e semiestruturada que permite clientes a especificarem seus próprios testes (Solis & Wang, 2011). Dessa forma, ao mesmo tempo que o BDD serve como uma prática para levantamento e documentação de requisitos, serve também como uma ferramenta para a construção de testes de aceitação (Wanderley et al., 2015).

Um comportamento em BDD está, geralmente, relacionado a uma historieta *(user story*) que traduz uma funcionalidade desejada em um breve texto. Visa responder a três questões:

- Quem é o usuário e qual papel desempenha?
- Do que se trata a funcionalidade desejada?
- E qual benefício o usuário poderá alcançar com a implementação dessa funcionalidade?

Um comportamento no BDD expande a historieta adicionado a ela os possíveis critérios de aceitação na forma de cenários de uso (North, 2006), onde cada cenário de uso descreve como o sistema que implementa uma funcionalidade deverá se comportar quando estando em um estado inicial e um evento (ou uma série de eventos) acontece. A saída é uma ação que altera o estado do sistema. Utilizaremos o termo ação ao invés de saída pois uma ação pode representar qualquer comportamento reativo do sistema. Geralmente a escrita de um cenário de uso de um comportamento tem o formato como apresentado na Tabela 1 (também conhecido como padrão *given-when-then*)

Tabela 1. Formato de um cenário.

Itens	Descrição
Cenário	Breve descrição do cenário.
Dado que	Contexto detalhado do estado atual.
Quando	Um evento ou uma sequência de eventos.
Então	Descrição do resultado esperado

Para ilustrar, um critério de aceitação (adaptado de North, 2006) que retrata um saque de caixa eletrônico. Primeiro apresentamos a historieta na Listagem 1:

Saque de dinheiro em um caixa eletrônico.
***Como** um cliente do banco,* ***Eu quero** realizar um saque de um caixa eletrônico,* ***Então**, não precisarei ficar esperando ser atendido na fila do caixa.*

Listagem 1. Exemplo de uma historieta.

Para entregar essa historieta muitas situações devem ser consideradas: talvez não haja saldo suficiente na conta, ou a conta pode ter saldo mas pode ter ultrapassado o limite diário de saque, ou o valor solicitado ultrapassa o limite diário de saque, entre outras. Considerando apenas a situação em que há saldo em conta e que o valor solicitado está dentro do limite disponível, um possível cenário poderia ser descrito da seguinte forma:

Cenário:	*Há dinheiro suficiente na conta*
Dado que:	*o cliente tem um saldo de $ 100,*
	o cartão é válido.
	e o caixa eletrônico contém suficiente dinheiro
Quando:	*o cliente solicita a quantia de $ 20,00*
Então:	*Deve-se certificar que o novo saldo do cliente é de $ 80,*
	certificar que o dinheiro foi liberado pelo caixa eletrônico
	e certificar que o cartão foi liberado pelo caixa eletrônico

Listagem 2 Exemplo de um cenário de uso.

Uma vez descritos os comportamentos, é esperado que o desenvolvedor possa identificar aquilo que deve ser implementado, e que tenha contexto suficiente para saber como implementar e como testar (Wynnen & Hellesoy, 2012). Caso seja necessária a presença do usuário, o uso de uma linguagem de fácil entendimento permite uma melhor comunicação.

2.2.
Descrição textual de casos de uso

Uma outra forma de documentar especificações é utilizar casos de uso, que descrevem como uma entidade (sistema a ser desenvolvido) interage com outras entidades (atores) comunicando e realizando ações internas para atingir um objetivo específico. Existem várias formas de captura de casos de uso, um consenso comum é especificar um caso de uso como uma sequência de passos, descrevendo um caso de sucesso (fluxo principal), com extensões e variações (fluxos alternativos) descrevendo casos complementares (Heumann, 2001).

Os casos de uso podem ser representados em forma gráfica ou textual. Embora a forma gráfica seja mais comum, a forma textual tem ganho bastante espaço e tem a vantagem do uso da linguagem natural para a descrição, o que facilita a leitura e a análise do caso de uso sem a necessidade de um treinamento especial (Sawant et al. 2014). Para criar um modelo de comportamento a partir da descrição textual de caso de uso, adotamos uma abordagem que compreende em um conjunto de regras de restrição bem definidas dispostos num formulário, projetado para facilitar e uniformizar seu preenchimento (Cockburn, 2000). De forma geral, podemos escolher certos itens que são úteis e podem ser necessários com frequência, como os propostos por Staa (2017) e mostrados a seguir na Tabela 2:

Tabela 2 - Formulário exemplo para a descrição textual de casos de uso Extraído de Pinto, (2014).

Itens	Descrição
Caso de uso	*Nome (identificação) do caso de uso.*
Resumo	*Descrição resumida do objetivo principal.*
Escopo	*O que é abrangido pelo caso de uso.*
Ator principal	*Nome do ator que ativa o caso de uso.*
Interessados	*Nome (identificação) dos interessados, contendo opcionalmente a descrição de seus interesses ou objetivos.*
Invariantes	*Condições que devem ser satisfeitas antes e após efetuar o caso de uso. (Devem envolver somente dados, arquivos, recursos e estados alterados pelo caso de uso).*

Pré-condições	*Condições que devem ser satisfeitas antes de efetuar o caso de uso. (Invariantes + Pré-condições devem envolver todos os dados, arquivos, recursos e estados necessários para poder realizar todo o caso de uso).*
Acionamento	*Condição ou evento que inicia o caso de uso, disparado pelo ator principal.*
Fluxo principal	*Sequência de ações a serem executadas no caso normal de funcionamento.*
Fluxos alternativos	*Sequência de ações disparadas por eventos previstos durante a execução de ações do fluxo principal, ou de fluxos alternativos já descritos. (Fluxos alternativos devem indicar onde iniciam e onde terminam)*
Pós-condições	*Condições que devem ser satisfeitas após efetuar o caso de uso que termina da forma esperada. (Invariantes + Pós-condições devem envolver tudo o que resta ao final do fluxo normal. Pós-condições devem poder ser utilizadas como oráculos de teste).*
Garantia mínima	*Condições que devem ser satisfeitas sempre que o caso de uso não termine de forma normal.*
Requisitos	*Requisitos adicionais, tais como requisitos não funcionais e características de Interação Humano-Computador (IHC).*
Regras de negócio	*Especificação das regras (assertivas) a serem satisfeitas nos passos.*
Casos de uso correlatos	*Relação dos casos de uso correlacionados ao presente caso de uso (ex.: o conjunto de características que implementam um propósito).*
Artefatos correlatos	*Relação de documentos, artigos, mensagens, fontes de informação, etc. que contém alguma informação relevante para o caso de uso.*

Para que esta especificação se torne verificável, seus itens precisam ser descritos com mais formalidade (do que a descrição em linguagem natural e de forma livre) e de forma mais precisa, especificando detalhes sobre os elementos de interface envolvidos (Pinto, 2013). Como menciona Pinto (2013) os fluxos de um caso de uso devem ser escritos, preferencialmente, em um vocabulário restrito e semiestruturado, como o adotado por Staa (2017). A Tabela 3 apresenta um exemplo de caso de uso, onde cada fluxo visa descrever cada passo do fluxo de forma simples e sucinta.

Tabela 3. Descrição de um caso de uso para a funcionalidade "Efetuar Login".

Caso de Uso:	*Efetuar o login*
Objetivo:	*Obter a autorização de uso segundo os direitos de uso com os quais foi registrado.*
Ator:	*Usuário*
Pré-condição:	*1. Usuário na página contendo formulário de login;* *2. Base de dados com usuários cadastrados:*
Pós-condição:	*Sistema retorna autorização de uso.*
Acionamento:	*Quando o sistema solicitar o fornecimento de uma autorização de uso.*
Fluxo Principal:	*1. Sistema exibe página de login com todos elementos em branco* *2. Sistema gera CAPTCHA* *3. Usuário digita sua identificação;* *4. Usuário digita sua senha;* *5. Usuário digita o* CAPTCHA*;* *6. Usuário clica "Login";* *7. Sistema verifica retorno da validação do CAPTCHA;* *8. Sistema verifica se a identificação foi corretamente preenchida e se está cadastrada no sistema;* *9. Sistema verifica par <usuário, senha>;* *10. Sistema encerra login;* *11. Sistema retorna "autorização de uso" e os direitos de uso correspondentes a <usuário, senha>*
	Fluxos Alternativos
FA 1	Evento E1/6: Falha na verificação do CAPTCHA. i. O controle de acesso informa ao usuário que o CAPTCHA não foi preenchido corretamente. ii. Repete a partir de 5. Fim evento E1.
FA 2	Evento E2/7: Falha na verificação do Usuário i. Sistema informa ao usuário que o nome digitado não consta na base de dados ou está bloqueado ii. Repete a partir de 1. Fim evento E2.
FA3	Evento E3/8: Falha na verificação da senha. I. Sistema exibe mensagem de erro. II. Sistema incrementa o número de tentativas com erro para o usuário. III. Sistema verifica o número de tentativas IV. Repete a partir de 1. Fim evento E3.
FA5	Evento E5: Cancela operação I. Sistema redireciona página para a página inicial; II. Sistema retorna ao sistema e não é fornecida autorização de uso.

	Fim evento E4.

Regras de negócio	
Usuário	Elemento do tipo texto. 1. O nome do usuário deve ser fornecido. 2. O usuário deve estar cadastrado no sistema. 3. O usuário não pode ter cometido 3 erros seguidos (tentativas igual a 3)
Senha	Elemento do tipo texto. 1. A senha deve ser fornecida. 2. O par <usuário, senha> deve está correto
CAPTCHA	Componente que recebe um texto e retorna um booleano. O componente deve permitir ou negar caso o texto digitado não seja igual ao fornecido pela imagem.
Login	Elemento do tipo botão que dispara a verificação do Usuário e da Senha

2.3. Descrição de fluxos em casos de uso

Todos casos de uso têm pelo menos um fluxo definido, o **fluxo principal**, e pode ter outros fluxos que surgem a partir do principal, os **fluxos alternativos**, que podem levar a conclusão do caso de uso, o retorno a uma etapa de um fluxo anterior, ou ao fim do caso de uso após uma solicitação de cancelamento ou erro irrecuperável.

Um **Fluxo Principal**, como definido por (Heumman, 2001), representa uma sequência básica de eventos que devem ocorrer para que o caso de uso chegue a sua conclusão com sucesso. Esse fluxo provavelmente representará aquele comportamento do usuário que mais se espera que aconteça. Enquanto isso, um **Fluxo Alternativo**, visa cobrir um comportamento opcional ou excepcional.

2.4.
Geração de cenários a partir de um caso de uso

Segundo Bergmann (2003), cenários são considerados descrições evolutivas de situações num ambiente, sendo compostos por um conjunto ordenado de interações entre seus participantes. Um cenário, no contexto do caso de uso, é um "caminho" possível dentro do caso de uso, uma sequência de passos que, saindo de um evento inicial, chega à uma conclusão (Heumman, 2001). O fluxo principal é um exemplo de cenário e a cada combinação de fluxos, partindo do fluxo principal, pode ser vista também como um cenário. A Figura 1 apresenta uma visualização gráfica dos fluxos de um caso de uso genérico, onde cada linha representa uma sequência de eventos, FP significa Fluxo Principal e FA, Fluxo Alternativo.

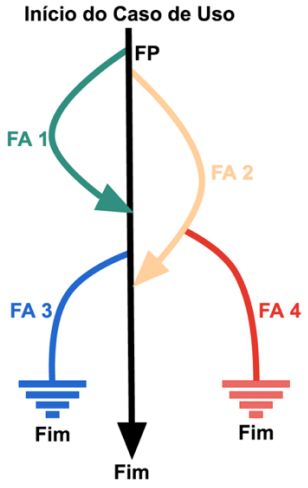


Figura 1. Representação gráfica de fluxos de um Caso de Uso. Adaptado de Heumann (2001).

Do caso de uso apresentado na Figura 1, é possível extrair os seguintes cenários:

Tabela 4. Cenários a partir das combinações de fluxos.

Cenários	Combinações
01	FP.
02	FP, FA 1.
03	FP, FA 1, FA 3.
04	FP, FA 3.
05	FP, FA 2.
06	FP, FA 2, FA 4.

Se o fluxo **FA 1** representasse um retrocesso (se a seta estivesse no sentido contrário), o número de combinações possíveis seria bem maior, potencialmente infinito, uma vez que esse fluxo estaria permitindo ciclos de eventos.

2.5. Exemplo de uso do software

Um exemplo é um cenário de uso do sistema, onde elementos são valorados e usuários definidos de forma que simule uma situação real. Ou seja, um exemplo é uma simulação de um cenário real, uma demonstração prática da execução de um cenário de uso. De acordo com Adzic (2009), ao construir exemplos é importante organizá-los de uma forma que seja possível identificar claramente quais são as pré-condições, quais são as etapas e quais são os resultados esperados do fluxo de trabalho, e divide um exemplo em 3 partes:

1. Dadas as *pré-condições.*
2. Quando *ações ou eventos acontece*m.
3. Então, consequências se verificam.

Seguindo essa abordagem, as Listagens 3 e 4 apresentam exemplos da aplicação dessa abordagem. Na Listagem 4, diferente do apresentado na Listagem 3, é utilizado uma notação tabular para resumir os exemplos. A notação tabular permite uma visão mais abrangente do cenário e ajuda a garantir que casos não serão esquecidos.

Entrega Grátis
A entrega grátis é oferecida para um cliente VIP desde que ele efetue uma compra superior a $50 e seja residente nos EUA.

Exemplos:

- **Dado que** Mark é um cliente VIP e reside nos EUA. **Quando** ele adiciona um arranjo de $ 50 de flores no carrinho e finaliza a compra. **Então**, na finalização será oferecida a ele a entrega gratuita.
- **Dado que** Mark é um cliente VIP, reside nos EUA e tem uma oferta de entrega gratuita não utilizada. **Quando** ele adiciona um arranjo de flores de US $ 30 para o carrinho de compras e vai para a finalização. **Então,** na finalização será oferecida a entrega gratuita.
- **Dado que** Mark é um cliente VIP e não tem nenhuma oferta de entrega gratuita não utilizada anteriormente. **Quando** ele adiciona um arranjo de flores de US $ 30 para o carrinho de compras e vai para a finalização. **Então,** na finalização, é oferecida uma entrega regular, independentemente dele residir ou não nos EUA.
- **(...)**

Listagem 3. Exemplo de exemplos de uso. Adaptado de Adzic (2009).

Entrega Grátis

A entrega grátis é oferecida para um cliente VIP desde que ele adicione uma determinada quantidade de livros no carrinho de compras. Clientes regulares e VIPs, com menos livros no carrinho que a quantidade determinada, não têm direito a entrega grátis.

Exemplos

Dado que a quantidade de mínima de livros no carrinho é de 10. Então é esperado que:

Tipo de Usuário	Número de Livros no carrinho	Entrega grátis?
VIP	8	Não
VIP	10	Sim
VIP	12	Sim
Comum	8	Não
Comum	14	Não
(...)	(...)	(...)

Listagem 4. Exemplo de exemplos de uso utilizando uma tabela. Adaptado de Adzic (2009).

O objetivo do uso de exemplos é consolidar uma interpretação de um cenário de uso e uniformizar o entendimento sobre eles. Assim como no BDD, os exemplos

são obtidos a partir de uma comunicação ativa entre time de desenvolvimento e clientes.

3
Trabalhos relacionados

O foco principal da dissertação é a geração automática de exemplos a partir da descrição textual de casos de uso, mas devido à pouca quantidade de publicações sobre a geração de exemplos de uso, a pesquisa teve um foco especial na geração de cenários de uso (que depois seriam parafraseados em exemplos) mapeados para construção de testes. Com isso, a pesquisa considerou trabalhos que tratavam de pelo menos um dos seguintes temas:

1. Uso de exemplo para a descrição de comportamentos
2. Uso de uma linguagem estruturada e limitada para a descrição de comportamento.
3. O uso de casos de uso para descrição e documentação de requisitos do software;
4. A geração automática de cenários de uso a partir da descrição de um modelo de comportamento do software;
5. O uso de exemplos como ferramenta auxiliar para a validação do comportamento do *software*

Neste contexto, vale ressaltar que não há a intenção de somente avaliar diferentes abordagens, mas também identificar as vantagens do uso de exemplos de uso do software como ferramenta de comunicação, documentação e validação de comportamentos.

3.1.
Uso de exemplos como ferramenta de comunicação

Adzic (2009) foi a primeira e a principal referência analisada sobre o uso de exemplos em métodos ágeis como meio de comunicação e validação de comportamentos. Em seu livro, Adzic apresenta um estudo e um relato de

experiência na utilização de exemplos como forma de especificação do software e a importância desses exemplos para a construção de casos de teste em um desenvolvimento dirigido por teste aceitação. Ele critica a forma simplória de descrição de comportamento, permitida no BDD proposto por North (2006), segundo ele: "Requisitos imperativos que comandam o que os usuários devem fazer sem demonstrá-lo, são especialmente ruins, mesmo quando parecem ser precisos, pois deixam espaço para erros". A justificativa é simples: cada um tem sua própria forma de pensar e de interpretar que afeta a maneira como entendemos essas declarações.

O objetivo principal do uso de exemplos, como proposto por Adzic (2009), é a construção de um entendimento compartilhado sobre o domínio que garanta que todos estarão "falando" uma mesma língua e compartilhando um mesmo entendimento consistente, onde o exemplo é de fato uma representação de um cenário concreto de uso, onde elementos são valorados e usuários identificados (como é mostrado na seção 2.5). Segundo o autor, os exemplos práticos transformados em teste de aceitação são, antes de tudo, dispositivos de comunicação, que no final das contas se tornam registros de como o sistema deve se comportar e, em último caso, como o sistema de fato se comporta uma vez que foi implementado. Para a construção e coleta de exemplos, Adzic (2009) propõe um evento, ao qual ele chama de *workshop*, que aconteceria antes do inicio do desenvolvimento e contaria com a presença de todos os envolvidos no projeto do software.

Como o objetivo dessa abordagem é facilitar a comunicação, é importante ter um cuidado com a quantidade de exemplos e observar a relevância, se o cenário descrito em um exemplo já está contido em outro, então esse exemplo tem pouca relevância. No entanto, um exemplo muito longo pode confundir ou tornar uma leitura cansativa. O mesmo é esperado quando a coleção de exemplos tem uma quantidade muito grande de exemplos que não agregam conhecimento (exemplos que ilustram o mesmo cenário de uso ou que já são cobertos por outros exemplos). Segundo Adzic (2009), o primeiro exemplo deve ter foco na história central ("*the spine story*") que contém a essência da historieta e os outros novos exemplos serão gerados a partir da adição de detalhes e exceções ao primeiro, mas deve se restringir ao escopo definido pela historieta.

Exemplos devem ajudar a identificar casos de “borda” (que exploram os limites definidos pelo comportamento), casos onde são permitidos retrocesso e repetição de etapas e cenários em que as coisas podem extrapolar o que era esperado daquele comportamento. Dessa forma, como cita Adzic, é interessante que exemplos identifique situações de:

i. Uso normal;
ii. Uso não convencional, mas razoável;
iii. Uso não convencional e não razoável.

O primeiro seria um cenário comum que logo leva ao objetivo. O segundo seria uma situação em que a funcionalidade seria utilizada de forma razoável, mas não da forma que seria considerada convencional. O último descreve algo que uma pessoa nunca faria, mas que é possível fazer.

Adzic, apesar de focar no uso de exemplos como ferramenta de comunicação, foca também no uso de exemplos como ferramenta para geração de teste de aceitação. Com isso, ele espera que a coleção de exemplos consiga cobrir um grande número de possíveis cenários de uso para que, dessa forma, seja possível gerar uma coleção de testes de aceitação que ofereça uma boa cobertura de código. Nesse sentido, quando o autor descreve a qualidade de um exemplo ele tem duas preocupações: a seleção de exemplos relevantes (apresentado em 2.5), e a cobertura do código pelos testes de aceitação gerados a partir dos exemplos.

3.2. Geração de cenários de uso a partir de modelos

Cenários de uso que descrevem um comportamento no BDD são, geralmente, feitos de forma manual e, por consequência, haverá uma tendência de que esses cenários representarão de forma pobre ou incompleta o comportamento, porque não há como garantir que os cenários pensados e escritos gerem testes de aceitação que garantam a cobertura de todo um comportamento (Li & Offutt, 2013). Como foco dessa dissertação é a geração de exemplos e não de testes, não haverá rigor no tratamento da cobertura de código, mas é importante explorar esse tema para

verificar a possibilidade de gerar exemplos que retratem os possíveis cenários de uso e que explorem comportamentos de forma abrangente.

Com um discurso voltado para a geração de testes de aceitação, Heumann (2001) aborda como a utilização da descrição textual de casos de uso pode contribuir para uma verificação prévia do software, na antecipação do processo de teste, e no desenvolvimento de uma metodologia de testes. Segundo ele, a descrição textual de casos de uso serve como meio de capturar o processo de negócio, documentar o comportamento do sistema e especificar seus requisitos. O processo proposto por Heumann (2001) se baseia na geração de testes de aceitação a partir de cenários concretos de uso do software que são extraídos da descrição textual de casos de uso, onde cenários são extraídos a partir da combinação de casos de uso. Nesse contexto, um cenário é um "caminho" possível dentro do caso de uso, uma sequência de eventos que, saindo de um evento inicial, chega a uma conclusão. Na abordagem, um caso de teste surge da tradução em código de uma possível concretização de um cenário de uso.

Similar ao proposto por Heumann (2001), Gutiérrez et al. (2008) apresenta um processo para a geração automática de diagramas UML de atividade a partir da descrição textual de casos de uso. Tanto a descrição textual do caso de uso quanto a representação em UML são uteis para a definição de cenários de uso. Segundo o autor, as notações gráficas para a descrição de cenários têm a vantagem de dar uma visão geral dos aspectos dinâmicos de um caso de uso. Além disso eles podem ser facilmente utilizados por uma ferramenta e ser analisados para fins como a geração de casos de teste. Por outro lado, gráficos podem ser difíceis de projetar ao contrário da descrição textual de casos de uso, onde cenários de uso são fáceis de escrever. Para facilitar o mapeamento e permitir a geração dos diagramas, o autor propõe a descrição do caso de uso utilizando XML, onde as *tags* definidas funcionam como especificação de elementos.

Wanderley et al. (2011) apresentam uma abordagem para a geração de modelos de requisitos a partir da descrição textual de casos de uso na forma de diagramas de processo e ontologia usando métodos baseados em linguística computacional. Para tanto é utilizada uma técnica de Processamento de Linguagem Natural (*Natural Language Processing*, NLP) para extrair processos de negócios dos casos de uso por meio de um modelo intermediário. De acordo com os

experimentos feitos pelos autores, o modelo de requisito apresentado na forma de diagrama de processo facilitou o entendimento e interpretação dos requisitos pelos clientes. Uma característica bem relevante dessa abordagem é que ela garante uma rastreabilidade entre modelos gerados e casos de uso.

Li & Offutt (2013) tratam o problema do mapeamento de testes abstratos em testes concretos, dentro do contexto de testes baseado em modelo (*Model-Based Test*, MBT), e apresenta uma abordagem para a criação automática de um mapeamento de testes abstratos em concretos, com um enfoque na: criação do mapeamento e na geração de valores de teste; na geração de caminhos a partir de um grafo, utilizando um critério de cobertura; na seleção de casos de testes; e na geração de testes concretos. O modelo em questão é representado por um diagrama UML de estados. Para verificação da abordagem, foi desenvolvido o STAL (*Structured Test Automation Language*), um framework para a conversão automática de diagramas UML de estados em uma suíte de testes. A solução apresentada foi avaliada a partir de experimentos que contou com a presença de 9 testadores e foram utilizados 17 programas. Os experimentos tinham como objetivo comparar a geração de testes utilizando o STAL com a geração manual de testes. Os resultados mostraram que utilizando o STAL o tempo gasto para a geração dos testes foi, em média, 26% do tempo gasto com geração manual. Mostraram também que 48 testes gerados manualmente estavam errados: testes concretos que não estavam corretamente relacionados aos testes abstratos.

Diepenbeck et al. (2013) apresentam um algoritmo para a geração automática de cenários que irão produzir testes que cobrem áreas ainda não testadas. Esses cenários são gerados com base nos outros cenários já existentes. De posse dos cenários previamente definidos, a abordagem cria uma estrutura de dados em grafo (um grafo de características) e explora esse grafo na procura por ramos não coberto por teste, quando encontrado, esse ramo fará parte de um novo cenário de uso que, posteriormente, dará origem a testes.

3.3.
Ferramenta de suporte ao BDD

Um objetivo do uso de exemplos é evitar ao máximo traduções, visto que ela pode gerar uma perda informação ou ser pouco precisa. Nesse sentido, Adzic enfatiza a importância de uma ferramenta que possa de forma automática converter exemplos em teste executáveis. Uma grande vantagem de tal ferramenta seria tornar possível que a atualização de uma documentação seja refletida automaticamente nos testes de aceitação. O trabalho do Adzic influenciou outros trabalhos como o de Wynnen & Hellesoy (2012) e a construção de ferramentas como o Cucumber (2016).

Wynnen & Hellesoy (2012) apresentam uma ferramenta de auxílio ao BDD, o Cucumber (Cucumber, 2016), que tem como proposta a combinação de testes automatizados de aceitação, requisitos funcionais e documentação em um formato que possa ser compreendido por pessoas envolvidas em um projeto de software sem conhecimento técnico. A ferramenta oferece uma linguagem de descrição de domínio própria, o Gherkin, para descrição de um cenário de uso do software em um texto semiestruturado, em inglês (também é possível utilizar outras línguas). O Gherkin oferece uma estrutura suficiente para que possa ser escrito de forma concisa um comportamento segundo o ponto de vista do usuário. A proposta é que cada comportamento esperado tenha uma descrição de cenários escritos em Gherkin e que, posteriormente, cada cenário dê origem a um conjunto de regras executáveis pela ferramenta Cucumber. A execução dessas regras irá verificar se de fato o sistema desenvolvido satisfaz ou não o comportamento esperado.

O Gherkin permite a descrição de cenários de uso no formato *given-when-then,* como apresentado no exemplo ilustrado na Listagem 5. Mas para que esse comportamento seja convertido em um teste de aceitação é necessário fornecer também como deve ser feito o mapeamento das informações, como apresentado na Listagem 6.

```
Feature: Link Click
Scenario: User clicks the link
Given I am on the homepage
When I click the provided link
Then I should see the link click confirmation.
```

Listagem 5. Descrição de um comportamento em Gherkin.

```
Given(/^I am on the homepage$/) do
  visit root_path
end
When(/^I click the provided link$/) do
  click_on "js-click-me"
end
Then(/^I should see the link click confirmation$/) do
  expect(page).to have_content("Link Clicked")
end
```

Listagem 6. Mapeamento em um teste de aceitação.

O Cucumber provê transparência sobre quais cenários de uso são cobertas pelos testes e como um cenário é mapeado em um teste de aceitação. No final das contas, o comportamento representado em Gherkin e interpretado pelo Cucumber serve como uma camada de abstração que permite a legibilidade e a edição de um exemplo de uso por pessoas sem conhecimento técnico.

Com o objetivo de gerar cenários de uso que explorem de forma abrangente os requisitos e que gerem testes de aceitação que garantam uma boa cobertura de código, Li et al. (2016) propõem o uso de práticas do Teste Baseado em Modelo para uma geração automática de cenário e apresentam o *Skyfire*, uma ferramenta voltada para o BDD onde as cenários de uso são gerados a partir de diagramas *UML* de estados. O cenário gerado representa um caminho possível dentro do diagrama de estados e está escrita em Gherkin para que possa ser utilizada no Cucumber.

Supsrisupachai (2009) propõe uma abordagem diferente e apresenta uma ferramenta para a paráfrase automática de cenários de uso escritos em uma linguagem natural, com uma estrutura e um vocabulário limitado, em diagramas de sequência. A justificativa é facilitar a leitura dos requisitos pelos clientes, que entendem com facilidade o cenário escrito, e pelos desenvolvedores que, segundo o autor, entendem melhor um cenário escrito na forma de diagramas. O seu objetivo é melhorar a comunicação do cliente com os desenvolvedores, oferecendo uma

ferramenta que funciona como um tradutor simultâneo, que permite que os cenários escritos pelos clientes sejam melhor entendidos pelos desenvolvedores. Uma proposta similar também é adotada por (Diepenbeck et al., 2013).

Wanderley et al. (2015) apresentam a avaliação de uma ferramenta, voltada para o BDD, para geração de cenários partir da descrição do comportamento em diagrama de mapa mental (*mind map*). A hipótese levantada é a de que o uso de mapas mentais para a modelagem de requisitos irá ajudar clientes a entender de forma mais fácil e, consequentemente, permitir que eles participem de uma forma mais efetiva do processo de desenvolvimento. Hipótese que foi confirmada após um experimento com 15 indivíduos sem conhecimentos técnicos.

Santos et al. (2010) apresentam uma ferramenta e um estudo que ilustra a aplicação de uma conversão de cenários escritos em diagrama de atividades para uma descrição em diagrama de sequência. O objetivo do estudo é mostrar a importância de uma abstração mais rica em detalhes na identificação rápida de erros e validação de comportamento pela equipe técnica. Com o uso do diagrama de sequência o tempo de análise e verificação de um cenário caiu quase que pela metade.

3.4. Comparativo

A solução proposta nesta dissertação verificou problemas, soluções e considerou as diversas ideias analisadas em outros trabalhos, reutilizou algumas e adaptou outras, um resumo é apresentado na Tabela 5. Importante ressaltar que as ideias que não contribuíram diretamente no projeto da solução proposta colaboraram reforçando a importância e a relevância da ideia proposta.

Tabela 5. Resumo sobre as ferramentas.

	Preocupação com a cobertura de código	Construção de um modelo intermediário	Geração de cenários escritos em linguagem natural	Suporte a cenários concretos de uso	Geração automática de cenários	Auxílio ao BDD
Wynnen & Hellesoy, (2012) (Cucumber)	Não	Não	Sim	Sim	Não	Sim
Li et al. (2016) (Skyfire)	Sim	Sim	Sim	Sim	Não	Sim
Supsrisupachai, (2009)	Sim	Sim	Não	Sim	Sim	Sim
Wanderley et al. (2015)	Sim	Sim	Sim	Sim	Não	Sim
Santos et al. (2010)	Não	Sim	Não	Sim	Não	Não

As principais influências, diferenças, contribuições e restrições são apresentadas a seguir.

3.4.1.
Principais influências

Alguns trabalhos tiveram uma importante influência na construção da solução:

- Como já foi citado, Adzic (2009) foi a principal referência no que tange ao uso de exemplos como ferramenta de comunicação, documentação e construção de testes de aceitação;

- Heumann (2009), Gutiérrez et al. (2008) e Wanderley et al. (2011) reforçaram a ideia da utilização da descrição textual de casos de uso como ferramenta para descrição de comportamento e requisitos funcionais.

- Pinto (2013) foi outra referência importante, a solução aqui proposta é um complemento à solução apresentada por ele. Com isso, a descrição textual dos casos de uso é a mesma utilizada por ele, que tem um layout de formulário como foi proposto por Staa (2017a), adaptado de Cockburn (2000).

- Li & Offutt (2013) e Pinto (2013) inspiraram a abordagem utilizada para a geração de cenários de uso concretos, valoração de elementos abstratos e a seleção de valores

- As abordagens apresentadas por Heumann (2009), Li & Offutt (2013) e Li et al. (2016) inspiraram a construção de um modelo intermediário para representação do caso de uso e geração de cenários.

- Wynnen & Hellesoy (2012) inspiraram o formato utilizado para a representação de exemplos e a construção da etapa de paráfrase de um cenário de uso em exemplo.

3.4.2.
Principais diferenças

As principais diferenças entre o que é proposto e o que foi pesquisado são:

- Quando o tema é geração de cenários, os trabalhos analisados que tratam do tema, exceto Santos et al. (2010), abordam a geração de cenários com foco na geração de testes e na cobertura de código. Diferente deles, além da preocupação com a geração de cenários, a solução proposta se preocupa também com a seleção de cenários que gerem uma coleção mínima de exemplos suficiente para uma representação abrangente de um comportamento descrito em um caso de uso. Ao invés da geração de uma grande massa de casos de teste, a solução foca na seleção de exemplos.

- Além do caso de uso descrevendo o comportamento a solução proposta permite que seja fornecido um conjunto de dados que simulam o conjunto de dados que estaria disponível em fonte de dados externo. O FunTester apresentado por Pinto (2013) permite o uso de fontes de dados externas na composição de regras de negócio. A solução aqui proposta faz uso de uma representação de dados que simula uma fonte de dados externa sem alterar as regras de negócio que fazem o uso da consulta a base de dados externas.

3.4.3.
Principais restrições

As principais restrições da solução proposta são:

- Diferente do proposto por Pinto (2013), a solução proposta não faz uso da combinação de casos de uso para a geração de cenários. Para cada caso de uso, individualmente, será gerada uma coleção de exemplos.

- O vocabulário utilizado na construção dos exemplos se restringe ao português, ao contrário do sugerido por Pinto (2013), que permite o uso de um vocabulário configurável.

- Não será feita uma validação da coerência entre as regras de negócio, ou seja, não será verificada se uma regra de negócio interfere ou invalida outra.

- A abrangência dos tipos de sistemas passiveis de geração de exemplos pela ferramenta se limita ao que é representável pelo FunTester: sistemas web onde a iteração entre usuário e sistema se restringe ao uso do mouse e teclado.

4
Metodologia

Este capítulo apresenta a solução proposta e as decisões de projeto relacionadas ao assunto da dissertação.

4.1
Visão geral

A Figura 2 apresenta o processo construído, que começa com a entrada dos casos de uso, criados pelo projetista ou analista (ou qualquer outra pessoa que tenha papel relacionado a especificação do projeto) com o auxílio do FunTester, e termina com a apresentação dos exemplos de uso gerados.

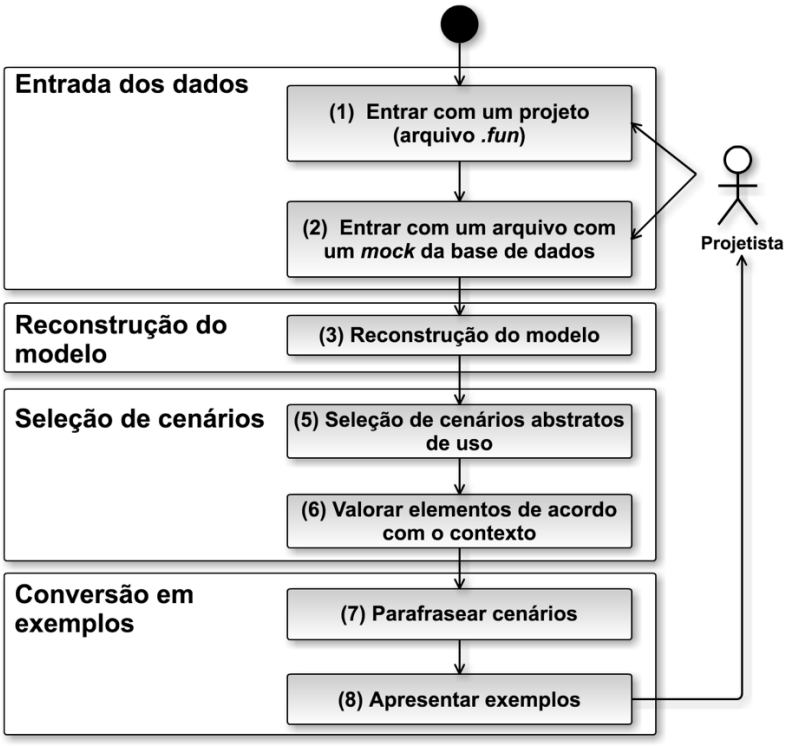


Figura 2. Processo para a Construção de Exemplos.

No processo ilustrado na Figura 2, temos as seguintes etapas:

1. **Entrar com um projeto (arquivo "*.fun*")**. Entrada de um arquivo criado com o FunTester.
2. **Entrar com um arquivo com um *mock* da base de dados**. Detalhamento, simulação, do conteúdo da base de dados que será consultada para validação de regras de negócio. Etapa facultativa que está condicionada à existência ou não de regras que façam uso de uma base de dados.
3. **Coleta dos casos de uso**. Reconstrução da informação, reconstrução e listagem dos casos de uso presentes no projeto.
4. **Identificar fluxos**. Para cada caso de uso, detalhamento dos fluxos principais e alternativos e como eles se relacionam.
5. **Seleção de cenários abstratos de uso**. Fazer a seleção de cenários de acordo com o relacionamento entre os fluxos, onde cenários são caminhos que podem ser percorridos por entre os fluxos identificados.
6. **Valorar elementos presentes em um cenário**. Conversão de um cenário abstrato em um cenário concreto com a valoração das variáveis presentes no cenário de acordo com o seu tipo, com as regras de negócio (seção 4.2.3) associada e com o contexto definido pelo cenário (condição de sucesso ou de falha).
7. **Parafrasear cenários (ou converter em exemplos)**. Reconstrução do cenário valorado em texto em linguagem natural.
8. **Apresentar resultados**. Apresentação dos exemplos para o usuário.

As etapas apresentadas podem ser vistas de uma forma macro como quatro fases distintas: (i) entrada de dados, etapas 1 e 2; (ii) reconstrução do modelo, etapas 3 e 4; (iii) seleção de cenários, etapas 5 e 6; (iv) conversão em exemplos, etapas 7 e 8. Nos tópicos seguintes, cada uma dessas fases e etapas serão apresentadas em detalhe.

4.2
Entrada de dados (*etapa 1*)

O FunTester representa os casos de uso em um nível rico de detalhe (como apresentado na seção 2.1). Dos campos disponibilizados pelo FunTester na representação de um caso de uso, utilizaremos apenas os campos exibidos na Tabela 6. O arquivo criado com o FunTester (extensão "*.fun"*), fornecido pelo projetista, contém todos os dados necessário para a reconstrução de um caso de uso.

Tabela 6. Campos da descrição de um Caso de Uso. Adaptado de Pinto (2013).

Campo	Utilizado para a geração de:	
	exemplos	testes
Nome	Sim	Não
Objetivo	Sim	Não
Importância	Não	Sim
Atores	Sim	Não
Precondições	Sim	Sim
Pós-condições	Sim	Sim
Acessível diretamente	Não	Sim
Fluxo Principal	Sim	Sim
Fluxos Alternativos	Sim	Sim
Regras de Negócio	Sim	Sim
Artefatos Correlatos	Sim	Não
Requisitos	Não	Não

A seção 5.1 apresenta exemplos práticos desta descrição textual. Na tabela 6 são apresentados dois campos que não estão listados na seção 2.2 e que serão apresentados a seguir:

- **Acessível diretamente**. Indica se há um acesso ao caso de uso pela interface, em outras palavras, se o evento que dispara o caso de uso é um elemento de interface. Importante para a geração de testes pelo FunTester, mas indiferente para a geração de exemplos.

- **Importância.** Utilizada pelo FunTester com viés para selecionar um conjunto de casos de uso para a construção de um conjunto de testes rápidos. Para a geração de exemplos esse campo não é relevante, visto que os exemplos são gerados por casos de uso indiferente à sua importância.

As subseções seguintes descrevem com detalhes a representação dos fluxos e das regras de negócio de um caso de uso.

4.2.1
Fluxos de um caso de uso

Seguindo a proposta de Kassel (2006), no FunTester, um fluxo de um caso de uso tem uma série de atributos que estão relacionados a questões como dificuldade de implementação, prioridade de construção e outras informações relevantes para o gerenciamento de atividades, no contexto de metodologias ágeis, e para a priorização na geração e execução de casos de testes (Pinto, 2013). Como um dos objetivos do uso de exemplos é ilustrar cenários possivelmente não imaginados, desconsideramos esses elementos e, dos atributos presentes em um fluxo, os únicos relevantes para a geração de exemplos são a sua sequência de passos e os atributos que estabelecem o relacionamento entre os fluxos. Mais detalhes sobre o relacionamento são apresentados na seção 4.4.

De acordo com o proposto por Heumann (2009), apresentado na seção 2.1, cada caso de uso é composto por um ou mais fluxos, onde cada fluxo representa uma sequência de passos. Todo caso de uso tem pelo menos um fluxo definido, o **fluxo principal**, e pode ter outros fluxos que surgem a partir do principal, os **fluxos alternativos**, que podem levar à conclusão do caso de uso, o retorno a uma etapa de um fluxo anterior, ou ao fim do caso de uso após uma solicitação de cancelamento ou erro irrecuperável.

No FunTester são definidos alguns possíveis tipos de fluxos alternativos, em que cada um deles tem pelos menos uma descrição (uma informação sobre o que comportamento o fluxo representa), e uma sequência de passos:

- **Fluxo Terminador** - um fluxo que é ativado após a conclusão de um passo de um outro fluxo qualquer e a execução dos passos descrito no fluxo leva à conclusão do caso de uso.
- **Fluxo Retornável** - assim como um fluxo terminador, ele é iniciado após um evento vindo de um outro fluxo, representa uma sequência de passos que, após a execução do último, retornará o fluxo da execução para um outro fluxo. Geralmente, seu retorno é para o fluxo que o originou.
- **Fluxo Cancelador** - um fluxo alternativo de exceção disparado pelo usuário a qualquer momento durante a execução do fluxo ao qual o fluxo cancelador está vinculado. O objetivo desse fluxo é interromper a execução do caso de uso sem que o objetivo seja alcançado.

Outros dois tipos de fluxos alternativos foram definidos por Pinto (2013), o **fluxo de exceção recuperável** e o **fluxo de exceção irrecuperável**, porém não foram implementadas na versão atual do FunTester, porque esses dois fluxos simulam condições (internas ou externas) que podem não ser fielmente traduzidas em testes funcionais. Devido a esse fato, esses tipos de fluxos serão desconsiderados para a geração de exemplo.

Os detalhes de uma sequência de passos serão apresentados na subseção a seguir.

4.2.1
Sequência de passos de um fluxo

Cada passo de um fluxo descreve uma ação que pode ser executada por um ator ou pelo sistema. No FunTester são definidos os seguintes tipos de passos:

1. **Ação.** Representa uma interação com o elemento (seção 4.2.2), quando o sistema ou ator realiza alguma operação sobre um elemento de interface gráfica, por exemplo, quando o sistema exibe uma janela, ou quando o usuário clica num botão. Um passo do tipo ação tem sempre um verbo atrelado a ele que indica qual é o tipo da ação, por exemplo: “clicar”, “selecionar” ou “digitar”.

2. **Verificação**. Representa um momento em que o sistema realiza uma verificação sobre as regras de negócio de um elemento de interface de valor editável (seção 4.2.3). Por exemplo, a validação de um e-mail digitado pelo usuário ou a verificação do formato de um texto digitado pelo usuário;
3. **Chamada a Caso de Uso**. Quando um passo representa a chamada de um outro caso de uso;
4. **Documentação**. Quando se deseja apenas registrar (documentar) uma ou mais operações que devem ser feitas internamente pelo sistema. Por exemplo, a geração de um relatório, ou a realização de um download de um arquivo.

4.2.2
Elementos

Um elemento definido no FunTester é um componente de uma interface gráfica do usuário (um *widget*) que pode ser definido como um elemento do tipo *button*, *textbox*, *combobox*, e assim por diante. Alguns desses elementos têm por objetivo receber dados do usuário e com isso gerar algum tipo de registro; esse tipo de elemento é classificado como elementos editável. Todo elemento editável deve especificar qual tipo de dado pode ser aceito por ele, podendo ser *String, Integer, Double, Boolean* e *DateTime.*

No FunTester um elemento, seja ele editável ou não, pode estar vinculado a uma ou mais regras de negócio, tópico que será abordado na subseção seguinte (4.2.3).

4.2.3
Regras de negócio

As regras de negócio são restrições necessárias para o negócio "acontecer". Quando o elemento é editável, a regra associada a ele tem por objetivo validar o dado fornecido pelo usuário. No FunTester, de acordo com Pinto (2013), as regras de negócio permitem:

a) Conhecer o que representa um determinado elemento (seu significado), para saber o que fazer com ele. Se o elemento for editável, por exemplo, podendo receber entrada de dados, será possível descrevê-lo com mais detalhes;
b) Inferir os possíveis valores que podem ser recebidos por elementos editáveis e gerá-los automaticamente;
c) Conhecer a expectativa de comportamento do sistema quando uma regra é infringida pelo usuário e gerar oráculos automaticamente;
d) Realizar a verificação automática destas regras, através de testes funcionais;
e) Interromper a necessidade do usuário de descrever fluxos alternativos para verificar estas regras de negócio, uma vez que a ferramenta gerará automaticamente valores para verifica-las

Desta forma, a definição das regras de negócio ao mesmo tempo habilita a geração automática de testes, geração de valores que satisfaçam ou não uma regra, e cria um benefício extra para a equipe de desenvolvimento, de não precisar descrever os vários fluxos alternativos que seriam necessários para tratar a verificação das regras de negócio (Pinto, 2013).

Na subseção seguinte é apresentado como é definido uma regra de negócio.

4.2.2.1
Definição de uma regra de negócio

Uma regra de negócio associada ao elemento possibilita informar uma ou mais faixas de valores, comprimentos, formatos, fonte de valores e etc. Atualmente as regras disponíveis são:

a) **Obrigatório**. Permite estabelecer se o elemento tem seu preenchimento obrigatório ou não;
b) **Valor Mínimo**. Permite estabelecer um valor mínimo para o dado fornecido;
c) **Valor Máximo**. Permite estabelecer um valor máximo para o dado fornecido;

d) **Comprimento Mínimo**. Permite estabelecer um comprimento mínimo para o dado fornecido (como limitar o tamanho do texto que pode ser inserido pelo usuário);
e) **Comprimento Máximo**. Permite estabelecer um comprimento máximo para o dado fornecido;
f) **Expressão Regular**. Permite definir o formato esperado para o dado fornecido a partir de uma expressão regular;
g) **Igual a**. Permite definir um único valor que pode ser aceito pelo elemento;
h) **Diferente de**. Permite definir um valor que não pode ser aceito pelo elemento;
i) **Presente na listagem**. Permite definir uma coleção de possíveis valores aceitos pelo elemento;
j) **Ausente na listagem**. Permite definir os possíveis valores não aceitos pelo elemento.

Para cada regra de negócio definida deve ser fornecida uma mensagem esperada para quando o valor informando pelo usuário não atender a essa Regra. O conteúdo dessa mensagem deve ser informativo o suficiente para descrever o porquê de o valor fornecido não ter atendido a regra. Espera-se que essa mesma mensagem seja exibida pela interface para o usuário. No FunTester, a presença da mensagem na interface é necessária para a validação ou não de um teste funcional gerado automaticamente.

Para cada Regra, com exceção da regra obrigatória, é possível ainda definir se o valor de referência para a comparação será definido manualmente, obtido de outro elemento do mesmo caso de uso ou obtido através de uma consulta a um banco de dados, sendo que as regras que fazem uso de expressão regular só podem ser definidas manualmente. A Tabela 7 apresenta um resumo sobre como podem ser definidos os valores de referência.

Tabela 7. Fonte dos valores de Referência. Adaptado de Pinto (2013).

Regra	Valor de referência:		
	Manual	**Consultado**	**De outro elemento**
Obrigatória	Não	Não	Não
Valor Mínimo	Sim	Sim	Sim
Valor Máximo	Sim	Sim	Sim
Comprimento Mínimo	Sim	Sim	Sim
Comprimento Máximo	Sim	Sim	Sim
Expressão Regular	Sim	Não	Não
Igual a	Sim	Sim	Sim
Diferente de	Sim	Sim	Sim
Presente na listagem	Sim	Sim	Não
Ausente na listagem	Sim	Sim	Não

Isto dá certa flexibilidade para a definição das regras e possibilita o uso de valores obtidos de bancos de dados confeccionados para teste usando dados com a mesma estrutura dos utilizados em produção. Adicionalmente, os parâmetros das consultas a banco de dados podem receber valores advindos da mesma estrutura, ou seja, uma consulta a banco de dados que possui uma restrição (*where*) dependente do valor de outro elemento, poderá recebê-lo no momento da geração de valores. Mais detalhes sobre a geração de valores a partir das regras de negócio são presentados nas seções 4.3 e 4.7 e exemplos serão exibidos no Capítulo 5.

4.3
Descrição da base de dados (*etapa 2*)

O uso de uma estrutura de dados para simular o conteúdo de uma base de dados externa ao sistema é algo que não é previsto pelo FunTester, já que ele possibilita o uso de uma base de dados real. Preferimos simular uma base de dados pois dessa forma simplificamos o processo e tornamos a ferramenta mais independente de componentes externos. Essa abordagem não impede que os casos

de uso construídos no FunTester façam uso de base de dados externas reais. Um usuário pode construir os casos de uso no FunTester com o objetivo de gerar, executar testes funcionais e aproveitar a mesma coleção de casos de uso para fazer a geração de exemplos, informando um conjunto mínimo de dados para a construção de uma base de dados. O objetivo é permitir que regras de negócios sejam exercitadas durante a geração de exemplos. Exemplo: supondo a existência de um elemento chamado "Usuário", que recebe valor do tipo *String*, que está associado à regra de negócio que faz a verificação da existência do valor digitado na base, para que essa regra seja exercitada é necessário no mínimo a existência de um valor correspondente na base de dados simulada que permita a execução com retorno da consulta.

O arquivo de entrada com os dados tem um formato do tipo CSV (*Comma-separated values*, valores separados por vírgula). O arquivo em si representa uma tabela que é esperada existir na base de dados real; cada coluna faz referência a uma coluna esperada e cada linha representa os valores possíveis. Exemplo, assumindo que a validação dos valores do par de elementos "usuário" e "senha" estejam condicionados à existência do mesmo par de valores na base de dados, para que seja possível exercitar essa regra é necessário que exista no mínimo um arquivo como descrito na Listagem 7. No arquivo, a primeira linha identifica as colunas e as linhas seguintes, os valores possíveis. O número de linhas, após a primeira, será proporcional à diversidade de exemplos gerados: quanto mais linhas, mais opções de valores na etapa de valoração de elementos.

```
usuario,senha
João,joao123
```

Listagem 7. Conteúdo do arquivo.

Essa base de dados simulada, na verdade, é uma base de dados real temporária. Para cada caso de uso será construído um banco de dados real temporário em SQLite (SQLite, 2017) que será destruída após a geração dos exemplos. O uso do SQLite permite a execução de consultas escritas seguindo o padrão SQL (ANSI/ISO/IEC, 2003). Duas grandes vantagens dessa tecnologia é a sua simplicidade e o fato de que a grande maioria dos comandos utilizados nas

consultas são compatíveis com qualquer outra ferramenta de banco de dados que também faz uso do padrão SQL. Mais detalhes são apresentados no exemplo apresentado na seção 5.1.

4.3
Coleta dos casos de uso (*etapa 3*) e identificação dos fluxos (*etapa 4*)

Para reconstrução do modelo que representa os casos de uso foi utilizado o pacote *funtester-core* (um componente do FunTester) (FunTester, 2017) implementados em Java. Com o uso desse componente foi possível reconstruir o objeto que representa um projeto de software e, a partir dele, coletar os objetos que representam os casos de uso. Do objeto que representa o projeto de software são coletados o perfil do projeto (que define a linguagem utilizada) os tipos possíveis de passos e as configurações de acesso e consulta à base externa de dados. O objeto definido para representar o caso de uso contém os elementos de interface disponíveis para o caso de uso e a sua coleção de fluxos.

Para cada caso de uso coletado, foi feito a identificação e detalhamento dos seus fluxos. Exceto o fluxo principal, o disparar de um fluxo alternativo está em um passo de um outro fluxo e a conclusão de um fluxo significa um retorno ao fluxo que o disparou. O fim da execução de um fluxo principal é também o fim da execução do caso de uso.

A geração de cenários abstratos será feita a partir da identificação e mapeamento do relacionamento entre os fluxos de um caso de uso, que será apresentado na subseção seguinte.

4.4
Geração de cenários abstratos (etapa 5)

Um cenário abstrato[3] é uma instância de um caso de uso, ou um caminho completo dentro do caso de uso, e que poderá dar origem a mais de um cenário concreto (seção 4.5). Cada cenário tem sua origem e fim no fluxo principal mas pode ser resultado da combinação do fluxo principal com fluxos alternativos.

A geração de cenários abstratos a partir da exploração de um caso de uso, assim como na exploração de um grafo direcionado, pode gerar infinitas cenários se houver pelo menos uma situação de recursão (ou ciclo), sem qualquer tipo de controle sobre o número de iterações. Para controlar o número de cenários gerados e evitar uma explosão combinatória, Heumann (2009) apresenta um processo para a geração de cenários a partir da combinação de fluxos, que influenciou a solução implementada por Pinto (2013) no FunTester. Na abordagem implementada no FunTester foram estabelecidas regras para controlar o comportamento exploratório do algoritmo de geração de cenários. Para a geração de cenários abstratos as regras serão ainda mais restritivas.

Tomando como exemplo o caso de uso ilustrado na Figura 3, onde cada estado é definido como um passo, e são ilustrados 4 fluxos:

- Um Fluxo Principal, FP;
- Um Fluxo Cancelador, FA1;
- Um Fluxo Retornável, FA2;
- Um Fluxo Terminador, FA3.

[3] Cenário expressa na forma de um modelo é chamado de cenário abstrato. Um cenário abstrato é definido a partir de elementos e objetos do modelo, mas que não pode ser executado em uma implementação. Por exemplo, se um modelo é expresso na forma de uma máquina de estados, um cenário abstrato é um caminho possível dentro do modelo. Cenário concreta é expresso em termos de implementação do modelo, representa um cenário real de uso, pronto para ser executado.

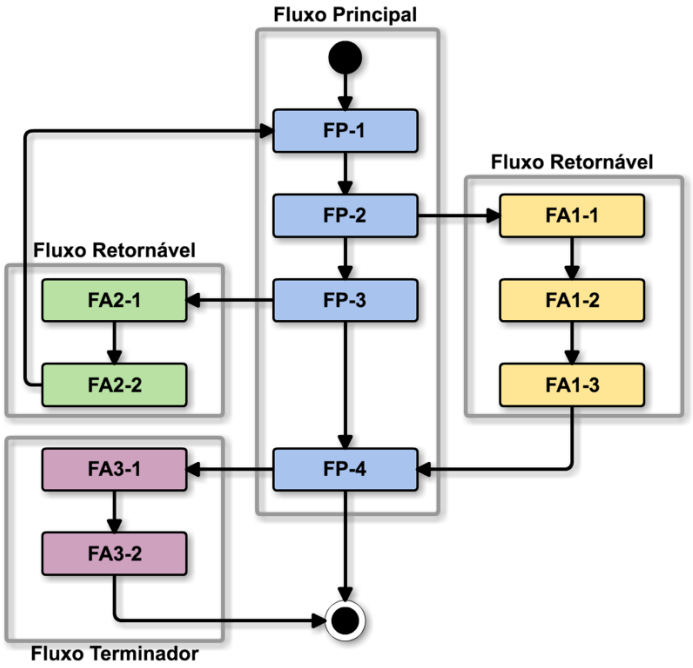


Figura 3. Diagrama de um caso de uso.

Apenas seguir o fluxo principal pode ser considerado um cenário abstrato. Seguir o fluxo principal mais o fluxo alternativo FA2 seria um outro cenário. O fluxo principal mais o fluxo FA1 seria um terceiro cenário e assim por diante. A Tabela 8 lista todas as combinações possíveis de cenários possíveis a partir do diagrama da Figura 3, começando no fluxo principal e o combinando com os outros fluxos.

Tabela 8. Cenários para o caso de uso ilustrado na Figura 3

Cenários	Combinações			
1	FP			
2	FP	FA1		
3	FP	FA3		
4	FP	FA1	FA3	
5	FP	FA2		
6	FP	FA2	FA1	
7	FP	FA2	FA3	
8	FP	FA2	FA1	FA3

O cenário 2 poderia ser escrita como a seguinte sequência de passos: FP-1, FP-2, FA-1, FA-2, FA-3, FP-4 e fim. No caso de uso ilustrado na Figura 2, o fluxo FA2 representa um retrocesso na execução do fluxo principal, que deu origem a uma série de outros possíveis cenários (os cenários 4, 5, 6, 7 e 8 da Tabela 8). Analisando o cenário 2, o retrocesso representa também um caso de recursão, onde a passagem sobre o fluxo FA2 pode se repetir resultando em uma longa sequência de passos (possivelmente infinita). Nesse sentido, um fluxo que representa um retrocesso, potencializa o número de combinações e, principalmente, o número de casos de recursões.

Mesmo quando a intenção é gerar cenários que serão convertidas em testes, o comportamento recursivo deve ser limitado para viabilizar a aplicação prática da geração de cenários. Como menciona Adzic (2009), para que exemplos sejam um meio efetivo de comunicação, é importante que a leitura dos exemplos não seja "cansativa" e para isso é interessante evitar ilustração de cenários com muita semelhança entre eles. Diante disso, para a limitação do comportamento recursivo e, consequentemente, limitação da geração de cenários abstratos semelhantes foram estabelecidas as seguintes regras:

1. Para cada fluxo definido no caso de uso será gerado pelo menos um cenário abstrato que o contenha;
2. Um fluxo retornável será combinado no máximo duas vezes, dará origem a no máximo dois cenários, sendo o segundo uma possível combinação dele com o fluxo principal e um outro fluxo alternativo.

Dessa forma, o número máximo de cenários abstratos será igual ao total de fluxos mais o número total de fluxos retornáveis. Seguindo essa abordagem, os cenários 7 e 8 não seriam gerados, mas a recursão seria exercitada com o cenário 6, e o número de cenários abstratos possíveis para o caso de uso ilustrado na Figura 3 seria de 6 cenários abstratos.

4.5
Conversão de um cenário abstrato em concreto (etapa 6)

Cada cenário abstrato dará origem a um cenário concreto que representa a execução de todos os passos até sua conclusão. Com isso o número total de cenários concretos será igual ao número de cenários abstratos. A atribuição de valores aos elementos de um cenário estará condicionada à existência de uma etapa futura de verificação e ao contexto dessa verificação.

Na subseção seguinte será apresentada a geração de valores para um elemento e, logo depois, na subseção 4.6 será apresentado o processo de conversão.

4.5.1
Geração de valores

No contexto da geração automática de testes, como apresenta Pinto (2013), o teste de software torna-se mais eficaz se os valores de teste são gerados baseados na análise das "condições de contorno" ou "condições limite". No contexto da geração automática de exemplos, os valores gerados serão valores gerados baseado com base em três possíveis situações de uso, como proposto por Adzic (2009) e apresentado na seção 3.1:

i. **Uso convencional**. Situação em que a verificação é satisfeita e o valor gerado é um valor intermediário. Por exemplo: se é esperado um valor numérico entre 0 e 100, será fornecido o valor 50.
ii. **Uso não convencional, mas razoável**. Situação em que a verificação é satisfeita e o valor gerado é próximo do limite. Por exemplo: se é esperado um valor numérico entre 0 e 100, será fornecido o valor 99.
iii. **Uso não convencional, não razoável.** Situação em que a verificação não é satisfeita e o valor gerado é um valor próximo de uma condição limite. Por exemplo: se é esperado que o usuário digite sobre um elemento textual um termo igual a “joão”, um valor não convencional, mas razoável, gerado será “joã” (o usuário esqueceu de digitar a última letra).

Essas três situações serão aplicadas levando em consideração qual é o tipo de valor que pode ser aceito pelo elemento e quais tipos de regras estão associadas a ele. A Tabela 9 apresenta quais situações são possíveis para cada tipo de regra.

Tabela 9. Situações exploradas de acordo com o tipo da regra.

Regra	Situações:		
	(i)	(ii)	(iii)
Obrigatória	Sim	Não	Sim
Valor mínimo	Sim	Não	Sim
Valor máximo	Sim	Não	Sim
Comprimento mínimo	Sim	Não	Sim
Comprimento máximo	Sim	Não	Sim
Expressão Regular	Sim	Não	Sim
Igual a	Sim	Não	Sim
Diferente de	Sim	Não	Sim
Presente na listagem	Sim	Não	Sim
Ausente na listagem	Sim	Não	Sim

Quando houver uma composição de regras relacionadas ao tamanho ou limites máximos e mínimos de um valor serão gerados valores considerando a situação (ii), e o valor gerado será um meio termo que satisfaça simultaneamente as duas regras.

Para cada elemento que precisa ter as regras verificadas serão gerados no máximo 2 valorações possíveis, um para um caso de sucesso e um outro para um caso de falha. Quando há uma composição de regras de negócio (quando um elemento está associado a mais de uma regra de negócio), o gerador de valores irá gerar valores considerando a seguinte ordem de relevância:

1. Regras que fazem uso de valores provenientes da base de dados, independentemente do tipo de regra;
2. Regra do tipo **diferente de**;
3. Regra do tipo **igual a**;
4. Regra do tipo **presente na listagem**;

5. Regra do tipo **ausente na listagem**;
6. Regras do tipo: **expressão regular, comprimento máximo, comprimento mínimo, tamanho máximo e tamanho mínimo**;
7. Regra do tipo **obrigatória**.

Dessa forma, se um elemento que aceita um valor do tipo *String* e possui três regras de negócio associadas a ele: uma que consulta a base de dados, uma que determina o tamanho máximo e outra que determina o tamanho mínimo, o valor gerado para esse elemento irá ser baseado apenas na primeira regra, a que tem maior relevância.

As regras do tipo expressão regular, comprimento máximo, comprimento mínimo, tamanho máximo e tamanho mínimo têm a mesma relevância. Na ocorrência de regras de mesma relevância será feita uma composição das regras. Exemplo: para um elemento que aceita um valor do tipo *string* que deve ter um formato definido por uma expressão regular, e ao mesmo tempo tem uma regra que define um comprimento mínimo, será gerado primeiro um texto, que será o resultado da combinação de caracteres que satisfazem a expressão regular, e depois será limitado o tamanho desse texto gerado, para satisfazer a condição relacionada ao tamanho mínimo.

Nesse trabalho não será feita a verificação da coerência entre as regras de negócio. Um exemplo de incoerência seria a existência de duas regras definidas para um elemento numérico, em que uma regra determina que o valor tem que ser maior que 10, e a outra determina que o valor deve ser igual a 10. Assumiremos que as regras foram corretamente definidas.

4.6
Conversão de um cenário abstrato em concreta (etapa 7)

Tomando como exemplo o caso de uso “Efetuar Login”, apresentado na Tabela 3 da seção 2.1. Um cenário abstrato que descreve uma falha após uma tentativa de login, em que o usuário apenas errou no momento de fornecer sua identificação, tem os passos da forma que é mostrada na Listagem 8.

1. Sistema exibe tela de login com todos elementos em branco
2. O usuário digita sua **identificação não correta**;
3. O usuário digita sua senha correta;
4. O usuário digita o *CAPTCHA* correto;
5. O usuário clica *Login*;
6. Sistema verifica *CAPTCHA*;
7. Sistema verifica identificação;
8. Sistema exibe erro: "**Usuário não cadastrado no sistema**".
9. Sistema exibe tela de login com todos elementos em branco
10. O usuário digita sua **identificação correta**;
11. O usuário digita senha correta;
12. O usuário digita o *CAPTCHA* correto;
13. O usuário clica *Login*;
14. Sistema verifica *CAPTCHA*;
15. Sistema verifica identificação;
16. Sistema verifica a senha
17. Sistema retorna permissão de uso de acordo com os direitos definidos para o usuário.

Listagem 8. Passos de um cenário abstrato.

Visto que o *CAPTCHA* é um elemento do tipo booleano (representa um componente externo que retorna apenas verdadeiro, caso tenha sido preenchido corretamente, e falso, caso contrário) e que a base de dados tem a seguinte tabela:

Tabela 10. Base de dados simulada.

Identificação	Senha
João	joao123
Maria	maria123

Para que esse cenário seja concretizado, representado uma situação de uso não convencional, é necessário saber quais serão os valores corretos para os elementos "Senha" e "CAPTCHA" e qual seria o valor errado para o elemento "Identificação". Como a verificação da senha está condicionada ao usuário, é preciso que primeiro seja definido um par senha e usuário válidos, valorar o elemento "Senha", condicionar o valor do elemento "Identificação" a representação do usuário definido anteriormente, e em seguida valorar o elemento "Identificação" com o valor definido com algum erro. Dessa forma o cenário concreto seria algo como mostrado na Listagem 9.

1. Sistema exibe tela de login com todos elementos em branco
2. **O usuário digita "Joã" no elemento identificação;**
3. O usuário digita "joao123" no elemento senha;
4. O usuário digita correto o CAPTCHA;
5. O usuário clica no elemento "Login";
6. Sistema verifica CAPTCHA;
7. Sistema verifica identificação;
8. Sistema exibe erro: "**Usuário não cadastrado no sistema**";
9. Sistema exibe tela de login com todos elementos em branco
10. **O usuário digita "João" no elemento identificação;**
11. O usuário digita "joao123" no elemento senha;
12. O usuário digita correto o CAPTCHA;
13. O usuário clica "Login";
14. Sistema verifica CAPTCHA;
15. Sistema verifica identificação;
16. Sistema verifica a senha
17. Sistema retorna permissão de uso de acordo com os direitos definidos para o usuário.

Listagem 9. Passos de um cenário concreto.

Com isso, na conversão de um cenário abstrato para um cenário concreto é importante verificar a dependência entre elementos, situações em que o valor de um elemento está condicionado ao valor exibido por outro. Para tanto, a valoração dos elementos é feita de trás para frente (o último elemento primeiro) e é mantida um mapa que armazena os valores aos quais os próximos elementos a serem valorados estarão condicionados.

No cenário concreto apresentado, uma alternativa simples para a valoração da identificação com erro seria gerar um valor aleatório, mas entendemos que, da forma como foi feito, removendo apenas a última letra do valor correto, representa uma situação com maior potencial de acontecer, e poderá facilitar um mapeamento das informações e a verificação visual do erro por parte de quem ler o exemplo.

No FunTester, o oráculo espera que quando houver um erro seja exibida uma mensagem informando a sua ocorrência. Não é prevista uma situação de recuperação do fluxo após erro. Na solução proposta, após a verificação de uma falha na validação das regras de negócio existirão duas possibilidades:

1. Finalização do caso de uso com a exibição da mensagem informando o erro, ou;
2. Disparo de um fluxo alternativo.

Por exemplo, na Figura 4, se o passo FP-2 (em destaque na figura) for um passo de verificação de regras, caso a verificação seja satisfeita a execução continuará sobre o fluxo principal executando o passo seguinte, o passo FP-3; no entanto, se a validação falhar, o passo seguinte a ser executado será o FA1-1, desviando a execução para o fluxo alternativo.

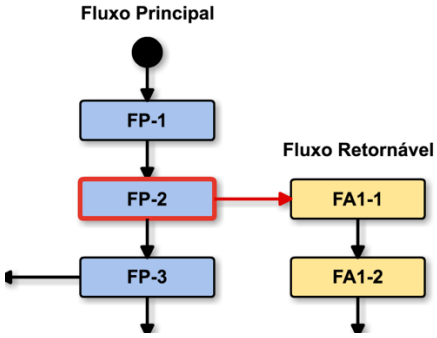


Figura 4. Desvio de fluxo após falha de verificação.

4.7
Paráfrase de um cenário concreto (*etapa 8*)

A estrutura proposta para a descrição de uma coleção de exemplos tem um formato semelhante ao utilizado para a descrição de uma coleção de cenários de uso concretos em Gherkin. A paráfrase de cenários concretos em exemplos se baseia na coleta e tratamento das informações e construção das seguintes sentenças:

- **Caso de uso**. Apresenta o nome do caso de uso que deu origem aos exemplos.
- **Base de dados**. Descreve o conjunto de dados que foram utilizados para a geração de valores.
- **Exemplo**. Descreve de forma breve o cenário que será apresentado.
- **Contexto inicial.** Descreve o estado do sistema imediatamente antes da execução do cenário.
- **Cenário.** Descreve a sequência de ações que compõe o cenário.
- **Resultado.** Descreve qual ação será disparada após a execução do cenário.

Serão apresentados nas subseções seguintes como cada uma dessas sentenças são construídas, exceto a referente ao caso de uso que uma simples cópia do nome do caso de uso.

4.7.1
Base de dados

Essa descrição é feita na forma de uma tabela e irá auxiliar o usuário na leitura dos exemplos, permitindo um mapeamento e verificação das informações apresentadas, o que permite uma transparência na forma como os elementos foram valorados. Essa informação tem o formato:

Dado que a base de dados contém: <*descrição da base de dados*>

Exemplo:

```
Dado que a base de dados contém:
  | usuário| senha   |
  | Joao   | joao123 |
  | Maria  | maria123|
```

Listagem 10. Exemplo de descrição da base de dados.

A <descrição da base de dados> é o resultado da conversão da base de dados local em uma representação textual utilizando uma notação tabular.

4.7.2
Exemplo

Essa informação oferece contexto e é resultado da composição das descrições dos fluxos (principal e alternativos) envolvidos no cenário. Quando se trata de um caso de sucesso, seu formato é o seguinte:

Exemplo: <descrição do fluxo básico> **com sucesso** (**após** <descrição fluxo alternativo>+)?**.**

Tomando como exemplo o cenário concreto apresentado na Listagem 9, essa sentença seria o derivado da combinação da descrição do fluxo principal “Efetuar

login" com a descrição do fluxo alternativo "Falha na identificação do usuário" e o resultado seria:

Exemplo: ***efetuar login*** *com sucesso após* ***falha na identificação do usuário****.*

Em uma situação de falha o formato é similar:

Exemplo: <descrição do fluxo básico> **sem sucesso** (**após** <descrição fluxo alternativo>+)?**.**

4.7.3
Contexto inicial

Essa informação lista as pré-condições necessárias para a execução do cenário, sendo que as pré-condições podem não ser explicitamente representadas, elas podem ser indiretamente informadas. Quando no caso de uso A, antes do primeiro passo que representa uma ação do usuário, há pelo uma chamada para um caso de uso B, as pós-condições de B serão pré-condições para A. Se após a chamada do caso de uso B houver uma chamada para o caso de uso C, as pós-condições de C também serão pré-condições de A. A construção do campo Cenário segue o seguinte formato:

Contexto inicial: {<pré-condições>[**,|e]** }+

Para o cenário concreto descrito na Listagem 2, gerado a partir Tabela 3, que tem duas pré-condições:

- "*Usuário na página de login";*
- *e "Base de dados com usuários cadastrados.*

A sentença para o campo cenário seria:

Contexto inicial: ***Usuário na página de login*** *e* ***base de dados com usuários cadastrados****.*

4.7.4
Cenário

Um cenário é a descrição da sequência de ações presentes no cenário, uma descrição da ação pode representar:

- Uma ação do usuário;
- Uma etapa de documentação que relata uma operação interna do sistema;
- Uma chamada de um caso de uso.

A descrição de uma ação do usuário varia de acordo com o tipo da ação, uma ação do usuário tem o formato:

Ação com atribuição de valor a elemento do tipo não booleano:

usuário *<verbo> <valor>* ***no elemento*** *<nome do elemento>****;***

Ação correta com atribuição de valor a elemento do tipo booleano:

usuário *<verbo>* ***correto o*** *<nome do elemento>****;***

Ação não correta com atribuição de valor a elemento do tipo booleano:

usuário <verbo> **não correto** <nome do elemento>**;**

Ação de documentação do sistema:

sistema *<descrição da documentação>****;***

Ação de chamada de caso de uso:

é chamado *<nome do caso de uso>****;***

O *<verbo>* referenciado é o verbo vinculado à ação descrita no caso de uso. No FunTester é definido um conjunto padrão de verbos, mas outros podem ser criados e adicionados ao caso de uso. Exemplos de verbo seria: *digitar*, *clicar*, *selecionar* e etc. O <nome do elemento> é o nome do elemento como ele é apresentado na interface. A descrição da documentação é uma descrição de uma

ação do sistema, como por exemplo: “verifica a base de dados”. Um exemplo de descrição de um cenário pode ser visto na Listagem 9.

4.5.4
Resultado

Um resultado é a descrição da conclusão da execução de um fluxo, onde são apresentadas as pós-condições e, consequentemente, o objetivo alcançado pelo usuário, no caso de sucesso, ou o detalhamento do erro no caso de falha. No caso de sucesso o formato é:

então: retorna ao sistema (<pós condições>**,**)+**.**

No caso de falha:

então, não retorna (<pós condições>**,**)+**.**

No caso de cancelamento:

Então, operação é cancelada e sistema não retorna (<pós condições>**,**)+**.**

4.9
Exibição dos exemplos (etapa 9)

Seguindo um formato similar ao da descrição de um cenário de BDD apresentado na seção 2.1 a nossa proposta é exibir os exemplos seguindo este formato:

```
Caso de Uso: <nome do caso de uso>

Dado que a base de dados contém: <descrição da base de dados>
Exemplo: <descrição do fluxo básico> com sucesso (após <descrição fluxo
alternativo>+)?.
Contexto inicial: {<pré-condições>[,|e] }+
Cenário:
sistema <descrição da documentação>;
usuário <verbo> correto o <nome do elemento>;
[mais ações] ...
então, não retorna (<pós condições>, )+.

... [outros exemplos]
```

Listagem 11. Formato de uma apresentação.

Tomando ainda como exemplo o caso de uso apresentado na Tabela 3 da seção 2.2, a Listagem 5 apresenta um trecho apresentação do caso de uso convertido em exemplos.

```
Caso de Uso: Efetuar o login

Dado que a base de dados contém:
  | tentativas|   user|      pass|
  |          3| daniel| daniel123|
  |          0|   joao|   joao123|
  |          1|  maria|  maria123|

Exemplo: efetuar login com sucesso.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'daniel' no elemento Usuário,
  usuário digita 'daniel123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Validar,
  sistema valida 'CAPTCHA',
  usuário clica sobre o elemento Login,
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, retorna ao sistema autorização de uso.

Exemplo: efetuar login com sucesso após falha na verificação do CAPTCHA.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'maria' no elemento Usuário,
  usuário digita 'maria123' no elemento Senha,
  usuário digita não correto o CAPTCHA,
  usuário clica sobre o elemento Validar,
  sistema não valida 'CAPTCHA', identifica erro: 'O CAPTCHA não foi
corretamente preenchido.'
  sistema gera novo CAPTCHA,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Validar,
  sistema valida 'CAPTCHA',
  usuário clica sobre o elemento Login,
  sistema valida 'Usuário',
```

```
  sistema valida 'Senha',
  sistema encerra login,
então, retorna ao sistema autorização de uso.

{mais exemplos} ...
```

Listagem 12. Exemplo de uma apresentação.

4.9
Resumo do processo

A Figura 5 apresenta um diagrama detalhado com um resumo do processo de geração de exemplos a partir da descrição textual de casos de uso.

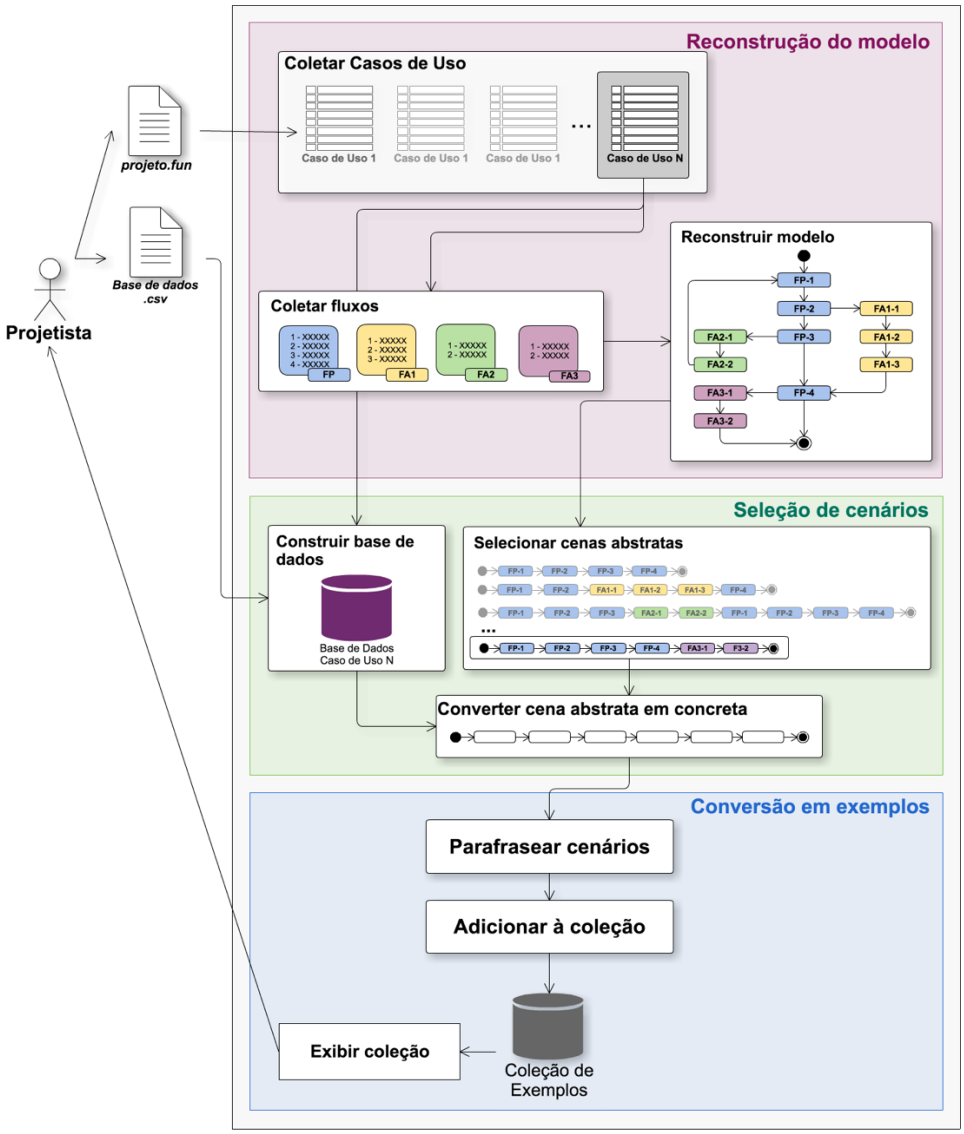


Figura 5. Resumo do Processo.

5
Prova de conceito

Este capítulo apresenta uma demonstração prática do uso do FunTester junto com o gerador de exemplos para a conversão da descrição textual de um caso de uso em uma coleção de exemplos práticos de uso. Serão apresentadas algumas características pontuais sobre a ferramenta implementada em complemento aos pontos levantados no Capítulo 4.

5.1
Casos de uso utilizados

Como demonstração prática foram criados com auxílio do FunTester dois casos de uso que representam as seguintes funcionalidades:

- Efetuar login (apresentado na Tabela 11). Representa uma página destinada à verificação de permissão de uso.
- Concluir Pedidos (apresentado na Tabela 13). Que representa uma página de verificação de um carrinho de compras, uma etapa anterior ao pagamento.

Esses casos de uso foram construídos inspirados em funcionalidades comuns de serem encontradas em sistemas web. Cada um dos casos de uso será apresentado nas duas seções seguintes e logo após serão discutidos alguns pontos relevantes.

5.1.1
Efetuar login

Representa um componente de verificação e validação de usuário, um componente genérico, que prevê serviços típicos de um login realizado com a ajuda

de um teclado, e que pode ser utilizado em vários sistemas. Na Figura 6 é apresentada uma ilustração desse componente. Para a validação de um usuário acontecer ele deve fornecer sua identificação, senha e preencher o CAPTCHA, e para que o componente retorne ao sistema o *status* de "uso autorizado", primeiro, o CAPTCHA deve ser corretamente preenchido, a identificação fornecida deve estar na base de dados do sistema e a senha fornecida deve casar com a senha presente na base de dados vinculada à identificação. Caso um usuário forneça uma identificação correta e uma senha errada, isso será considerado uma tentativa com falha que deve ser contada após 3 tentativas seguidas de falhas o usuário deve ter seu acesso bloqueado, não poderá mais fazer o acesso, comportamento comum na validação de usuário de sistemas bancários para evitar o uso indevido de uma conta.

Figura 6. Componente web "Efetuar Login".

O caso de uso tem um fluxo principal e quatro fluxos complementares, o caso de uso é apresentado na Tabela 11.

Tabela 11. Caso de uso "Efetuar Login".

Caso de Uso:	*Efetuar o login*
Objetivo:	*Obter a autorização de uso segundo os direitos de uso com os quais foi registrado.*
Ator:	*Usuário*
Pré-condição:	*3. Usuário na página contendo formulário de login;* *4. Base de dados com usuários cadastrados:*
Pós-condição:	*Componente retorna autorização de uso.*

Acionamento:	*Quando o sistema solicitar o fornecimento de uma autorização de uso.*
Fluxo Principal:	*12. Sistema exibe página de login com todos elementos em branco* *13. Sistema gera CAPTCHA* *14. Usuário digita sua identificação;* *15. Usuário digita sua senha;* *16. Usuário digita o* CAPTCHA*;* *17. Usuário clica "Login";* *18. Sistema verifica retorno da validação do CAPTCHA;* *19. Sistema verifica se a identificação foi corretamente preenchida e se está cadastrada no sistema;* *20. Sistema verifica par <usuário, senha>;* *21. Sistema encerra login;* *22. Componente retorna "autorização de uso"*
	Fluxos Alternativos
FA 1	Evento E1/6: Falha na verificação do CAPTCHA. iii. O controle de acesso informa ao usuário que o CAPTCHA não foi preenchido corretamente. iv. Repete a partir de 5. Fim evento E1.
FA 2	Evento E2/7: Falha na verificação do Usuário iii. Sistema informa ao usuário que o nome digitado não consta na base de dados ou está bloqueado iv. Repete a partir de 1. Fim evento E2.
FA3	Evento E3/8: Falha na verificação da senha. V. Sistema exibe mensagem de erro. VI. Sistema incrementa o número de tentativas com erro para o usuário. VII. Sistema verifica o número de tentativas VIII. Repete a partir de 1. Fim evento E3.
FA5	Evento E5: Cancela operação III. Sistema redireciona página para a página inicial; IV. Sistema retorna ao sistema e não é fornecida autorização de uso. Fim evento E4.

Os elementos presentes na interface para este caso de uso são:

- **Usuário**. Elemento que recebe valor do tipo textual destinado para que o usuário forneça sua identificação. Possui duas regras de negócio:
  1. O Elemento deve ser preenchido;
  2. O valor fornecido deve estar presente no resultado da consulta ao banco "*SELECT usuario FROM usuarios WHERE usuario = ? and tentativas < 3;*", sendo a interrogação substituída pelo valor fornecido.

- **Senha**. Elemento que recebe valor do tipo textual destinado para que o usuário forneça sua identificação. Possui duas regras de negócio:
  1. O Elemento deve ser preenchido;
  2. O valor fornecido deve estar presente no resultado da consulta ao banco "*SELECT senha FROM usuarios WHERE usuario = ? and senha = ?;*", sendo a primeira interrogação substituído pelo valor presente no elemento Usuário e a segunda, pelo valor no elemento Senha.
- **CAPTCHA**. Elemento que funciona com um componente externo, no qual o usuário digita o texto exibido pelo componente e este retorna o resultado da validação (um booleano);
  1. O Elemento deve ser preenchido e retornar verdadeiro ou falso.
- **Login**. Elemento de interface do tipo botão;
- **Cancelar.** É tratado como um elemento de interface, mas na verdade representa uma ação que pode levar ao cancelamento da operação, exemplo: fechar o browser ou voltar a página.

Como pré-condições, o usuário deve estar na página de login e deve existir uma base de dados com usuários cadastrados. Para a geração de exemplos será fornecido para construção da base de dados um arquivo com os valores exibidos na Tabela 12. Nessa base de dados, há uma coluna com o número de tentativas seguidas de falhas.

Tabela 12. Base de dados, caso de uso "Efetuar Login".

usuario	senha	tentativas
maria	maria123	1
marcia	marcia123	3
joao	joao123	1
daniel	daniel123	1

5.1.1
Concluir pedido

Esse caso de uso descreve uma página de conclusão de compra, onde o usuário verifica os itens presentes no carrinho e pode editá-los, alterar quantidade ou remover o item. A Figura 7, ilustra bem o objetivo dessa funcionalidade, na figura é ilustrada uma página de finalização de pedido de um site de supermercado.

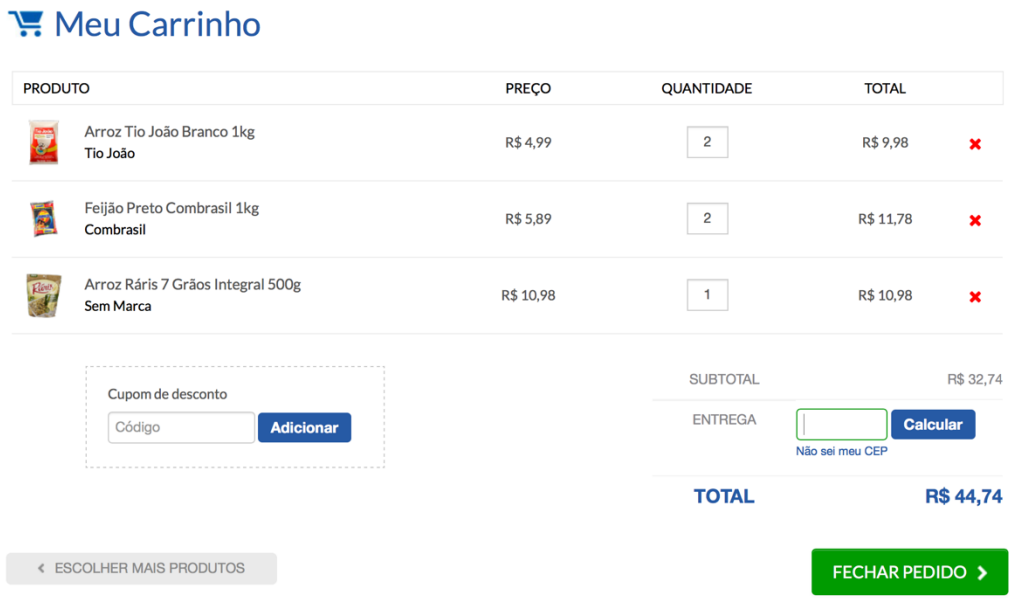


Figura 7. Concluir Compra.

O caso de uso que descreve essa funcionalidade é apresentado na Tabela 13. Nele há um fluxo principal e 7 fluxos alternativos, sendo seis deles retornáveis.

Tabela 13. Descrição de um caso de uso para a funcionalidade "Concluir Pedido".

Caso de Uso:	**Concluir Compra**
Objetivo:	*Confirmar itens presentes no carrinho*
Ator:	*Usuário*
Pré-condição:	*0. Usuário na página do carrinho de compras;* *1. Carrinho com pelo menos um item.*
Pós-condição:	*Sistema retorna a seleção de pedidos.*
Fluxo Principal:	*1. Sistema exibe itens do carrinho;* *2. Usuário digita CEP para entrega;* *3. Usuário clica em Validar CEP;*

	4. Sistema checa as regras de negócio de CEP para entrega; *5. Sistema informa o valor do frete;* *6. Sistema adiciona o valor do frete ao valor total da compra;* *7. Usuário clica em Fechar Pedido;* *8. Sistema prossegue para o pagamento.*
	Fluxos Alternativos
FA 1	Evento E1/4: Falha na verificação do CEP. I. Sistema exibe mensagem de erro II. Repete a partir de 2. Fim evento E1.
FA 2	Evento E2/1: Atualizar quantidade de um item. I. Usuário seleciona Item; II. Usuário digita Quantidade; III. Sistema checa as regras de negócio de Quantidade; IV. Sistema atualiza a quantidade; V. Sistema atualiza valor total da compra; VI. Repete a partir de 1. Fim evento E2.
FA 3	Evento E3/E2/III: Falha na verificação da quantidade disponível de um item. I. Sistema exibe mensagem de erro; II. Sistema informa quantidade disponível III. Repete a partir de 1. Fim evento E3.
FA 4	Evento E4/1: Inserir cupom de desconto. I. Usuário digita Cupom de Desconto II. Usuário clica Validar Cupom III. Sistema checa as regras de negócio de Cupom de Desconto IV. Sistema atualiza valor total da compra V. Sistema exibe novo valor total VI. Repete a partir de 1. Fim evento E4.
FA 5	Evento E5/4: Falha ao validar cupom de desconto. I. Sistema exibe mensagem de erro II. Repete a partir de 2. Fim evento E5.
FA 6	Evento E6/1-6: Cancelar conclusão para comprar mais. I. Usuário clica “Escolher mais produtos”. II. Sistema retorna para a página inicial do sistema Fim evento E6.
FA 7	Evento E7/1: Remover item. I. Usuário seleciona “Item” II. Usuário clica “Remover item” III. Sistema atualiza carrinho de compras IV. Sistema atualiza valor total Fim evento E7.

Os elementos presentes na interface para este caso de uso são:

- **Item**. Elemento que recebe valor do tipo textual que identifica um item presente no carrinho do usuário. Possui duas regras de negócio:
  1. O Elemento deve ser preenchido;
  2. O valor fornecido deve estar presente no resultado da consulta ao banco "*SELECT item FROM carrinho WHERE item = ?;*", sendo a interrogação substituída pelo valor armazenado no elemento.
- **Quantidade**. Elemento que recebe valor do tipo numérico que indica a quantidade de um Item no carrinho. Possui duas regras de negócio:
  1. O Elemento deve ser preenchido;
  2. O valor fornecido deve ser menor o valor do resultado da consulta ao banco "*SELECT quantidade FROM carrinho WHERE item = ?;*", sendo a interrogação substituída pelo valor presente no elemento.
- **Cupom de Desconto**. Elemento que funciona com um componente externo, no qual o usuário digita o texto exibido pela componente e a componente retorna o resultado da validação (um booleano);
- **CEP**. Elemento que recebe valor do tipo textual. Possui duas regras de negócio:
  1. O Elemento deve ser preenchido;
  2. Deve ter um comprimento de 9 caracteres;
  3. O valor fornecido deve satisfazer a expressão regular "[0-9]{5}-[0-9]{3}".
- **Fechar Pedido**. Elemento de interface do tipo botão;
- **Escolher mais produtos.** É tratado como um elemento de interface, mas na verdade representa uma ação que pode levar ao cancelamento da operação, exemplo: fechar o browser, voltar a página, acessar um

Como pré-condições, o usuário deve estar na página de conclusão de compra e deve existir uma base de dados listando os itens que estão no carrinho. Para a geração de exemplos será fornecido, para construção da base de dados, um arquivo com a seguintes tabela de valores:

Tabela 14. Base de dados, caso de uso “Concluir Pedido”.

id	item	valor	total_selecionado	total_estoque
1	Produto A	4.99	2	423
2	Produto B	6.80	3	125
3	Produto C	5.89	2	13
4	Produto D	5.29	1	17

5.2
Geração de exemplos

A partir desse caso de uso, de acordo com o descrito no capítulo anterior, foram gerados 8 exemplos para o caso de uso “Efetuar Login” que retratam:

1. Efetuar login com sucesso.
2. Efetuar login com sucesso após falha na verificação do CAPTCHA.
3. Efetuar login com sucesso após falha na verificação do CAPTCHA e falha na verificação do usuário.
4. Efetuar login com sucesso após falha na verificação do usuário.
5. Efetuar login com sucesso após falha na verificação do usuário e falha na verificação do CAPTCHA.
6. Efetuar login com sucesso após falha na verificação da senha.
7. Efetuar login com sucesso após falha na verificação da senha e falha na verificação do CAPTCHA.
8. Efetuar login sem sucesso após cancelar operação.

Para ilustrar, os exemplos 1 e 2 do caso de uso “Efetuar Login” são apresentados nas Listagens 1 e 2. O primeiro, deriva do fluxo principal e o segundo é o resultado da combinação do fluxo principal com um fluxo alternativo.

```
Dado que a base de dados contém:
  |usuario|     senha| tentativas|
  |  maria|  maria123|          1|
  | marcia| marcia123|          3|
  | daniel| daniel123|          1|
  |   joao|   joao123|          1|

Exemplo: efetuar login com sucesso.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'maria' no elemento Usuário,
  usuário digita 'maria123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna a autorização de uso.
```

Listagem 13. Exemplo: Efetuar login com sucesso.

```
Dado que a base de dados contém:
  |   user|      pass| tentativas|
  |  maria|  maria123|          1|
  | marcia| marcia123|          3|
  | daniel| daniel123|          1|
  |   joao|   joao123|          1|

Exemplo: efetuar login com sucesso após falha na verificação do CAPTCHA.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'joao' no elemento Usuário,
  usuário digita 'joao123' no elemento Senha,
  usuário digita não correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  não valida 'CAPTCHA', identifica erro: 'O CAPTCHA não foi corretamente
preenchido.'
  sistema gera novo CAPTCHA,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna a autorização de uso.
```

Listagem 14. Exemplo: Efetuar login com sucesso após falha na verificação do CAPTCHA.

A partir do caso de uso “Concluir Pedidos” foram gerados 13 exemplos que retratam:

1. Concluir pedidos com sucesso.
2. Concluir pedidos com sucesso após falha na verificação do CEP.
3. Concluir pedidos com sucesso após atualizar quantidade de um item.
4. Concluir pedidos com sucesso após atualizar quantidade de um item e falha na verificação do CEP.
5. Concluir pedidos com sucesso após atualizar quantidade de um item e falha na verificação da quantidade disponível de um item.
6. Concluir pedidos com sucesso após atualizar quantidade de um item, falha na verificação da quantidade disponível de um item e falha na verificação do CEP.
7. Concluir pedidos com sucesso após inserir cupom de desconto.
8. Concluir pedidos com sucesso após inserir cupom de desconto e falha na verificação do CEP.
9. Concluir pedidos com sucesso após inserir cupom de desconto e falha ao validar cupom de desconto.
10. Concluir pedidos com sucesso após inserir cupom de desconto, falha ao validar cupom de desconto e falha na verificação do CEP.
11. Concluir pedidos sem sucesso após cancelar conclusão para comprar mais.
12. Concluir pedidos com sucesso após remover item.
13. Concluir pedidos com sucesso após remover item e falha na verificação do CEP.

Para ilustrar, os exemplos 1 e 3 do caso de uso “Concluir Pedido” são apresentados nas Listagens 15 e 16. O primeiro deriva do fluxo principal e o segundo é o resultado da combinação do fluxo principal com um fluxo alternativo.

```
Dado que a base de dados contém:
  | id|      item|  valor| total_selecionado|total_estoque|
  |  3| Produto C|   5.89|                 2|           13|
  |  4| Produto D|   5.29|                 1|           17|
  |  1| Produto A|   4.99|                 2|          423|
  |  2| Produto B|   6.80|                 3|          125|

Exemplo: concluir pedidos com sucesso.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita '22740-443' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
então, sistema retorna a seleção de pedidos.
```

Listagem 15. Exemplo: Concluir pedidos com sucesso.


```
Dado que a base de dados contém:
  | id|      item|  valor| total_selecionado|total_estoque|
  |  3| Produto C|   5.89|                 2|           13|
  |  4| Produto D|   5.29|                 1|           17|
  |  1| Produto A|   4.99|                 2|          423|
  |  2| Produto B|   6.80|                 3|          125|

Exemplo: concluir pedidos com sucesso após atualizar quantidade de um
item.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário seleciona 'Produto B' no elemento Item,
  usuário digita '127' no elemento Quantidade,
  não valida 'Quantidade', identifica erro: 'A quantidade solicitada é
maior que a disponível em estoque'
  sistema atualiza a quantidade,
  sistema atualiza valor total da compra,
  usuário digita '22639-054' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
então, sistema retorna a seleção de pedidos.
```

Listagem 16. Exemplo: Concluir pedidos com sucesso após atualizar quantidade de um item.

5.3
Exploração do caso de uso

Como apresentado na seção 4.4, todos os fluxos foram considerados na geração de cenários. Todo fluxo alternativo foi combinado com os fluxos com os quais relacionam. No exemplo 10 do caso de uso "Concluir pedido" retrata uma situação que foi resultado da combinação de três fluxos: o fluxo principal "concluir pedido", o fluxo alternativo "inserir cupom de desconto" e o fluxo alternativo "falha ao validar cupom de desconto", o resultado é o exemplo apresentado na Listagem 17.

```
Dado que a base de dados contém:
  | id|      item|  valor| total_selecionado|total_estoque|
  |  3| Produto C|   5.89|                 2|           13|
  |  4| Produto D|   5.29|                 1|           17|
  |  1| Produto A|   4.99|                 2|          423|
  |  2| Produto B|   6.80|                 3|          125|

Exemplo: concluir pedidos com sucesso após inserir cupom de desconto e
falha ao validar cupom de desconto.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita não correto o Cupom de Desconto,
  usuário clica sobre o elemento Validar Cupom,
  sistema não valida 'Cupom de Desconto', identifica erro: 'Não foi
possível validar o cupom'
  sistema exibe mensagem de erro,
  usuário digita '22887-894' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
então, sistema retorna a seleção de pedidos.
```

Listagem 17. Exemplo: concluir pedidos com sucesso após inserir cupom de desconto e falha ao validar cupom de desconto.

Exemplos de combinação de até 3 fluxos estão presentes nos exemplos 3, 5 e 7 do caso de uso "Efetuar Login" e nos exemplos 5, 6, 7, 9, 10, 11 e 14 do caso de uso "Concluir Pedido" (Todos os exemplos são apresentados no Apêndice A). O comportamento recursivo foi exercitado sempre que necessário e em quase todos os exemplos houve a combinação de casos de uso.

5.4
Atualização de um caso de uso

A atualização de um caso de uso significa também a geração de novos exemplos. A remoção de um fluxo implica na remoção de pelo menos um exemplo à coleção, da mesma forma, a adição de um exemplo representa a adição de pelo menos um exemplo. A edição de um fluxo edita também os exemplos que influenciados por esse fluxo. Para ilustrar essa possibilidade, será alterado o caso de uso "Concluir Pedido" e será removido uso do CEP, o que implica na remoção do Elemento "CEP" e o fluxo que trata da sua validação. Como consequência dois exemplos serão removidos e, como a informação do CEP era obrigatória, todos os exemplos foram alterados. Como um dos resultados dessa alteração a Listagem 18 apresenta o mesmo exemplo na Listagem 17 após a alteração.

```
Dado que a base de dados contém:
  | id|      item|  valor| total_selecionado|total_estoque|
  |  3| Produto C|   5.89|                 2|           13|
  |  4| Produto D|   5.29|                 1|           17|
  |  1| Produto A|   4.99|                 2|          423|
  |  2| Produto B|   6.80|                 3|          125|

Exemplo: concluir pedidos com sucesso após inserir cupom de desconto e
falha ao validar cupom de desconto.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita não correto o Cupom de Desconto,
  usuário clica sobre o elemento Validar Cupom,
  sistema não valida 'Cupom de Desconto', identifica erro: 'Não foi
possível validar o cupom'
  sistema exibe mensagem de erro,
  usuário clica sobre o elemento Fechar Pedido,
então, sistema retorna a seleção de pedidos.
```

Listagem 18. Alteração do exemplo "concluir pedidos com sucesso após inserir cupom de desconto e falha ao validar cupom de desconto".

5.4
Limitação das regras de negócio

Na Listagem 19 é apresentado um exemplo em que o usuário não foi validado devido a um erro no preenchimento da identificação e a mensagem de erro exibida faz referência tanto ao fato de se preencher o usuário errado quanto à possibilidade do usuário ter sido bloqueado. Um elemento pode ter associada a ele uma série de regras de negócio. No entanto devido a forma como uma etapa de verificação foi definida no FunTester, não é possível a construção de etapas distintas de verificação para cada regra de negócio de um mesmo elemento. Talvez o ideal, para este caso, seria ter uma regra de negócio específica para identificar um usuário bloqueado e outra para apenas validar se o usuário está cadastrado na base.

```
Dado que a base de dados contém:
  |usuario|     senha| tentativas|
  |  maria|  maria123|          1|
  | marcia| marcia123|          3|
  | daniel| daniel123|          3|
  |   joao|   joao123|          1|

Exemplo: efetuar login com sucesso após falha na verificação da senha e
falha na verificação do CAPTCHA.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
   sistema exibe página de login com todos elementos em branco,
   sistema gera CAPTCHA,
   usuário digita 'maria' no elemento Usuário,
   usuário digita 'maria12' no elemento Senha,
   usuário digita correto o CAPTCHA,
   usuário clica sobre o elemento Login,
   sistema valida 'CAPTCHA',
   sistema valida 'Usuário',
   sistema não valida 'Senha', identifica erro: 'A senha não confere'
   sistema exibe mensagem de erro,
   sistema incrementa o número de tentativas com falhas,
   sistema exibe página de login com todos elementos em branco,
   sistema gera CAPTCHA,
   usuário digita 'maria' no elemento Usuário,
   usuário digita 'maria123' no elemento Senha,
   usuário digita não correto o CAPTCHA,
   usuário clica sobre o elemento Login,
   não valida 'CAPTCHA', identifica erro: 'O CAPTCHA não foi corretamente
preenchido.'
   sistema gera novo CAPTCHA,
   usuário digita correto o CAPTCHA,
   usuário clica sobre o elemento Login,
```

```
   sistema valida 'CAPTCHA',
   sistema valida 'Usuário',
   sistema valida 'Senha',
   sistema encerra login,
então, retorna ao sistema autorização de uso.
```

Listagem 19. Exemplo: Efetuar login com sucesso após falha na verificação da senha e falha na verificação do CAPTCHA.

6
Estudo de caso

Runeson e Höst (2008) definem o estudo de caso como um método empírico destinado a investigar fenômenos contemporâneos em seu contexto, especialmente quando a relação entre o fenômeno e o contexto não é clara, o que, segundo eles, se aplica ao contexto da engenharia de software, já que os estudos de caso oferecem uma abordagem que não necessita de uma fronteira rígida entre o objeto estudado e seu ambiente. Nesse contexto, Runeson e Höst (2008) comparam a engenharia de software às ciências humanas onde o uso de casos de estudo é largamente utilizado. Já que a prova é difícil de se obter pela falta de uma teoria "pesada", enquanto que o aprendizado é certamente possível de se verificar (Flyvbjerg, 2016). Especificamente, Runeson e Höst (Runeson e Höst, 2008) propõem em uma estrutura para a organização de um estudo de caso em engenharia de software, em que devem ser definidos: o objetivo do caso de estudo; a definição do caso de estudo; o método de coleta de evidências, a análise dos dados e a geração de um relatório contendo os resultados identificados.

Como foi apresentado no Capítulo 1, dois dos objetivos específicos desse trabalho são:

- *Avaliar a possibilidade do uso de exemplos, no formato proposto, como um canal efetivo de comunicação e compartilhamento de conhecimento de domínio.*
- *Avaliar o uso de exemplos como uma ferramenta de validação de especificações.*

Nesse sentido, o nosso estudo de caso segue uma abordagem qualitativa com o objetivo de verificar se a nossa proposta para a geração e representação de exemplos auxiliam pessoas, com ou sem conhecimento técnico, no entendimento de requisitos funcionais e na tomada de decisões. De forma mais precisa, o estudo conduzido teve como objetivo responder as seguintes hipóteses:

i. O uso de exemplos no formato proposto **permite um entendimento claro[4] sobre o comportamento representado** pelo fluxo de eventos.

ii. O uso de exemplos práticos de uso **permite que pessoas com ou sem conhecimento técnico identifiquem falhas e/ou comportamentos indesejáveis** de uma funcionalidade ainda não implementada de um software.

No nosso estudo de caso, participantes foram convidados para avaliar a nossa abordagem da seguinte forma: foram apresentadas as funcionalidades de um sistema não desenvolvido e foi solicitado ao participante que lesse e verificasse algumas descrições de exemplos práticos de uso dessas funcionalidades. Essa verificação foi feita tendo como base as informações apresentadas durante o procedimento e na experiência prévia de cada um dos participantes no uso de softwares que implementam funcionalidades semelhantes. Durante a realização dessa atividade foram feitas algumas perguntas sobre as impressões do participante sobre os exemplos apresentados.

Nas subseções seguintes serão apresentados: a abordagem utilizada para a seleção dos exemplos de uso utilizados no caso de estudo, o procedimento para a coleta de evidências, os participantes do estudo, a apresentação dos dados e as análises feitas sobre os dados.

6.1
Seleção de exemplos

De acordo com Flyvbjerg (2016), o teste de hipóteses está diretamente relacionado com a questão do quão representativo pode ser um estudo de caso, o que por sua vez, está relacionado à seleção de casos e amostras. Para a verificação das hipóteses levantadas, a seleção de amostras foi feita com o objetivo de selecionar exemplos de uso que representem situações distintas, que garanta que um exemplo não represente um cenário contido em outro exemplo. Para a geração

[4] O entendimento claro, será aqui observado como um entendimento suficiente para a identificação de erros e sugestão de melhorias.

desses exemplos foram criados três casos de uso (utilizando o FunTester) descrevendo funcionalidades comumente encontradas em sistemas web, são elas:

- Efetuar Login: Caso de uso que descreve a funcionalidade de controle de acesso de um sistema (similar ao apresentado na Tabela 11, seção 5.1, que pode ser visto no Apêndice B.1).
- Seleção de ingresso: Caso de uso que descreve uma seleção de ingressos de um cinema (apresentado no Apêndice B.2).
- Cadastro de usuário: Caso de uso que descreve um cadastro de usuário em que é solicitado, nome, endereço, CEP, CPF e número do telefone celular (apresentado no Apêndice B.3).

Um objetivo, com esses casos de uso, é permitir que o participante desse estudo tenha a capacidade de assumir o papel de um cliente ou usuário do sistema que implementa essas funcionalidades, visto que elas representam comportamentos que provavelmente ele já vivenciou.

Para cada caso de uso foi gerada uma coleção de exemplos de uso e, dessa coleção foram selecionados três exemplos, sendo que um deles foi derivado do fluxo principal. No total foram utilizados nove exemplos (Apêndice C), em três deles foram inseridos uma falha para simular a omissão de uma etapa de verificação ou de ação do usuário. É esperado que o participante seja capaz de identificar esses erros, o que será importante para a verificação da hipótese (ii), e fornecerá dados que irão corroborar com a verificação da hipótese (i), já que para a verificação do erro é primeiro necessário ter um entendimento sobre o que está sendo apresentado.

Alguns desses erros foram planejados para parecerem nítidos, de fácil identificação, como no quinto exemplo apresentado intitulado “Seleção de ingresso com sucesso após falha na verificação de quantidade disponível” (exibido na Listagem 20), em que o usuário insere um valor nulo na quantidade e ainda assim há uma validação do campo.

Dado que a base de dados contém:

quantidade	filme	sessao
2	Zootopia	18:00
2	Deadpool	15:30
1	Zootopia	16:30
3	Deadpool	18:00

Exemplo: seleção de ingresso com sucesso após falha na verificação de quantidade disponível.
Contexto inicial: usuário na página seleção de ingressos e base de dados com usuários cadastrados.
Cenário:
usuário seleciona 'Deadpool' no elemento Filme,
usuário seleciona '15:30' no elemento Sessão,
usuário digita '3' no elemento Quantidade,
usuário clica no elemento Selecionar,
sistema não valida 'Quantidade', identifica erro: 'A quantidade solicitada é maior que a disponível'
sistema exibe mensagem de erro,
sistema limpa campo quantidade,
usuário digita '0' no elemento Quantidade,
usuário clica no elemento Selecionar,
sistema exibe confirmação de seleção,
então, retorna ao sistema seleção de ingresso.

Listagem 20. Seleção de ingresso com sucesso após falha na verificação de quantidade disponível.

No sétimo exemplo apresentado, do caso de uso "Cadastro de cliente", intitulado "cadastro de cliente com sucesso após falha na verificação do CPF" (exibido na Listagem 21), após a correção do erro é omitida a ação do usuário de clicar no elemento "Cadastrar" e ainda assim o sistema conclui com sucesso. Vale ressaltar que a etapa omitida neste exemplo está presente nos outros dois exemplos apresentados referentes ao mesmo caso de uso.

Exemplo: cadastro de cliente com sucesso após falha na verificação de cpf.
Contexto inicial: Usuário na página de cadastro.
Cenário:
sistema exibe página de cadastro com todos elementos em branco,
usuário digita 'Lorem ipsum dolor sit amet, elit. Ae' no elemento Nome,
usuário digita '@#!436.715.993' no elemento CPF,
usuário digita 'Lorem ipsum dolor sit amet, consectetuer' no elemento Endereço,
usuário digita '01849-375' no elemento CEP,
usuário digita '(03)70598-4258' no elemento Celular,
usuário clica sobre o elemento Cadastrar,
sistema valida 'Nome',
sistema não valida 'CPF', identifica erro: 'O CPF deve ter o formato "123.123.123-12"'
sistema exibe mensagem de erro,
usuário digita '687.000.447-21' no elemento CPF,
sistema valida 'CPF',
sistema valida 'Endereço',
sistema valida 'CEP',
sistema valida 'Celular',
sistema encerra cadastro,
então, envia cadastro para a base de dados.

Listagem 21. Cadastro de cliente com sucesso após falha na verificação do CPF.

No último exemplo apresentado, do caso de uso "Cadastro de cliente", intitulado "cadastro de cliente com sucesso após falha na verificação do CEP"

(exibido na Listagem 22), após a verificação do CEP o usuário não insere novamente o CEP, o sistema não valida novamente o CEP e permite o envio do cadastro para a base de dados.

Exemplo: cadastro de cliente com sucesso após falha na verificação do cep.
Contexto inicial: Usuário na página de cadastro.
Cenário:
sistema exibe página de cadastro com todos elementos em branco,
usuário digita 'Lorem ipsum dolor sit amet, elit. Ae' no elemento Nome,
usuário digita '562.776.269-97' no elemento CPF,
usuário digita 'Lorem ipsum dolor sit amet, consectetuer' no elemento Endereço,
usuário digita '@#!21306-' no elemento CEP,
usuário digita '(22)63349-7294' no elemento Celular,
usuário clica sobre o elemento Cadastrar,
sistema valida 'Nome',
sistema valida 'CPF',
sistema valida 'Endereço',
sistema não valida 'CEP', identifica erro: 'CEP deve ser no formato "12345-123"'
sistema exibe mensagem de erro,
usuário digita '(22)63349-7294' no elemento Celular,
sistema valida 'Celular',
sistema encerra cadastro,
então, envia cadastro para a base de dados.

Listagem 22. Cadastro de cliente com sucesso após falha na verificação do CEP.

6.2.
O procedimento

Primeiramente, para cada caso de uso, foi apresentada ao participante a funcionalidade, foi descrita a interface, foram detalhados os elementos presentes na interface (mencionados no caso de uso) e as regras de negócio associadas a cada elemento, em seguida foram apresentados os exemplos, explicado a estrutura do exemplo e então foi solicitado que o participante lesse cada exemplo e para cada um verificasse se:

- Há erro? Qual seria?
- Poderia ser melhor? Como?

A definição de erro, nesse caso, será aquilo que segundo o participante não está de acordo o que foi apresentado ou que, segundo ele, representa um erro de especificação, algo que para ele não faz sentido de acontecer.

6.3.
Aplicação e avaliação dos resultados

Os participantes convidados são classificados em dois grupos: aqueles que atuam profissionalmente como desenvolvedores de software, que têm experiência no desenvolvimento de sistemas web, e aqueles que possuem pouco ou muito pouco conhecimento técnico, mas que se consideram experientes usuários de sistemas web. Participaram desse estudo de caso 8 pessoas:

- 5 com pouco ou muito pouco conhecimento técnico;
- 3 profissionais desenvolvedores de softwares;

Todos os participantes confirmaram ter experiência, como usuário, com as funcionalidades apresentadas. O estudo de caso foi executado individualmente com cada um dos participantes, o ambiente da aplicação foi um lugar simples com mesa e cadeira e com pouca circulação de pessoas. A única tecnologia utilizada foi um gravador para gravar a discussão ao longo do processo. Cada participante concordou com um termo de consentimento (Apêndice D).

6.4.
Apresentação dos dados

Os dados coletados com o estudo foram divididos em 3 grupos de dados:

- Identificação de erros;
- Sugestão de melhorias;
- Participação individual.

Os grupos serão apresentados nas subseções seguintes.

6.4.1
Identificação de erros

Os erros foram inseridos nos exemplos 5, 7 e 9. Como pode ser visto na Figura 8. Dos 8 participantes, 7 identificaram o erro do exemplo 5, apenas 5 identificaram o erro do exemplo 9, somente 3 identificaram o erro do exemplo 7. Também foram identificados erros nos exemplos 6 e 8 que, neste caso, foram erros de especificação, como por exemplo, no exemplo 6 foram considerados erros de especificação permitir que nome pudesse ter vírgula e o fato de não haver uma validação mínima do endereço. Não foram identificados erros por nenhum dos participantes nos exemplos 1, 2, 3 e 4.

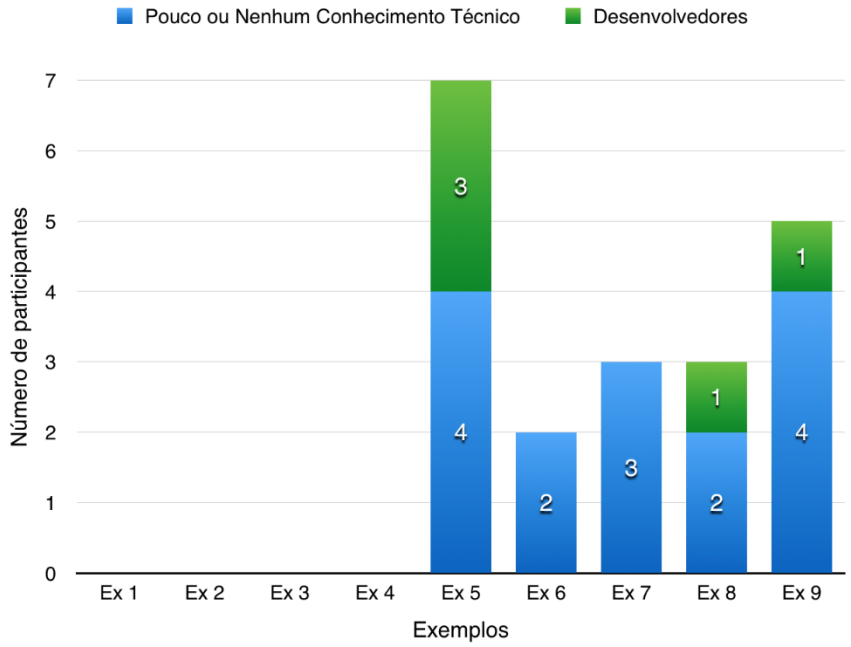


Figura 8. Identificação de Erros.

Todos os erros inseridos foram identificados por pelo menos 3 dos participantes. O erro inserido no exemplo 5, o que foi considerado como mais nítido, foi identificado por quase todos. Além dos erros inseridos, foram identificados erros considerados erros de especificação. Os erros foram observados tanto por indivíduos com e sem conhecimento técnico, sendo que aqueles com pouco ou nenhum conhecimento técnico foram melhores nessa tarefa. Esses dados dão suporte à hipótese (ii), mostram que o uso de exemplos práticos de uso, no formato proposto, permite a identificação de erros por pessoas com ou sem

conhecimento técnico. Esses dados também indicam que houve um entendimento mínimo e suficiente do que era proposto pelos exemplos para a verificação de erros.

6.4.2
Sugestão de melhorias

Para todos os exemplos, exceto o exemplo 4, foram sugeridas melhorias nas especificações. Muitas dessas sugestões representam alteração funcionais e outras não funcionais que representam alterações na forma como acontece a iteração do usuário com o sistema. Todas as sugestões feitas são propostas possíveis de serem implementadas, ou seja, não houve nenhuma sugestão que fugisse do escopo definido pela funcionalidade ou que representasse algo irreal. Como pode ser observado na Figura 9, o primeiro caso de uso foi o que mais teve sugestões de melhorias. Para cada caso de uso, as sugestões mais frequentes são apresentadas a seguir.

Caso de Uso Efetuar Login:

- Remover o botão validar CAPTCHA e fazer essa verificação após o clique sobre o botão login;
- Remover a verificação do CAPTCHA. Essa verificação é um exagero já que é previsto o bloqueio de usuário após o quarto erro seguido;
- Não limpar todos os campos após erro;
- Não permitir a troca de usuário após falha de validação, como apresentado no exemplo 4.
- A mensagem de erro deve informar se o erro de validação do usuário foi consequência da não identificação do usuário na base ou se o usuário foi bloqueado.

Caso de Uso: Seleção de Ingresso

- Informar a quantidade de ingressos disponíveis na interface do sistema;
- Informar a quantidade de ingressos disponíveis na mensagem de erro;

Caso de Uso: Cadastro de Cliente

- Não permitir nomes com símbolos como vírgula;
- Fazer uma validação do nome;
- Fazer uma validação do endereço;
- Fazer uma validação prévia dos campos antes do clicar em cadastrar;
- Preencher automaticamente o endereço a partir do CEP;
- Validar consultando uma base de dados o CEP e o CPF.

A Figura 9, mostra a quantidade de sugestões feitas por exemplo. É interessante observar que o exemplo que teve mais sugestões foi o exemplo com erro do caso de uso "Seleção de Ingresso", somente ao avaliar esse exemplo é que foi sugerido informar ao cliente a quantidade disponível de ingressos, nenhuma sugestão foi feita com relação ao primeiro exemplo do mesmo caso de uso.

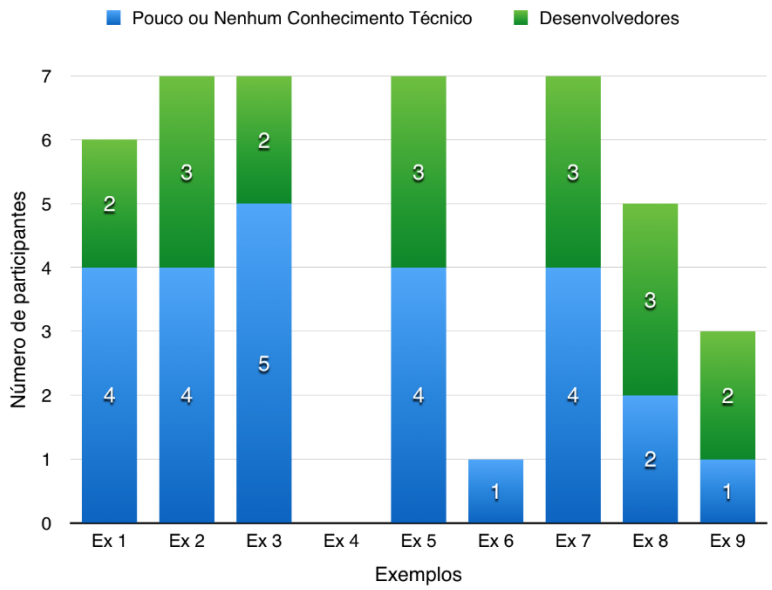


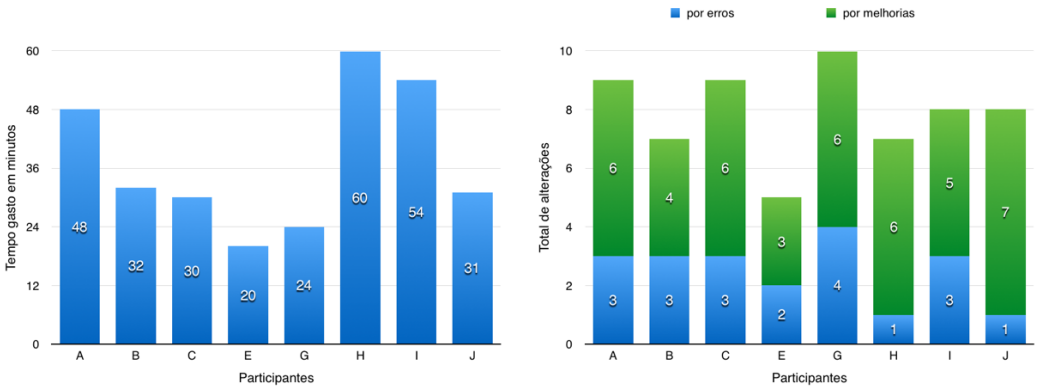
Figura 9. Sugestões de melhorias.

A grande quantidade delas se concentrou nos exemplos derivados do primeiro caso de uso, o "Efetuar Login", ao mesmo tempo, esses exemplos foram considerados sem erros, o que leva a crer que houve uma compreensão do que era proposto nos exemplos, o suficiente para que o participante pudesse sugerir melhorias. O resultado obtido com a identificação de falhas e com as sugestões de

melhorias mostram que o entendimento do exemplo permitiu ao participante identificar possíveis falhas e fazer a sugestão factíveis de melhorias. Com base nesse estudo de caso, não foi encontrada uma evidência que invalidasse a hipótese (i). Dessa forma é possível afirmar que o uso de exemplos não prejudicou o entendimento de um comportamento.

6.4.3
Participação individual

O tempo gasto por cada participante variou de 20 a 60 minutos para a verificação dos 9 exemplos. A Figura 10.a mostra o tempo gasto por cada participante e a Figura 10.b identifica a quantidade de erros e sugestões de melhorias identificados por cada um, sendo que os participantes H, I e J são os participantes com conhecimento técnico.



(a) Tempo gasto (b) Erros e melhorias

Figura 10. Comparativo

Os participantes com conhecimento técnico precisaram de mais tempo para analisar os exemplos (uma média de 48 minutos), proporcionalmente, o "desempenho" dos participantes com pouco conhecimento técnico foi próxima do dos participantes que possuem conhecimento técnico, o que reforça a ideia que o uso de exemplos permite que haja uma certa equiparação de conhecimento de domínio entre clientes e desenvolvedores, o que dá suporte ao o que é apresentado pelas hipóteses (i) e (ii).

6.5.
Discussão

Um participante comentou que a fluidez com que os eventos são apresentados o induziram, em um primeiro momento, a crer que estava tudo certo. Segundo ele, só depois de uma lida com mais atenção que ele foi capaz de identificar que estava faltando algo. Outros dois participantes reclamaram da quantidade de texto. Como já era esperado, a falha inserida mais evidente foi identificada por quase todos, enquanto que a mais discreta foi observada por poucos. Interessante observar que os falsos negativos, nesse caso, foram na verdade características consideradas erros de especificação.

Com relação às sugestões de melhorias, a grande quantidade delas se concentrou no caso de uso, o "Efetuar Login", em segundo ficou o caso de uso efetuar cadastro, e o menos criticado foi o da seleção de ingresso. Uma possibilidade para o fato da terceira ser a menos criticada está relacionado ao tamanho e a simplicidade do caso de uso, o que reforça a ideia de construção de casos de uso pequenos, com um escopo reduzido, para facilitar a sua validação. O fato do primeiro caso de uso ter sido o mais criticado e, ao mesmo tempo, ter sido considerado o único sem falhas leva a crer que houve uma compreensão do que era proposto no exemplo, o suficiente para que o participante pudesse sugerir melhorias. O caso de uso cadastrar cliente, mostrou que campos textuais sem uma regra clara de negócio representam potenciais barreiras no entendimento, principalmente quando há uma geração de texto. É interessante que haja uma validação dos campos textuais e que a valoração desse tipo de elemento não seja uma simples geração de texto que satisfaça uma expressão regular, ou que respeite apenas limites de tamanho; ao invés disso, seria melhor a geração de um texto coerente com o contexto do exemplo, uma opção seria utilizar uma base de dados local para o armazenamento de possíveis valores.

7
Conclusões

A qualidade de um software pode ser vista como uma concordância entre os requisitos de performance e funcionais declarados explicitamente, com os padrões de desenvolvimento devidamente documentados, e com as características implícitas que são esperadas de todo o software. A garantia da qualidade de um software, segundo (Silva et al, 2014), é vista como uma atividade planejada e sistemática para garantir a conformidade dos processos, práticas, padrões e procedimentos estabelecidos em todas as fases do desenvolvimento de um software, começando pela análise de requisitos, codificação, teste e implantação.

Nas metodologias ágeis, a garantia da qualidade é implícita no processo de desenvolvimento e está presente em práticas como reuniões e retrospectivas ao final de ciclos de desenvolvimento, onde o cliente é convidado para avaliar e verificar se o que é entregue tem a qualidade que ele deseja. Essa prática é eficaz, mas não é suficiente para identificação antecipada de falhas e defeitos. Ao contrário dos métodos ágeis, os métodos tradicionais são famosos pelo tempo gasto no planejamento e na especificação do software, onde é comum o uso de métodos formais leves para especificação e documentação de funcionalidades. Tal abordagem, além de guiar de forma clara o desenvolvedor, permite também um desenvolvimento correto por construção, garantindo assim a qualidade do software desde o início do seu desenvolvimento.

O BDD visa a participação do cliente como algo essencial para a construção e avaliação de especificação, já que só o cliente sabe e entende quais são as características de qualidade implícitas esperadas do software. O BDD aproxima o cliente no momento do levantamento de requisitos de uma funcionalidade, aproximando a linguagem utilizada para a descrição de especificações à linguagem do cliente. Como mostramos, nessa abordagem há uma elevada tendência de que

essas especificações não sejam adequadas. Uma das justificativas é a ausência de uma metodologia clara para a construção e descrição de comportamentos.

Nesta dissertação de mestrado foi apresentada uma abordagem aplicada ao BDD que visa trazer um pouco do formalismo às especificações e traduzi-las em exemplos práticos de uso que irão compor a descrição de um comportamento. Essa tradução é feita de forma automática e tem como objetivo explorar a especificação e evidenciar para o cliente o comportamento que de fato a especificação representa. Para tanto, foi necessário trazer ao método ágil um pouco mais de planejamento e um esforço maior na especificação do que se pretende desenvolver, o que no final das contas pode representar um retardo no início do desenvolvimento. No entanto, o preço a ser pago pode compensar. O uso de exemplos reais de uso permite uma validação das especificações e reduz as chances de erros ou falhas durante o desenvolvimento, contribuindo para a redução do retrabalho inútil. Além disso, a formalização presente nos casos de uso permite um desenvolvimento soluções mais próximas do correto por construção.

1. Apresentar um processo de geração e seleção de cenários abstratos de uso a partir da exploração de um modelo de comportamento gerado a partir da descrição textual de casos de uso.
2. Gerar uma coleção de exemplos para cada caso de uso.
3. Converter um cenário abstrato de uso em um cenário concreto de uso.
4. Fazer a paráfrase de um cenário concreto de uso em um exemplo de uso escrito usando texto semiestruturado.
5. Avaliar a possibilidade do uso de exemplos como um canal efetivo de comunicação e compartilhamento de conhecimento de domínio.
6. Avaliar o uso de exemplos como uma ferramenta de validação de comportamento.

7.1
Resultados alcançados

O presente trabalho apresentou um processo para a construção e seleção de cenários que tem como objetivo reduzir o número de exemplos gerados sem que haja uma perda significativa na representação do comportamento. Buscou-se um meio termo que permitisse uma boa representação do comportamento, explorando cada fluxo presente no caso de uso, e que ao mesmo tempo permitisse uma leitura que não fosse cansativa para o cliente.

No processo apresentado para a conversão de cenários abstratos em concretas houve uma preocupação em gerar valores que fossem também próximos do real, do cotidiano, ao contrário do que se vê em abordagens voltadas para testes. Nesse processo, uma valoração intencionalmente errada não é um valor randômico (ou coisa do tipo), mas é um valor "quase certo", pois, tenta imitar um erro humano, um caractere esquecido ou um valor um pouco acima do permitido.

A solução proposta é independente de tecnologias externas de armazenamento, e para que seja possível a geração de exemplos, a ferramenta proposta necessita apenas dos casos de uso construídos com o FunTester e da descrição de uma base de dados para cada caso de uso, se o caso de uso faz uso de dados vindos de uma base externa.

Na paráfrase de um cenário é apresentado um formato para a descrição de exemplos que tenta ilustrar exemplos de uma forma menos formal e mais próxima do natural. Os exemplos são apresentados como um cenário com um roteiro bem definido.

Como foi mostrado no Capítulo 5, a abordagem proposta facilita a manutenção, já que alterações ou atualizações na especificação serão refletidas de imediato nos exemplos, o que permite uma identificação precisa da parte do comportamento que precisa ser revista e reavaliada pelo cliente e permite que desenvolvedores localize onde deve ser feita uma alteração. Os exemplos alterados durante uma manutenção também podem ser uma forma de comunicar ao usuário a atualização feita. Ao invés das tradicionais mensagens de atualização, indecifráveis na maioria das vezes, os exemplos podem comunicar de forma eficiente ao usuário

a alteração no comportamento do sistema, na forma: “O que antes era assim, agora é desse jeito”.

Como apresentado no Capítulo 6, os exemplos auxiliam clientes e desenvolvedores no entendimento sobre o comportamento da funcionalidade a ser implementada. Tanto pessoas com ou sem conhecimento técnico puderam entender o exemplo, identificar falhas de implementação e de especificação e ainda propor melhorias coerentes com a funcionalidade apresentada. O custo dessa verificação se limitou ao tempo gasto na formalização da especificação e no tempo utilizado para a validação, tempo que certamente é menor que o tempo que seria gasto para a verificação após implementação.

A abordagem aqui apresentada para a geração de exemplos, aliada ao que é proposto pelo FunTester possibilita uma verificação de baixo custo de especificações, permite um uso de melhor qualidade do BDD, permite a geração e verificação automática de testes. Ainda que com limitações, essas duas ferramentas podem de fato contribuir no desenvolvimento de sistemas web.

7.2
Trabalhos futuros

É necessária uma melhor definição das regras de negócio que permita que as regras tenham um comportamento mais reativo, que não somente sirvam como condicionais para a seleção de fluxos, e que cada regra possa estar relacionada a uma etapa de verificação. Como apresentado no capítulo 5, as regras de negócio de um mesmo elemento não podem ser verificadas em etapas diferentes. Uma etapa de verificação necessariamente verifica todas as regras associadas ao elemento, o que inviabiliza um tratamento específico para cada regra. Essa alteração ampliaria as possibilidades de uso da solução aqui proposta, e também do FunTester.

Seria interessante a possibilidade da definição de variáveis simples de sistemas, como contadores e acumuladores. A ferramenta desenvolvida tem sua atuação limitada aos elementos de interface com os quais o usuário pode editar o estado. No caso, as variáveis simples de sistemas possibilitariam um controle maior sobre o fluxo e uma especificação melhor de cenários.

É importante que novos casos de estudos sejam feitos em ambientes reais de desenvolvimento de software e que avaliações mais rigorosas sejam feitas sobre a abordagem, no sentido de verificar o esforço de clientes na verificação das especificações e o esforço dos desenvolvedores na construção de especificações.

Seria interessante tornar a solução proposta em uma aplicação web, ao invés de uma aplicação Java. Ela seria uma aplicação web acessível de todo e qualquer browser com acesso à internet e com interface adaptável aos diferentes tipos de tela. Para tanto, seria necessário que pelo menos parte do FunTester também fosse convertida em aplicação web. Isso garantiria uma total portabilidade do sistema.

A proposta aqui apresentada se limita aos sistemas web, no entanto ela pode ser adaptada a outros contextos, como ao das aplicações móveis, tão comum nos dias de hoje, ou também adaptadas ao contexto das APIs (Interface de programação de aplicações, *Application Programming Interface*). As aplicações móveis estão cada vez mais próximas das aplicações web e, de fato, uma grande parcela destas aplicações são aplicações web embarcadas em aplicações móveis. As APIs são utilizadas como interface entre a página web exibida no *browser* e a aplicação em execução no servidor. Em uma API são definidas URLs e os parâmetros que podem ser enviados por essas URLs. Cada URL faz referência à uma funcionalidade e os parâmetros são as variáveis necessária para que a funcionalidade seja possível. Como resposta uma requisição retorna o conteúdo solicitado, com base nos parâmetros enviados. Por exemplo, uma API pode definir uma URL para a validação de um usuário que recebe como parâmetro o login e senha e retorna uma variável de sessão que concede ou não a autorização de uso.

8
Referências bibliográficas

ADZIC, G. **Bridging the Communication Gap: Specification by Example and Agile Acceptance Testing**. Neuri Limited. Kindle Edition, 2009.

BECK, K. at al. **Manifesto for Agile Software Development**. 2001. Disponível em: <http://agilemanifesto.org>

BERGMANN, U., **Evolução de Cenários Através de um Mecanismo de Rastreamento Baseado em Transformações**. Dissertação de Mestrado, Pontifícia Universidade Católica do Rio de Janeiro, 2003.

CARTER, J; GARDNER, W. B. **BHive: Towards Behaviour-Driven Development Supported by B-Method**. *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, Pittsburgh, PA, 2016, pp. 249-256.

CUCUMBER, **Getting started with Cucumber.** Disponível em: <https://cucumber.io/docs>. Acessado em 10/08/2016.

DIEPENBECK, M. et al. **Towards automatic scenario generation from coverage information.** 2013 8th International Workshop on Automation of Software Test (AST), San Francisco, CA, 2013, pp. 82-88.

DYBÅ, T.; DINGØYR, T. **Empirical studies of agile software development: A systematic review**. Inf. Softw. Technol. 50, pp 9-10, 2008.

FLYVBJERG, B,. **Five Misunderstandings About Case-Study Research.** v. 12, n. 2. Sage Publications. Aalborg University, Denmark, 2006.

FUNTESTER; Manual. Disponível em: < http://funtester.org/manual/>. Acessado em 10/02/2017.

GHERKIN. **Reference**. Disponível em: <https://cucumber.io/docs/ reference>. Acessado em 10/10/2016.

GONZÁLEZ-DE-ALEDO, P. et al. **Towards a Verification Flow Across Abstraction Levels Verifying Implementations Against Their Formal Specification.** IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 3, pp. 475-488, 2017.

GUTIÉRREZ J.J. et al. **Visualization of Use Cases through Automatically Generated Activity Diagrams**. Model Driven Engineering Languages and Systems. MODELS 2008. Lecture Notes in Computer Science, vol 5301. Springer, Berlin, Heidelberg, 2008.

HEUMANN, J. **Generating Tests from Use Cases**. *The Rational Edge-zine*. 2001. Disponível em <http://www.ibm.com/developerworks/rational /library/content/RationalEdge/jun01/GeneratingTestCasesFromUseCasesJune01 .pdf>. Acessado em 10/09/2016.

IBARRA, U. ÁLVAREZ, F.; & VARGAS, M. **Use processes, modeling requirements based on elements of BPMN and UML Use Case Diagrams**, *Software Technology and Engineering (ICSTE), 2010 2nd International Conference on*, San Juan, PR, 2010, pp.

ISO. **ISO/IEC 14882:2011 Information technology --- Programming languages --- C++**. Geneva, Switzerland: International Organization for Standardization (2012).

KASSEL, N.W., 2006. **An approach to automate test case generation from structured use cases**. Clemson University, 2006.

LI, A; OFFUTT, J. **A test automation language framework for behavioral models**, Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on, Graz, pp. 1-10, 2015.

LI, A; et al. **Skyfire: Model-Based Testing with Cucumber.** 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), Chicago, IL, 2016, pp. 393-400.

MASON, P.; SUPSRISUPACHAI, S. **Paraphrasing use case descriptions and Sequence Diagrams: An approach with tool support,** Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on, Pattaya, Chonburi, 2009

NORTH, D. **Introducing BDD**. 2006. Disponível em: <https://dannorth.net/introducing-bdd/> Acessado em 10/02/2016.

PINTO, D. T. **Uma ferramenta para geração e execução automática de testes funcionais baseados na descrição textual de ca- sos de uso.** Orientador: Arndt Von Staa. Tese (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2013.

RUNESON, P.; HÖST, M. **Guidelines for Conducting and Reporting Case Study Research in Software Engineering**. Empirical Software Engineering, v. 14, n. 2, p. 131–164, dez. 2008.

SANTOS, J. P, et al; **Increasing Quality in Scenario Modelling with Model-Driven Development**, Seventh International Conference on the Quality of Information and Communications Technology, Porto, 2010, pp. 204-209.

SAWANT, K. P, et al. **Deriving requirements model from textual use cases**. In *Companion Proceedings of the 36th International Conference on Software Engineering* (ICSE Companion 2014). ACM, New York, USA, 2014.C.

SILVA, F SELLERI. et al. **A Reference Model for Agile Quality Assurance: Combining Agile Methodologies and Maturity Models**. Quality of Information and Communications Technology (QUATIC), 2014 9th International Conference on the , vol., no., pp.139,144, 23-26 Sept. 2014

SOLIS, C.; WANG, X. **A Study of the Characteristics of Behaviour Driven Development**. *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*, Oulu, 2011, pp. 383-387

SQLITE; **Documentation**. Disponível em: <https://sqlite.org/docs.html>. Acessado em: 02/03/2017.

STAA, A.V.**.** ***Especificações***. Rio de Janeiro: Departamento de Informática, PUC-Rio. Disponível em: <http://www.inf.puc-rio.br/~inf1413>. Acessado em 02/03/1017.

WANDERLEY, F; SILVA, , A.; ARAÚJO, J. **Evaluation of BehaviorMap: A user-centered behavior language**. *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, Athens, 2015, pp. 309-320.

WYNNE, M. & HELLESOY, A. **The cucumber book:** behaviour-driven development for testers and developers. Pragmatic Bookshelf, 2012.J.

A
Exemplos gerados

A.1
Caso de uso “Efetuar Login”

A listagem a seguir apresenta os exemplos gerados para o caso de uso “Efetuar Login”.

```
Caso de Uso: Efetuar Login

Dado que a base de dados contém:
  |usuario|     senha| tentativas|
  |  maria|  maria123|          1|
  | marcia| marcia123|          3|
  | daniel| daniel123|          1|
  |   joao|   joao123|          1|

Exemplo: efetuar login com sucesso.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe itens do carrinho. ,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'daniel' no elemento Usuário,
  usuário digita 'daniel123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna autorização de uso.


Exemplo: efetuar login com sucesso após falha na verificação do CAPTCHA.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe itens do carrinho. ,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'daniel' no elemento Usuário,
```

```
  usuário digita 'daniel123' no elemento Senha,
  usuário digita não correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema não valida 'CAPTCHA', identifica erro: 'O CAPTCHA não foi
corretamente preenchido.'
  sistema gera novo CAPTCHA,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna autorização de uso.


Exemplo: efetuar login com sucesso após falha na verificação do CAPTCHA
e falha na verificação do usuário..
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe itens do carrinho. ,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'marci' no elemento Usuário,
  usuário digita 'marcia123' no elemento Senha,
  usuário digita não correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema não valida 'CAPTCHA', identifica erro: 'O CAPTCHA não foi
corretamente preenchido.'
  sistema gera novo CAPTCHA,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema não valida 'Usuário', identifica erro: 'Usuário não cadastrado
no sistema ou bloqueado.'
  sistema exibe mensagem de erro,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'marcia' no elemento Usuário,
  usuário digita 'marcia123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna autorização de uso.
```

```
Exemplo: efetuar login com sucesso após falha na verificação do usuário..
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe itens do carrinho. ,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'danie' no elemento Usuário,
  usuário digita 'daniel123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema não valida 'Usuário', identifica erro: 'Usuário não cadastrado
no sistema ou bloqueado.'
  sistema exibe mensagem de erro,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'daniel' no elemento Usuário,
  usuário digita 'daniel123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna autorização de uso.


Exemplo: efetuar login com sucesso após falha na verificação do usuário.
e falha na verificação do CAPTCHA.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe itens do carrinho. ,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'mari' no elemento Usuário,
  usuário digita 'maria123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema não valida 'Usuário', identifica erro: 'Usuário não cadastrado
no sistema ou bloqueado.'
  sistema exibe mensagem de erro,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'maria' no elemento Usuário,
  usuário digita 'maria123' no elemento Senha,
  usuário digita não correto o CAPTCHA,
```

```
  usuário clica sobre o elemento Login,
  sistema não valida 'CAPTCHA', identifica erro: 'O CAPTCHA não foi
corretamente preenchido.'
  sistema gera novo CAPTCHA,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna autorização de uso.


Exemplo: efetuar login com sucesso após falha na verificação da senha.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe itens do carrinho. ,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'marcia' no elemento Usuário,
  usuário digita 'marcia12' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema não valida 'Senha', identifica erro: 'A senha não confere'
  sistema exibe mensagem de erro,
  sistema incrementa o número de tentativas com falhas,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'marcia' no elemento Usuário,
  usuário digita 'marcia123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna autorização de uso.


Exemplo: efetuar login com sucesso após falha na verificação da senha e
falha na verificação do CAPTCHA.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe itens do carrinho. ,
  sistema exibe página de login com todos elementos em branco,
```

```
  sistema gera CAPTCHA,
  usuário digita 'daniel' no elemento Usuário,
  usuário digita 'daniel12' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema não valida 'Senha', identifica erro: 'A senha não confere'
  sistema exibe mensagem de erro,
  sistema incrementa o número de tentativas com falhas,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'daniel' no elemento Usuário,
  usuário digita 'daniel123' no elemento Senha,
  usuário digita não correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema não valida 'CAPTCHA', identifica erro: 'O CAPTCHA não foi
corretamente preenchido.'
  sistema gera novo CAPTCHA,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Login,
  sistema valida 'CAPTCHA',
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna autorização de uso.


Exemplo: efetuar login sem sucesso após cancelar operação.
Contexto inicial: usuário na página contendo formulário de login e base
de dados com usuários cadastrados.
Cenário:
  sistema exibe itens do carrinho. ,
  sistema exibe página de login com todos elementos em branco,
  sistema gera CAPTCHA,
  usuário digita 'joao' no elemento Usuário,
  usuário clica sobre o elemento Cancelar,
  sistema redireciona página para a página inicial,
então, não é fornecida autorização de uso.
```

A.2
Caso de uso “Concluir Pedidos”

A listagem a seguir apresenta os exemplos gerados para o caso de uso “Concluir Pedidos”.

```
Caso de Uso: Concluir Pedidos

Dado que a base de dados contém:
  | id|      item|  valor| total_selecionado|total_estoque|
  |  3| Produto C|   5.89|                 2|           13|
  |  4| Produto D|   5.29|                 1|           17|
  |  1| Produto A|   4.99|                 2|          423|
  |  2| Produto B|   6.80|                 3|          125|

Exemplo: concluir pedidos com sucesso.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita '22090-818' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após falha na verificação do cep.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita '@#!22012-4' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema não valida 'CEP para entrega', identifica erro: 'O CEP Deve
conter 7 números, exemplo: 22012-010'
  sistema exibe mensagem de erro,
  usuário digita '22229-009' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
```

```
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após atualizar quantidade de um
item.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário seleciona 'Produto D' no elemento Item,
  usuário digita '19' no elemento Quantidade,
  sistema não valida 'Quantidade', identifica erro: 'A quantidade
solicitada é maior que a disponível em estoque'
  sistema atualiza a quantidade,
  sistema atualiza valor total da compra,
  usuário digita '22525-568' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após atualizar quantidade de um
item.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário seleciona 'Produto B' no elemento Item,
  usuário digita '127' no elemento Quantidade,
  sistema não valida 'Quantidade', identifica erro: 'A quantidade
solicitada é maior que a disponível em estoque'
  sistema atualiza a quantidade,
  sistema atualiza valor total da compra,
  usuário digita '22858-531' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após atualizar quantidade de um
item e falha na verificação do cep.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
```

```
  usuário seleciona 'Produto A' no elemento Item,
  usuário digita '425' no elemento Quantidade,
  sistema não valida 'Quantidade', identifica erro: 'A quantidade
solicitada é maior que a disponível em estoque'
  sistema atualiza a quantidade,
  sistema atualiza valor total da compra,
  usuário digita '@#!22857-7' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema não valida 'CEP para entrega', identifica erro: 'O CEP Deve
conter 7 números, exemplo: 22012-010'
  sistema exibe mensagem de erro,
  usuário digita '22984-971' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após atualizar quantidade de um
item e falha na verificação da quantidade disponível de um item.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário seleciona 'Produto B' no elemento Item,
  usuário digita '127' no elemento Quantidade,
  sistema não valida 'Quantidade', identifica erro: 'A quantidade
solicitada é maior que a disponível em estoque'
  sistema exibe mensagem de erro e informa quantidade disponível,
  usuário digita '127' no elemento Quantidade,
  sistema não valida 'Quantidade', identifica erro: 'A quantidade
solicitada é maior que a disponível em estoque'
  sistema atualiza a quantidade,
  sistema atualiza valor total da compra,
  usuário digita '22547-562' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após atualizar quantidade de um
item, falha na verificação da quantidade disponível de um item e falha
na verificação do cep.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário seleciona 'Produto C' no elemento Item,
  usuário digita '15' no elemento Quantidade,
```

```
  sistema não valida 'Quantidade', identifica erro: 'A quantidade
solicitada é maior que a disponível em estoque'
  sistema exibe mensagem de erro e informa quantidade disponível,
  usuário digita '15' no elemento Quantidade,
  sistema não valida 'Quantidade', identifica erro: 'A quantidade
solicitada é maior que a disponível em estoque'
  sistema atualiza a quantidade,
  sistema atualiza valor total da compra,
  usuário digita '@#!22381-5' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema não valida 'CEP para entrega', identifica erro: 'O CEP Deve
conter 7 números, exemplo: 22012-010'
  sistema exibe mensagem de erro,
  usuário digita '22180-909' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após insere cupomcupom de
desconto.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita não correto o Cupom de Desconto,
  usuário clica sobre o elemento Validar Cupom,
  sistema não valida 'Cupom de Desconto', identifica erro: 'Não foi
possível validar o cupom'
  sistema atualiza valor total da compra,
  sistema exibe novo valor total,
  usuário digita '22721-678' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após insere cupom de desconto e
falha na verificação do cep.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita não correto o Cupom de Desconto,
  usuário clica sobre o elemento Validar Cupom,
  sistema não valida 'Cupom de Desconto', identifica erro: 'Não foi
possível validar o cupom'
  sistema atualiza valor total da compra,
```

```
  sistema exibe novo valor total,
  usuário digita '@#!22453-3' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema não valida 'CEP para entrega', identifica erro: 'O CEP Deve
conter 7 números, exemplo: 22012-010'
  sistema exibe mensagem de erro,
  usuário digita '22813-488' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após insere cupom de desconto e
falha ao validar cupom de desconto.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita não correto o Cupom de Desconto,
  usuário clica sobre o elemento Validar Cupom,
  sistema não valida 'Cupom de Desconto', identifica erro: 'Não foi
possível validar o cupom'
  sistema exibe mensagem de erro,
  usuário digita '22455-556' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após insere cupom de desconto,
falha ao validar cupom de desconto e falha na verificação do cep.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita não correto o Cupom de Desconto,
  usuário clica sobre o elemento Validar Cupom,
  sistema não valida 'Cupom de Desconto', identifica erro: 'Não foi
possível validar o cupom'
  sistema sistema exibe mensagem de erro,
  usuário digita '@#!22960-9' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema não valida 'CEP para entrega', identifica erro: 'O CEP Deve
conter 7 números, exemplo: 22012-010'
  sistema exibe mensagem de erro,
  usuário digita '22141-828' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
```

```
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos sem sucesso após cancelar conclusão para
comprar mais.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário digita '@#!22668-7' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema não valida 'CEP para entrega', identifica erro: 'O CEP Deve
conter 7 números, exemplo: 22012-010'
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Escolher mais produtos,
então, sistema retorna para a página inicial do sistema.


Exemplo: concluir pedidos com sucesso após remover item.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário seleciona 'Produto A' no elemento Item,
  usuário clica sobre o elemento Remover item,
  sistema atualiza carrinho de compras,
  sistema atualiza valor total,
  sistema exibe itens no carrinho,
  usuário digita '22185-113' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.


Exemplo: concluir pedidos com sucesso após remover item e falha na
verificação do cep.
Contexto inicial: usuário na página do carrinho de compras e carrinho
com pelo menos um item.
Cenário:
  sistema exibe itens no carrinho,
  usuário seleciona 'Produto D' no elemento Item,
  usuário clica sobre o elemento Remover item,
  sistema atualiza carrinho de compras,
  sistema atualiza valor total,
  sistema exibe itens no carrinho,
  usuário digita '@#!22318-3' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
```

```
  sistema não valida 'CEP para entrega', identifica erro: 'O CEP Deve
conter 7 números, exemplo: 22012-010'
  sistema exibe mensagem de erro,
  usuário digita '22264-168' no elemento CEP para entrega,
  usuário clica sobre o elemento Validar CEP,
  sistema valida 'CEP para entrega',
  sistema informa valor do frete,
  sistema adiciona valor do frete ao valor total da compra,
  usuário clica sobre o elemento Fechar Pedido,
  sistema prossegue para o pagamento,
então, componente retorna a seleção de pedidos.
```

B
Casos de uso

B.1
Efetuar login

Caso de Uso:	*Efetuar o login*
Objetivo:	*Obter a autorização de uso segundo os direitos de uso com os quais foi registrado.*
Ator:	*Usuário*

Pré-condição:	*1. Usuário na página contendo formulário de login;*
	2. Base de dados com usuários cadastrados:
	`| usuario | senha | tentativas |`
	`| Maria | maria123 | 1 |`
	`| João | joao123 | 1 |`
	`| Daniel | daniel123 | 0 |`
	`| Marcia | marcia123 | 3 |`

Pós-condição:	*Componente retorna autorização de uso.*
Acionamento:	*Quando o sistema solicitar pela primeira vez o fornecimento de uma autorização de uso*
Fluxo Principal:	*1. Sistema exibe página de login com todos elementos em branco* *2. Sistema gera CAPTCHA* *3. Usuário digita sua identificação;* *4. Usuário digita sua senha;* *5. Usuário digita o* CAPTCHA*;* *6. Usuário clica "Validar";* *7. Sistema verifica retorno da validação do CAPTCHA;* *8. Usuário clica "Login";* *9. Sistema verifica se a identificação foi corretamente preenchida e se está cadastrada no sistema;* *10. Sistema verifica <usuário, senha>;* *11. Sistema encerra login;* *12. Componente retorna "autorização de uso".*
	Fluxos Alternativos
FA 1	Evento E1/6: Falha na verificação do CAPTCHA. I. O controle de acesso informa ao usuário que o CAPTCHA não foi preenchido corretamente. II. Repete a partir de 5. Fim evento E1.

FA 2	Evento E2/7: Falha na verificação do Usuário i. Sistema informa ao usuário que o nome digitado não consta na base de dados ou está bloqueado ii. Repete a partir de 1. Fim evento E2.
FA3	Evento E3/8: Falha na verificação da senha. i. Sistema exibe mensagem de erro. ii. Sistema incrementa o número de tentativas com erro para o usuário. iii. Sistema verifica o número de tentativas iv. Repete a partir de 1. Fim evento E3.
FA4	Evento E4/E3/III: Bloquei de Usuário. i. Sistema informa usuário bloqueado; ii. Sistema redireciona página para a página inicial; iii. Sistema retorna ao sistema e não é fornecida autorização de uso. Fim evento E4.
FA5	Evento E5: Cancela operação i. Sistema redireciona página para a página inicial; ii. Sistema retorna ao sistema e não é fornecida autorização de uso. Fim evento E4.
	Regras de negócio
Usuário	Elemento do tipo texto; O nome do usuário deve ser fornecido; O campo não deve conter caracteres diacríticos, espaços em branco, tabulações, nem os caracteres '%', '\', '/', '?', '*', '@'; O nome do usuário deve conter pelo menos 6 caracteres; O usuário deve estar cadastrado no sistema.
Senha	Elemento do tipo texto; A senha deve ser fornecida; O nome do usuário deve conter pelo menos 8 caracteres; O par <usuário, senha> deve está correto.
CAPTCHA	Componente que recebe um texto e retorna um booleano. O componente deve permitir ou negar caso o texto digitado não seja igual ao fornecido pela imagem.
Validar	Elemento do tipo botão que dispara a verificação do CAPTCHA
Login	Elemento do tipo botão que dispara a verificação do Usuário e da Senha

B.2
Seleção de ingresso

Caso de Uso:	***Seleção de Ingresso***
Objetivo:	*Selecionar um ou mais ingressos de um mesmo filme*
Ator:	*Usuário*

Pré-condição:	*1. Usuário na página seleção de ingressos;*
	2. Base de dados com usuários cadastrados:
	`| filme | sessao | quantidade |`
	`| Deadpool | 1 | 2 |`
	`| Deadpool | 2 | 3 |`
	`| Zootipia | 1 | 12 |`
	`| Zootipia | 2 | 1 |`
	`| Zootipia | 3 | 2 |`

Pós-condição:	*Componente retorna autorização de compra dos ingressos selecionados.*
Fluxo Principal:	*1. Usuário seleciona filme;* *2. Usuário seleciona sessão;* *3. Usuário digita quantidade de ingressos;* *4. Usuário clica em comprar* *5. Sistema verifica quantidade disponível de ingressos;* *6. Sistema exibe confirmação de seleção;* *7. Componente retorna seleção de ingresso;*
	Fluxos Alternativos
FA 1	Evento E1/6: Falha na seleção de ingressos. I. *Sistema informa erro* II. *Sistema exibe mensagem: "Quantidade selecionada é superior a disponível".* III. Repete a partir de 1. Fim evento E1
FA 2	Evento E2/6: Cancelamento. I. *Usuário clica em cancelar;* II. *Sistema encerra página de seleção* III. *Sistema não retorna seleção de ingresso* Fim evento E2
	Regras de negócio
Filme	Elemento do tipo texto. Deve está contido na lista de filmes disponíveis.
Sessão	Elemento do tipo numérico. Deve está contido na lista de sessões disponíveis para um filme selecionado.
Quantidade	Elemento do tipo numérico. Deve está contido na lista de sessões disponíveis para um filme e sessão selecionados.
Comprar	Elemento do tipo botão.

B.3
Cadastro de usuário

Caso de Uso:	***Cadastro de Usuário***
Objetivo:	*Cadastrar novo cliente à base de dados.*
Ator:	*Usuário*
Pré-condição:	*1. Usuário na página seleção de ingressos;*
Pós-condição:	*Envio de cadastro para a base de dados*
Fluxo Principal:	*1. Sistema exibe página de cadastro com todos elementos em branco* *2. Usuário digita Nome;* *3. Usuário digita CPF;* *4. Usuário digita Endereço;* *5. Usuário digita CEP;* *6. Usuário digita número do celular;* *7. Usuário clica em cadastrar;* *8. Sistema verifica elemento Nome;* *9. Sistema verifica elemento CPF;* *10. Sistema verifica elemento Endereço;* *11. Sistema verifica elemento CEP;* *12. Sistema verifica elemento número do celular;* *13. Sistema encerra cadastro;* *14. Componente envia cadastro para base de dados.*
	Fluxos Alternativos
FA 1	Evento E1/8: Falha ao verificar Nome I. *Sistema exibe mensagem de erro;* II. *Usuário digita Nome* III. Volta para 8. Fim evento E1
FA 2	Evento E2/9: Falha ao verificar CPF I. *Sistema exibe mensagem de erro;* II. *Usuário digita CPF;* III. Volta para 9. Fim evento E2
FA 3	Evento E3/10: Falha ao verificar Endereço I. *Sistema exibe mensagem de erro;* II. *Usuário digita Endereço* III. Volta para 10. Fim evento E2
FA 4	Evento E4/11: Falha ao verificar CEP I. *Sistema exibe mensagem de erro;* II. *Usuário digita CEP* III. Volta para 11. Fim evento E4
FA 5	Evento E5/12: Falha ao verificar Número do celular I. *Sistema exibe mensagem de erro;* II. *Usuário digita número de celular* III. Volta para 12.

	Fim evento E5
FA 6	Evento E5/12: Cancelamento I. *Usuário clica em cancelar;* II. *Sistema encerra cadastro* III. *Sistema não envia cadastro para a base de dados.* Fim evento E5

C
Exemplos selecionados para o estudo de caso.

Caso de Uso: Efetuar Login

Dado que a base de dados contém:

tentativas	user	pass
3	daniel	daniel123
3	marcia	marcia123
1	joao	joao123
1	maria	maria123

Exemplo 01: efetuar login com sucesso.
Contexto inicial: usuário na página contendo formulário de login e base de dados com usuários cadastrados.
sistema exibe página de login com todos elementos em branco,
sistema gera captcha,
usuário digita 'daniel' no elemento Usuário,
usuário digita 'daniel123' no elemento Senha,
usuário digita correto o CAPTCHA,
usuário clica sobre o elemento Validar,
sistema valida 'CAPTCHA',
usuário clica sobre o elemento Login,
sistema valida 'Usuário',
sistema valida 'Senha',
sistema encerra login,
então, componente retorna ao sistema autorização de uso.

Exemplo 02: efetuar login com sucesso após falha na verificação do captcha.
Contexto inicial: usuário na página contendo formulário de login e base de dados com usuários cadastrados.
Cenário:
sistema exibe página de login com todos elementos em branco,
sistema gera captcha,
usuário digita 'maria' no elemento Usuário,
usuário digita 'maria123' no elemento Senha,
usuário digita não correto o CAPTCHA,
usuário clica sobre o elemento Validar,
sistema não valida 'CAPTCHA', identifica erro: 'O captcha não foi corretamente preenchido.'
sistema gera novo captcha,
usuário digita correto o CAPTCHA,
usuário clica sobre o elemento Validar,
sistema valida 'CAPTCHA',
usuário clica sobre o elemento Login,
sistema valida 'Usuário',
sistema valida 'Senha',
sistema encerra login,
então, componente retorna ao sistema autorização de uso.

Exemplo 03: efetuar login com sucesso após falha na verificação do usuário.
Contexto inicial: usuário na página contendo formulário de login e base de dados com usuários cadastrados.
Cenário:
  sistema exibe página de login com todos elementos em branco,
  sistema gera captcha,
  usuário digita 'danie' no elemento Usuário,
  usuário digita 'daniel123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Validar,
  sistema valida 'CAPTCHA',
  usuário clica sobre o elemento Login,
  sistema não valida 'Usuário', identifica erro: 'Usuário não cadastrado no sistema ou bloqueado.'
  sistema exibe mensagem de erro,
  sistema exibe página de login com todos elementos em branco,
  sistema gera captcha,
  usuário digita 'maria' no elemento Usuário,
  usuário digita 'maria123' no elemento Senha,
  usuário digita correto o CAPTCHA,
  usuário clica sobre o elemento Validar,
  sistema valida 'CAPTCHA',
  usuário clica sobre o elemento Login,
  sistema valida 'Usuário',
  sistema valida 'Senha',
  sistema encerra login,
então, componente retorna ao sistema autorização de uso.

Caso de uso: Seleção de Ingressos

Dado que a base de dados contém:

quantidade	filme	sessao
2	Zootopia	18:00
2	Deadpool	15:30
1	Zootopia	16:30
12	Zootopia	14:00
3	Deadpool	18:00

Exemplo 04: seleção de ingresso com sucesso.
Contexto inicial: usuário na página seleção de ingressos e base de dados com sessões cadastradas.
Cenário:
 usuário seleciona 'Zootopia' no elemento Filme,
 usuário seleciona '18:00' no elemento Sessão,
 usuário digita '2' no elemento Quantidade,
 usuário clica no elemento Selecionar,
 sistema valida 'Quantidade',
 sistema exibe confirmação de seleção,
então, componente retorna ao sistema seleção de ingresso.

Exemplo 05: seleção de ingresso com sucesso após falha na verificação de quantidade disponível.
Contexto inicial: usuário na página seleção de ingressos e base de dados com sessões cadastradas.
Cenário:
 usuário seleciona 'Deadpool' no elemento Filme,

```
 usuário seleciona '18:00' no elemento Sessão,
 usuário digita '3' no elemento Quantidade,
 usuário clica no elemento Selecionar,
 sistema não valida 'Quantidade', identifica erro: 'A quantidade solicitada
é maior que a disponível'
 sistema exibe mensagem de erro,
 sistema limpa campo quantidade,
 usuário digita '0' no elemento Quantidade,
 usuário clica no elemento Selecionar,
 sistema exibe confirmação de seleção,
então, retorna ao sistema seleção de ingresso.
```

Exemplo 06: seleção de ingresso sem sucesso após cancela seleção.

```
Contexto inicial: usuário na página seleção de ingressos e base de dados
com sessões cadastradas.
 usuário seleciona 'Deadpool' no elemento Filme,
 usuário seleciona '15:30' no elemento Sessão,
 usuário clica sobre o elemento Cancelar,
então, retorna ao sistema sem seleção de ingresso.
```

Caso de Uso: Cadastro de Cliente

Exemplo 07: cadastro de cliente com sucesso.

```
Contexto inicial: Usuário na página de cadastro.
Cenário:
  sistema exibe página de cadastro com todos elementos em branco,
  usuário digita 'Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Ae' no elemento Nome,
  usuário digita '746.396.630-49' no elemento CPF,
  usuário digita 'Lorem ipsum dolor sit amet, consectetuer' no elemento
Endereço,
  usuário digita '22396-364' no elemento CEP,
  usuário digita '(82)67529-0771' no elemento Celular,
  usuário clica sobre o elemento Cadastrar,
  sistema valida 'Nome',
  sistema valida 'CPF',
  sistema valida 'Endereço',
  sistema valida 'CEP',
  sistema valida 'Celular',
  sistema encerra cadastro,
então, envia cadastro para a base de dados.
```

Exemplo 08: cadastro de cliente com sucesso após falha na verificação de cpf.

```
Contexto inicial: Usuário na página de cadastro.
  sistema exibe página de cadastro com todos elementos em branco,
  usuário digita 'Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Ae' no elemento Nome,
  usuário digita '@#!436.715.993' no elemento CPF,
  usuário digita 'Lorem ipsum dolor sit amet, consectetuer' no elemento
Endereço,
  usuário digita '01849-375' no elemento CEP,
  usuário digita '(03)70598-4258' no elemento Celular,
  usuário clica sobre o elemento Cadastrar,
  sistema valida 'Nome',
```

```
  não valida 'CPF', identifica erro: 'O CPF deve ter o formato
"123.123.123-12"'
  sistema exibe mensagem de erro,
  usuário digita '687.000.447-21' no elemento CPF,
  sistema valida 'CPF',
  sistema valida 'Endereço',
  sistema valida 'CEP',
  sistema valida 'Celular',
  sistema encerra cadastro,
então, envia cadastro para a base de dados
```

Exemplo 09: cadastro de cliente com sucesso após falha na verificação do cep.

```
Contexto inicial: Usuário na página de cadastro.
  sistema exibe página de cadastro com todos elementos em branco,
  usuário digita 'Lorem ipsum dolor sit amet, consectetuer adipiscing elit.
Ae' no elemento Nome,
  usuário digita '562.776.269-97' no elemento CPF,
  usuário digita 'Lorem ipsum dolor sit amet, consectetuer' no elemento
Endereço,
  usuário digita '@#!21306-' no elemento CEP,
  usuário digita '(22)63349-7294' no elemento Celular,
  usuário clica sobre o elemento Cadastrar,
  sistema valida 'Nome',
  sistema valida 'CPF',
  sistema valida 'Endereço',
  não valida 'CEP', identifica erro: 'CEP deve ser no formato "12345-123"'
  sistema exibe mensagem de erro,
  usuário digita '(22)63349-7294' no elemento Celular,
  sistema valida 'Celular',
  sistema encerra cadastro,
então, envia cadastro para a base de dados.
```

D
Termo de consentimento

TERMO DE CONSENTIMENTO

Título da pesquisa: Exemplos de uso como ferramenta de comunicação e verificação de Requisitos

Pesquisador:
Fernando Alberto Correia dos Santos Junior (Mestrando DI / PUC-Rio) - fjunior@inf.puc-rio.br Departamento de Informática, PUC-Rio
R. Marques de São Vicente 225, Gávea
CEP 22451-900 - Rio de Janeiro, RJ – Brasil,
Tel.: (21) 3527-1500 / Fax: (21) 3527-1530 / Cel.: (21) 97239-8357
Orientador:
Prof. Arndt von Staa (arndt@inf.puc-rio.br)

Participante: _____

Caro(a) participante,

Desenvolvemos aqui no Departamento de Informática da PUC-Rio pesquisas na área de Engenharia de Software que estuda teorias, métodos e tecnologias para projeto, avaliação e implementação de sistemas que visam melhorar e tornar mais eficiente o processo de planejamento, desenvolvimento, teste e manutenção de um software. Com grande honra gostaríamos de convidar você para participar de nossa pesquisa, que busca verificar o uso de exemplos prático de uso de um software como ferramenta de comunicação e validação de comportamento.

Propósito: O objetivo desta pesquisa é entender como uso de exemplos práticos de comportamento de um software gerados automaticamente a partir da descrição textual de um caso de uso pode, de fato, contribuir na compreensão e validação dos requisitos de um sistema por usuários e clientes, antes do início do desenvolvimento do sistema. Serão apresentadas funcionalidades de um sistema não desenvolvido e será solicitado ao participante que leia e verifique algumas descrições de exemplos práticos de uso dessas funcionalidades. Essa verificação será feita com base na apresentação feita e com base na experiência prévia do usuário no uso de softwares que implementam funcionalidades semelhantes. A realização dessa atividade será observada e faremos algumas entrevistas sobre as impressões do usuário sobre os exemplos apresentados que nos fornecerão dados para determinar se e como o uso de exemplos pode oferecer benefícios para a produção de requisitos de um software.

Procedimentos: A participação neste estudo irá envolver três tipos de atividades. Primeiramente, você será entrevistado e observado no que diz respeito às suas necessidades, dificuldades e hábitos de interação com softwares e sistemas web. Isso será feito com o objetivo de coletar dados que nos permitirão verificar seu grau de conhecimento sobre uso de sistemas web. Depois disso, lhe serão apresentadas funcionalidades de um sistema ainda não desenvolvido e serão apresentadas algumas descrições de exemplos práticos de uso dessas funcionalidades. Será solicitado que você leia cada exemplo e valide

o comportamento descrito com base na apresentação feita e na sua experiência no uso de sistemas que implementam funcionalidades semelhantes. Nessa validação você informará se o uso descrito no exemplo é valido ou não e justificar brevemente sua reposta. Finalmente, ao final, poderemos questionar as suas impressões, e percepções sobre a atividade realizada.

Coleta de dados: Faremos nossas anotações e gravações de áudio quando houver necessidade, se você assim permitir, como forma de coletar os dados para analisarmos posteriormente.

Riscos: As tecnologias a serem utilizadas são comuns e já estão presentes em diversos equipamentos que cercam o seu cotidiano (microcomputadores, telefones celulares, televisores, tablets, internet, etc.). Não se pretende com elas interferir em atividades ligadas à sua saúde ou sobrevivência (ex: alimentação e higiene pessoal), terapêuticas, ou de forma irreversível em nenhuma atividade do seu cotidiano. Eventualmente, pode-se adicionar riscos de segurança digital ao se introduzir mais pontos de conexão com a internet em seu ambiente domestico. Estes não serão maiores do que os já enfrentados normalmente por qualquer usuário de internet em sua casa, com vários dispositivos ligados a um ponto de acesso, e poderão ser desligados a qualquer momento que você quiser. Você pode sentir cansaço ou desconforto durante a participação neste estudo. Serão concedidas todas as oportunidades necessárias para você interromper ou descansar. Você tem pleno direito de solicitar esclarecimentos adicionais, de interromper ou terminar as sessões quando e como quiser. Não há qualquer impedimento para isto nem qualquer necessidade de apresentar uma justificativa ou explicação.

Confidencialidade: As informações coletadas neste estudo serão tratadas dentro das normas éticas de conduta em pesquisa: 1) serão mantidas em arquivos e servidores seguros e serão acessíveis apenas pela equipe de pesquisadores envolvidas no projeto e somente com a finalidade de pesquisa que você consentir; 2) os resultados da pesquisa serão apresentados respeitando-se rigorosamente a sua privacidade e o anonimato de todos participantes, sem a divulgação de nomes, imagens e outros dados que permitam a sua identificação; 3) você poderá solicitar os resultados publicados desta pesquisa se e quando desejar.

Você pode solicitar esclarecimentos adicionais ou optar por não colaborar mais com este estudo a qualquer momento, temporária ou definitivamente, quando e como quiser. Não há qualquer impedimento para isto nem qualquer necessidade de apresentar uma justificativa ou explicação.

Para prosseguir, porém, pedimos que manifeste o seu consentimento por escrito marcando as opções abaixo e assinando este termo, do qual você̂ receberá uma cópia para os seus arquivos:

Li e entendi as informações neste termo: _____
As informações neste termo foram explicadas e esclarecidas para mim: _____
Consinto em participar das atividades descritas acima: _____
Autorizo o uso dados de uso dos sistemas coletados sobre mim: _____

Rio de Janeiro,

_____ Data: _____
Participante

_____ Data: _____
Pesquisador