Ernani César Dos Santos

APLICAÇÃO DE TÉCNICAS DE TESTES DE ACEITAÇÃO AUTOMATIZADOS PARA ESPECIFICAÇÃO DE SOFTWARE EM EDITAIS DE LICITAÇÃO

Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Ciência da Computação
Orientadora: Profª. Drª. Patrícia Vilain

Florianópolis
2019

Ficha de identificação da obra elaborada pelo autor
através do Programa de Geração Automática da Biblioteca Universitária
da UFSC.

Dos Santos, Ernani César
Aplicação de Técnicas de Testes de Aceitação Automatizados para Especificação de Software em Editais de Licitação / Ernani César Dos Santos; orientadora, Patrícia Vilain, 2019.
242 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-Graduação em Ciência da Computação, Florianópolis, 2019.

Inclui referências.

1. Ciência da Computação. 2. Engenharia de Software. 3. Engenharia de Requisitos. 4. Validação de Software. 5. Testes de Aceitação. I. Vilain, Patrícia. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. III. Título.

Ernani César Dos Santos

APLICAÇÃO DE TÉCNICAS DE TESTES DE ACEITAÇÃO AUTOMATIZADOS PARA ESPECIFICAÇÃO DE SOFTWARE EM EDITAIS DE LICITAÇÃO

Esta Dissertação foi julgada adequada para obtenção do Título de "Mestre em Ciência de Computação" e aprovada em sua forma final pelo Programa de Pós-graduação em Ciência da Computação

Florianópolis, 25 de fevereiro de 2019.

_____
Prof. José Luís Almada Güntzel, Dr.
Coordenador do Curso

Banca Examinadora:

_____
Profª. Patrícia Vilain, Drª.
Orientadora
Universidade Federal de Santa Catarina

_____
Profª. Fabiane Barreto Vavassori Benitti, Drª.
Universidade Federal de Santa Catarina

_____
Prof. João Araújo, Dr. (videoconferência)
Universidade Nova de Lisboa

_____
Prof. Raul Sidnei Wazlawick, Dr.
Universidade Federal de Santa Catarina

Este trabalho é dedicado a todos os professores, mestres que não poupam esforços para ensinar e orientar com sabedoria.

AGRADECIMENTOS

À professora Dra. Patrícia Vilain, orientadora deste trabalho, pela oportunidade concedida para fazer parte do Laboratório de Banco de Dados e Engenharia de Software (LEB) da Universidade Federal de Santa Catarina, pelo tempo dedicado, paciência e compreensão, e por compartilhar comigo suas experiências e seu conhecimento.

Aos colegas do LEB que contribuíram em diversas etapas do desenvolvimento deste trabalho.

A todos os meus *subjects* (amostras da população dos experimentos): alunos de graduação do Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Catarina, alunos de graduação do Curso de Bacharelado em Sistemas de Informação da Faculdade Estácio de Florianópolis, colegas do Tribunal de Justiça do Estado de Santa Catarina e aos meus amigos e familiares.

Ao Programa de Pós-graduação em Ciência da Computação, ao Departamento de Informática e Estatística, à Universidade Federal de Santa Catarina por oferecerem este curso gratuitamente e pelo apoio financeiro concedido para a apresentação de artigo científico na *19th International Conference on Agile Software and Systems Development*.

E por fim, não poderia de deixar de agradecer ao meus pais, minha irmã e minha namorada que sempre me deram todo apoio possível (e impossível) em todas as minhas decisões, em todas as minhas escolhas e em todos os meus desafios.

“It is better to write and run incomplete tests than not to run complete tests”.
(Kent Beck, 1999)

RESUMO

Atualmente, as organizações públicas têm optado por terceirizar Serviços de Tecnologia da Informação (STI). A contratação destes serviços é realizada através de um processo licitatório. Neste processo a administração deverá detalhar em um edital de licitação o objeto a ser contratado. Quando o objeto da contratação é um software, seus requisitos são especificados em tópicos utilizando Linguagem Natural (LN). Todavia, especificar requisitos utilizando LN pode resultar em uma série de problemas, como: ambiguidade, inconsistência, incompletude, informações em excesso e repetição de requisitos. Consequentemente, as organizações públicas acabam recebendo produtos que não atendem as suas necessidades. Portanto, é necessário estabelecer uma técnica que permita às organizações públicas definirem uma correlação objetiva entre os requisitos descritos no edital de licitação e o software que deverá ser entregue pelo fornecedor. Acredita-se que, ao utilizar testes de aceitação como ferramenta para auxiliar a especificação de requisitos em editais de licitação, as organizações públicas estarão estabelecendo esta correlação. Existem várias técnicas para especificar testes de aceitação. Este trabalho define critérios técnicos e legais para seleção de técnicas de testes de aceitação que podem ser utilizadas para auxiliar na especificação de requisitos funcionais de software em editais de licitação. Os critérios de seleção definidos foram aplicados em um conjunto de técnicas encontradas a partir de uma Revisão Sistemática de Literatura (RSL). Para avaliar a aderência das técnicas aos critérios de seleção foram utilizados os resultados obtidos através da RSL e da realização de três experimentos. Por fim, para validar os resultados foi conduzido um quarto experimento, cujo objetivo é avaliar a exatidão, completude e esforço da aplicação das técnicas selecionadas em requisitos funcionais similares aos encontrados em editais de licitação. Os resultados mostram que as *Fit tables* e a *Gherkin language* atendem aos critérios estabelecidos e que podem ser empregadas em editas e licitação para auxiliar na especificação de requisitos funcionais.

Palavras-chave: Validação de Software, Testes de Aceitação, Especificação de Requisitos, Terceirização, Licitação.

ABSTRACT

Nowadays, public organizations have opted to outsource information technology services. The process to contracting these services is carried out through a public notice. In this process, public administration must detail in a document the desired object. When the object is software, its requirements are specified in topics using Natural Language (NL). However, requirements specifications written in NL can result in problems, such as: ambiguity, inconsistency, incompleteness, over-specification, and repetition of requirements. As result, public organizations end up receiving products that do not attempt their needs. Therefore, it is necessary to establish a technique that allows public organizations to define an objective correlation between the requirements described in the public notice and the software that must be delivered by the supplier. It is believed that when using acceptance tests as a tool to assist in the specification of requirements in public notices, public organizations will be establishing this correlation. There are several techniques for specifying acceptance tests. This work defines technical and legal criteria for acceptance testing techniques selection that can be used to assist in the specification of functional software requirements in public notices. The defined selection criteria were applied in a set of techniques found through a Systematic Literature Review (SLR). To evaluate the adherence of these techniques to the selection criteria, the results obtained through RSL and three experiments were used. Finally, to validate the results, a fourth experiment was conducted, whose objective is to evaluate the accuracy, completeness and effort of applying the selected techniques in functional requirements similar to those found in public notices. The results show that the Fit tables and Gherkin language meet the established criteria and that can be used in public notices to assist functional requirements specification.

Keywords: Software Validation, Acceptance Testing, Requirements Specification, Outsourcing, Public Notice.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ATDD – *Acceptance Test-Driven Development*
BDD – *Behavior-Driven Development*
CNJ – Conselho Nacional de Justiça
DSL – *Domain Specific Language*
Fit – *Framework for integrated tests*
GUI – *Graphical User Interface*
IN – Instrução Normativa
LEB – Laboratório de Engenharia de Software e Banco de Dados
LN – Linguagem Natural
MPOG – Ministério do Planejamento, Orçamento e Gestão
PPP – Parcerias Público-Privadas
RSL – Revisão Sistemática de Literatura
SERPRO – Serviço Federal de Processamento de Dados
SISP – Sistema de Administração dos Recursos de Informação
SLTI – Secretaria de Logística e Informação
STI – Soluções de Tecnologia da Informação
TDD – *Test-Driven Development*
TI – Tecnologia da Informação
US-UID – *User Scenarios Through User Interactions Diagrams*
UFSC – Universidade Federal de Santa Catarina

SUMÁRIO

1 INTRODUÇÃO

A terceirização corresponde à transferência de um determinado serviço a outra empresa (HÄTONEN e ERIKSSON, 2009). Esta prática é apontada, desde o final da década de 60, como solução para tornar as organizações públicas e privadas mais efetivas (OLIVEIRA, 2005).

Existem várias razões para terceirizar funções do negócio de uma organização, a exemplo da redução de custos e o aumento na qualidade dos serviços prestados, que são alguns dos benefícios que justificam a decisão pela terceirização (KHAN, NIAZI e AHMAD, 2011). Adicionalmente, de acordo com Hätonen e Eriksson (2009), na maioria dos casos, a decisão pela terceirização é fundamentada em objetivos normalmente relacionados à redução de custos, flexibilidade de contratação, falta de mão-de-obra da própria organização e/ou a busca por profissionais com conhecimento especializado.

As organizações públicas têm seguido a tendência definida pelo setor privado, onde as atividades relacionadas aos serviços de Tecnologia da Informação (TI) estão entre as mais terceirizadas (LACITY e WILLCOCKS, 1997). Um estudo realizado em 2014 pela *Grant Thornton*, com líderes de 3.300 empresas, mostra que os serviços de TI ocupam segundo lugar no *ranking* mundial da terceirização, 46% das empresas entrevistadas terceirizam os serviços de TI; perdendo somente para os serviços de contabilidade, que são terceirizados por 49% dos entrevistados. Entre os países que mais terceirizam no mundo, em primeiro lugar está a Suécia e em segundo lugar o Brasil (GRANT THORNTON INTERNATIONAL BUSINESS REPORT, 2014). De acordo com Vazquez, Simões e Albert (2013), podem existir vários níveis de terceirização de serviços de TI, desde a codificação do software até a terceirização de toda a equipe de desenvolvimento.

Todavia, a celebração de um contrato entre a administração pública e um fornecedor requer um processo licitatório, em que a administração deverá descrever em um edital de licitação o objeto da contratação, as regras de julgamento das propostas e o contrato a ser celebrado. A partir deste instrumento de contratação, é promovida uma disputa pública que concede a um ou mais fornecedores o direito de fornecer o objeto licitado à administração pública, dentro das condições estabelecidas no edital (BRASIL, 1993).

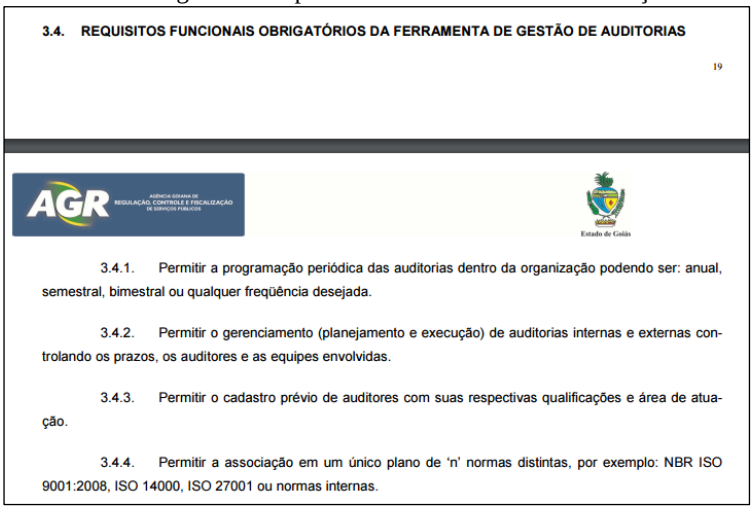
A contratação de Soluções de Tecnologia da Informação (STI), em organizações públicas integrantes do Sistema de Administração dos Recursos de Informação e Informática (SISP) do Poder Executivo Federal, deve estar de acordo com o definido na Instrução Normativa nº

04 de 11 de setembro de 2014 (IN 04/2014) (BRASIL, 2014). Para apoiar (ISO/IEC 25010, 2011)a administração pública na aplicação da IN 04/2014, o governo federal, através do Núcleo de Contratação de TI (NCTI), disponibilizou no portal do Governo Eletrônico, o Guia de Boas Práticas em Contratações de STI, que se trata de um Modelo de Contratação de STI, desenvolvido com base nas fases e processos descritos na IN 04/2014 (BRASIL, 2017).

De acordo com o processo "Planejamento da Contratação de Tecnologia da Informação (PCTI) – Termo de Referência / Projeto Básico" da IN 04/2014, a Equipe de Planejamento da Contratação deverá especificar detalhadamente os requisitos gerais e tecnológicos para contratação (BRASIL, 2014). Assim como exemplificado no Guia de Boas Práticas em Contratações de STI, estes requisitos são apresentados em tópicos numerados e descritos utilizando Linguagem Natural (LN) (BRASIL, 2017).

A Figura 1 ilustra um trecho de um edital de licitação para contratação de um software para gestão de auditorias, que utiliza LN para especificar os requisitos do software.

Figura 1 – Captura de tela de um edital de licitação



3.4. REQUISITOS FUNCIONAIS OBRIGATÓRIOS DA FERRAMENTA DE GESTÃO DE AUDITORIAS

19

AGR AGÊNCIA GOIÂNIA DE REGULAÇÃO, CONTROLE E FISCALIZAÇÃO DE SERVIÇOS PÚBLICOS

Estado de Goiás

3.4.1. Permitir a programação periódica das auditorias dentro da organização podendo ser: anual, semestral, bimestral ou qualquer freqüência desejada.

3.4.2. Permitir o gerenciamento (planejamento e execução) de auditorias internas e externas controlando os prazos, os auditores e as equipes envolvidas.

3.4.3. Permitir o cadastro prévio de auditores com suas respectivas qualificações e área de atuação.

3.4.4. Permitir a associação em um único plano de 'n' normas distintas, por exemplo: NBR ISO 9001:2008, ISO 14000, ISO 27001 ou normas internas.

Fonte: Agência Goiânia de Regulação, Controle e Fiscalização de Serviços Públicos (2012).

O requisito 3.4.1, primeiro item da Figura 1, descreve que o software deve permitir programação periódica de auditorias, podendo ser

anual, semestral, bimestral ou qualquer frequência desejada. A descrição apresentada para o requisito 3.4.1 não é completa e é ambígua, pois, o termo "qualquer frequência desejada" não explicita a real necessidade do usuário. Várias interpretações poderiam ser extraídas deste requisito, por exemplo, que a auditoria pode ser realizada a qualquer momento, ou ainda, que uma outra frequência qualquer, além de anual e bimestral, pode definir a periodicidade das auditorias.

Portanto, especificar requisitos funcionais de um software utilizando LN pode desencadear problemas na interpretação e na comunicação destes requisitos. As principais causas destes problemas são ambiguidade, incompletude, inconsistência, ruídos de especificação, requisitos perdidos, especificação em excesso e requisitos longos (MEYER, 1985) (YOUNG, 2001). Então, não sendo a LN suficiente para comunicar requisitos funcionais de um software, é interessante a aplicação de técnicas que possam complementar estas especificações. Os testes de aceitação, por exemplo, são apontados como uma boa ferramenta para comunicação de requisitos, pois, expressam de maneira objetiva as necessidades do usuário do software (MELNIK, MAURER e CHIASSON, 2006) (RICCA, TORCHIANO, *et al.*, 2009) (TORCHIANO, RICCA e PENTA, 2007).

1.1 JUSTIFICATIVA

Uma pesquisa realizada pelo *Standish Group* (2015), em mais de 10.000 projetos executados entre os anos de 2011 e 2015, com características semelhantes aos dos processos licitatórios no Brasil, aponta que: 7% dos projetos de software foram entregues dentro dos prazos e custos acordados, e com todas as funcionalidades especificadas; 25% foram cancelados antes mesmo de terem sido completados e 68% foram entregues com prazos e custos maiores ou com falta de funcionalidades especificadas no início do projeto. Além disto, esta mesma pesquisa afirma que nos casos em que os projetos respeitaram os limites de prazo e custo, observaram-se características de baixa qualidade no desenvolvimento.

Portanto, no modelo de processo de contratação adotado pelas organizações públicas há falta de correlação entre a especificação do objeto detalhado no edital de licitação e o software entregue pelo fornecedor à administração pública, resultando no insucesso dos projetos de software.

Logo, existe a necessidade de estabelecer uma técnica que permita à administração pública definir uma correlação objetiva entre os

requisitos funcionais de um software, descritos em um edital de licitação, e o produto entregue pelo fornecedor contratado. De acordo com (MELNIK, MAURER e CHIASSON, 2006) (RICCA, TORCHIANO, *et al.*, 2009) (TORCHIANO, RICCA e PENTA, 2007) esta correlação pode ser criada através de testes de aceitação.

Por esta razão, este trabalho propõe a aplicação de testes de aceitação como ferramenta para auxiliar a especificação de requisitos funcionais em editais de licitação para contratação de software, em substituição aos tópicos numerados utilizando LN. Espera-se, portanto, que contratar o desenvolvimento de um software utilizando testes de aceitação como ferramenta para auxiliar a especificação dos requisitos, possibilite à administração pública verificar, de forma objetiva, se o que foi especificado no edital de licitação está sendo efetivamente entregue pelo fornecedor.

1.2 PROBLEMA DE PESQUISA

Quais técnicas de testes de aceitação podem ser utilizadas para auxiliar na especificação de requisitos funcionais em editais de licitação para terceirização do desenvolvimento de *software*?

1.3 OBJETIVOS

1.3.1 Objetivo geral

O objetivo deste trabalho é avaliar a aplicação de diferentes técnicas de testes de aceitação automatizados como ferramentas para auxiliar a especificação e validação de requisitos funcionais em editais de licitação para terceirização de software.

1.3.2 Objetivos específicos

Para realizar o objetivo geral, os seguintes objetivos específicos deverão ser alcançados:

1. Identificar as técnicas de testes de aceitação existentes;
2. Selecionar técnicas de testes de aceitação automatizados que possam ser utilizadas como artefatos para auxiliar na especificação de requisitos em contratos de terceirização de desenvolvimento de software;

3. Aplicar e avaliar as técnicas selecionadas utilizando requisitos funcionais de software semelhantes aos encontrados em editais de licitação.

1.4 METODOLOGIA

O desenvolvimento deste trabalho será conduzido em 6 etapas descritas a seguir:

- Etapa 1: Realização de uma Revisão Sistemática de Literatura (SLR) para identificar as técnicas de testes de aceitação existentes;
- Etapa 2: Definição dos requisitos necessários que devem ser atendidos por uma determinada técnica de teste de aceitação para que ela possa ser utilizada como fermenta de especificação de requisitos em editais de licitação para contratação de software;
- Etapa 3: Seleção das técnicas de teste de aceitação identificadas na Etapa 1 que atendam aos requisitos definidos na Etapa 2;
- Etapa 4: Planejamento dos experimentos;
- Etapa 5: Condução dos experimentos planejados na Etapa 4 utilizando as técnicas de testes de aceitação selecionadas na Etapa 3;
- Etapa 6: Análise da aplicação das técnicas de testes de aceitação selecionadas na Etapa 3, verificando: (i) legibilidade: quão compreensível, por usuários não-técnicos, é a notação utilizada pela técnica de teste de aceitação; (ii) universalidade da especificação: verificar se usuários não-técnicos conseguem especificar sozinhos testes de aceitação para um dado requisito funcional utilizando a técnica; (iii) comunicabilidade: a capacidade da técnica em comunicar requisitos entre as áreas de negócio e os desenvolvedores de software; (iv) completude: a capacidade da técnica em especificar totalmente um requisito que foi anteriormente descrito em LN.

Esta dissertação está dividida em sete capítulos. O capítulo 2 é uma fundamentação teórica que apresenta os conceitos dos principais assuntos abordados neste trabalho. O capítulo 3 é uma Revisão Sistemática de Literatura (RSL) sobre a utilização de técnicas de teste de aceitação automatizadas como ferramenta para auxiliar na especificação de requisitos funcionais de um software.

O capítulo 4 apresenta a definição dos critérios técnicos e legais para a seleção de técnicas de teste de aceitação, que compõem a proposta

desta dissertação. No capítulo 5 são planejados e executados experimentos para, juntamente com os dados obtidos através da RSL, avaliar a aderência das técnicas de teste de aceitação aos critérios definidos no capítulo 4 e selecionar uma ou mais técnicas que atendam aos critérios estabelecidos.

Por fim, o capítulo 6 avalia a aplicação das técnicas de teste aceitação selecionadas no capítulo 5 como ferramentas para auxiliar a especificação de requisitos funcionais de software similares aos encontrados em editais de licitação. O capítulo 7 expõe as considerações finais e propõe trabalhos futuros.

2 FUNDAMENTAÇAO TEÓRICA

Nesta seção serão apresentados os conceitos de licitação, qualidade de software e testes de aceitação.

2.1 LICITAÇÃO

Licitação é um procedimento obrigatório à administração pública, para contratação de serviços, obras de engenharia e aquisição de bens de consumo e patrimoniais. O objetivo da licitação é contratar o fornecedor que apresenta a oferta mais vantajosa, menos onerosa e com melhor qualidade possível para a administração pública. A licitação não pode ocorrer de forma sigilosa, deverá sempre ser um processo público e transparente (MIRANDA, 2012). Celso Antônio Bandeira de Mello (2000) conceitua licitação como:

> "Licitação é o procedimento administrativo pelo qual uma pessoa governamental, pretendendo alienar, adquirir ou locar bens, realizar obras ou executar serviços, outorgar concessões, permissões de obra, serviço ou de uso exclusivo de bem público, segundo condições por ela estipuladas previamente, convoca interessados na apresentação de propostas, a fim de selecionar a que se revele mais conveniente em função de parâmetros antecipadamente estabelecidos e divulgados."

O governo federal, os governos estaduais e municipais, as sociedades de economia mista, as fundações, as autarquias, as empresas estatais e demais entidades vinculadas ao poder público são obrigados a licitar (BITTENCOURT, 2014). Os processos licitatórios devem ser regidos pelos princípios constitucionais da isonomia, legalidade, da impessoalidade, da moralidade, da igualdade, da publicidade, da probidade administrativa, da vinculação ao instrumento convocatório e do julgamento objetivo (MIRANDA, 2012).

2.1.1 Princípios constitucionais da licitação

De acordo com Bittencourt (2014), são os princípios constitucionais de licitação:

- Isonomia: todos os interessados na licitação deverão ser tratados igualmente;
- Legalidade: o agente público só pode fazer aquilo que a lei permite, não podendo agir na omissão dela;
- Impessoalidade: o agente público deve agir sempre a favor do bem comum e não em defesa de interesses pessoais ou de terceiros;
- Moralidade: o procedimento licitatório deve ser ético e o agente público que o conduz deve ter um comportamento honesto e dentro da lei;
- Igualdade: além de escolher a melhor proposta, a administração pública deve assegurar a todos os licitantes igualdade de direitos; são proibidas cláusulas que, no edital ou convite, favoreçam uns em detrimento de outros;
- Publicidade: os atos da administração pública devem ser acessíveis a todos os interessados;
- Probidade Administrativa: o administrador público deve atuar com moralidade, honestidade e exatidão;
- Vinculação ao instrumento convocatório: a administração não pode descumprir as normas e condições do instrumento convocatório (edital ou convite), ao qual o processo licitatório está vinculado;
- Julgamento Objetivo: é proibida a utilização de qualquer elemento, fator sigiloso ou critério secreto, que diminua a igualdade entre os licitantes.

2.1.2 Modalidades de licitação

A modalidade de uma licitação indica o procedimento que irá reger o processo licitatório. A Lei 8.666/1993 prevê cinco modalidades de licitação: concorrência, tomada de preços, convite, concurso, leilão (BRASIL, 1993).

Concorrência é a modalidade de licitação na qual os interessados devem comprovar na fase inicial, denominada habilitação, que possuem os requisitos mínimos de qualificação exigidos no edital para execução de seu objeto. Esta modalidade serve para contratações de qualquer valor, sendo obrigatória para contrações de obras e serviços de engenharia com valor estimado maior que um milhão e meio de reais e para os demais casos quando o valor estimado do objeto for maior que seiscentos e cinquenta mil reais. Esta modalidade é usualmente utilizada para compra

de imóveis, alienação de imóveis públicos, concessão de direito real de uso, licitações internacionais, celebração de contratos de concessão de serviços públicos e celebração de contratos de parcerias público-privadas (PPP) (BITTENCOURT, 2014).

Na modalidade Tomada de Preços, os interessados devem estar devidamente cadastrados ou devem atender a todas as condições exigidas para cadastramento até o terceiro dia anterior à data do recebimento das propostas. Esta modalidade pode ser utilizada para contratações de obras e serviços de engenharia que não possuem valor estimado maior que um milhão e meio de reais e para os demais casos quando o valor estimado do objeto não for maior que seiscentos e cinquenta mil reais (BITTENCOURT, 2014).

O Convite é uma modalidade de licitação utilizada para contratações de obras e serviços de engenharia com valor estimado de até cento e cinquenta mil reais e para os demais casos quando o valor estimado do objeto não ultrapassar oitenta mil reais. Nesta modalidade a administração deverá convidar através de um instrumento convocatório, denominado convite, no mínimo três licitantes, cadastrados ou não, denominados convidados. Uma cópia do convite deverá ser afixada em local apropriado, para que a convocação seja estendida a outros interessados também cadastrados e que manifestarem seu interesse com antecedência de até vinte e quatro horas da apresentação das propostas (BITTENCOURT, 2014).

Concurso é a modalidade de licitação utilizada para escolha de trabalho técnico, científico ou artístico, com prêmios ou remuneração aos vencedores, conforme critérios constantes no edital (BITTENCOURT, 2014).

O Leilão é a modalidade de licitação utilizada para realizar a venda de bens móveis sem utilidade para a administração ou de produtos legalmente apreendidos ou penhorados, ou para a alienação de bens imóveis, a quem oferecer o maior lance, que deverá ser igual ou superior ao valor da avaliação (BITTENCOURT, 2014).

A Lei 10.520/2002 prevê a modalidade pregão. O Pregão é adotado para aquisição de bens e serviços comuns de qualquer valor. São considerados bens e serviços comuns aqueles cujos padrões de desempenho e qualidade podem ser objetivamente definidos em um edital, por meio de especificações. A disputa pelo fornecimento dos bens e serviços é feita em sessão pública. Primeiro os licitantes entregam uma proposta de preço escrita, a partir do valor da menor proposta iniciam-se os lances verbais, vence o licitante que apresentar o menor valor. (BRASIL, 2002). O Decreto 5.450/2005 regulamentou o pregão na forma

eletrônica. A diferença entre o pregão presencial e o eletrônico restringe-se somente à forma de apresentação das propostas e realização dos lances, que no pregão eletrônico são realizados através de plataformas virtuais, como por exemplo o Comprasnet, que é uma plataforma de serviços para compras públicas desenvolvido pelo Serviço Federal de Processamento de Dados (Serpro) (BRASIL, 2005).

2.1.3 Instrumento convocatório

O instrumento convocatório é um documento público que define todas as normas e critérios aplicáveis a uma determinada licitação. Através deste documento, os possíveis interessados tomam conhecimento do processo licitatório (MIRANDA, 2012). Neste documento deverão constar a especificação do objeto – produto ou serviço – a ser licitado, o procedimento adotado, as condições de realização da licitação, a forma de participação dos licitantes, os critérios de aceitabilidade e de julgamento das propostas, e as formas de execução do futuro contrato (BITTENCOURT, 2014).

Há duas formas de apresentação de um instrumento convocatório: edital e convite. O edital é o instrumento convocatório das modalidades: concorrência, pregão, concurso, tomada de preços e leilão. Através deste documento a administração pública faz uma oferta de realização de contrato a todos os interessados que atendam às exigências nele estabelecidas. O edital deve ser publicado na imprensa oficial e em jornal de grande circulação (BITTENCOURT, 2014).

O convite ou carta-convite é o instrumento convocatório utilizado exclusivamente na modalidade de licitação Convite. Este instrumento torna público o desejo da administração em contratar por meio da modalidade convite. O convite não necessita ser publicado na imprensa oficial nem em jornal de grande circulação, porém deve ser afixado com antecedência mínima de 5 dias úteis, em local apropriado, no órgão ou entidade que realiza a licitação (BITTENCOURT, 2014).

2.1.4 Instrução normativa nº 4 de 11 de setembro de 2014

Expedida pela Secretaria de Logística e Informação (SLTI) do Ministério do Planejamento, Orçamento e Gestão (MPOG), a Instrução Normativa nº 04/2014 (IN 04/2014) estabelece as regras disciplinares do processo de contratação de Soluções de Tecnologia da Informação (STI) (BRASIL, 2014), em substituição às previstas pela Instrução Normativa nº 04/2010 (IN 04/2010).

Assim como a IN nº 04/2010, a IN nº 04/2014 deve ser observada pelos órgãos e entidades que integram o Sistema de Administração dos Recursos de Informação e Informática (SISP) (BRASIL, 2014).

2.1.5 SISP

O SISP é disciplinado pelo Decreto nº 7.579 de 11/10/2011, e tem por finalidade: facilitar o acesso e prover suporte a informação; promover a integração e a articulação entre programas de governo, projeto e atividades; estimular o uso racional de recursos de tecnologia da informação (TI); estimular o desenvolvimento, padronização, a integração, a interoperabilidade, a normatização dos serviços de produção e disseminação de informações; propor adaptações institucionais necessárias ao aperfeiçoamento dos mecanismos de gestão dos recursos de TI; estimular e promover a formação, o desenvolvimento e o treinamento de servidores que atuam na área de TI; e definir a política estratégica de gestão de TI (BRASIL, 2017).

2.2 QUALIDADE DE SOFTWARE

A Qualidade de software é uma disciplina da engenharia de software que tem por objetivo garantir a qualidade através da definição e normatização dos processos de desenvolvimento (PRESSMAN e MAXIM, 2016). Entretanto a definição de qualidade de software é algo bastante subjetivo, pois o produto de software é algo intangível. De uma maneira geral, um software pode ser dito de qualidade quando é adequado ao uso e atende aos requisitos funcionais e não funcionais (SOMMERVILLE, 2010).

Segundo a norma ISO/IEC 25010 (2011), para avaliar a qualidade externa e interna do software é necessário levar em consideração seis características: adequação funcional, eficiência de desempenho, compatibilidade, usabilidade, confiabilidade, segurança, manutenibilidade e portabilidade. Estas oito características são subdivididas em subcaracterísticas, que podem ser medidas através de métricas externas e internas.

Existem áreas de processo da engenharia de software que são responsáveis por avaliar a qualidade e conformidade do software produzido. Estas áreas de processo têm por objetivo verificar se a estrutura estática do software (documentação e código fonte) foi construída de forma adequada, se o produto final possui todas as funcionalidades especificadas no início do projeto e se não apresenta

problemas de funcionamento (SOMMERVILLE, 2010). Estas áreas de processo correspondem à validação e verificação do produto (Ver & Val, ou ainda, V & V) (CMMI PRODUCT TEAM, 2010). A validação certifica que está sendo construído o produto correto. A verificação, por sua vez, verifica se o produto está sendo construído da forma correta (SOMMERVILLE, 2010). O foco desta dissertação é a validação do software.

2.2.1 Validação do software

A validação do software é um exame de alto nível do produto de software cujo objetivo é conferir se o software desenvolvido satisfaz as necessidades dos *stakeholders* (usuários, operadores, administradores, gerentes, investidores, etc.). Há duas formas de realizar a validação do software: interna e externa (SOMMERVILLE, 2010).

Na validação interna do software, é considerado que todos requisitos dos *stakeholders* foram entendidos corretamente e que estes requisitos foram devidamente e precisamente documentados e são compreensíveis. Neste caso, se o software atender a todos os requisitos descritos nos documentos, ele é considerado internamente validado (SOMMERVILLE, 2010).

Já a validação externa corresponde ao processo em que os *stakeholders* são indagados quanto à correspondência do produto de software desenvolvido em relação às suas necessidades. Uma validação externa é dita de sucesso quando todos os *stakeholders* aceitam o produto de software e expressam que este satisfaz suas necessidades. A validação externa compreende a realização de testes de aceitação pelos usuários do software (SOMMERVILLE, 2010).

2.2.2 Testes de caixa-preta

Os testes de caixa-preta, também conhecidos como testes de comportamento, são aqueles nos quais o indivíduo que realiza os testes não conhece a estrutura interna do software, seu projeto ou sua implementação. O objetivo deste tipo de teste é verificar se as saídas do software ou de funções do software estão corretas para um conjunto de dados de entrada. O teste de aceitação é um exemplo de teste de caixa-preta (SEO e CHOI, 2006).

2.2.3 Testes de aceitação

Os testes de aceitação são descrições do comportamento esperado do produto de software, geralmente expressadas através de exemplos ou cenários de utilização (MELNIK e MAURER, 2005) (TORCHIANO, RICCA e PENTA, 2007) (LONGO e VILAIN, 2015). A execução dos testes de aceitação é uma etapa inerente no desenvolvimento de sistemas customizados, pois, nesta etapa o cliente/usuário fará um teste formal para decidir se ele deve ou não aceitar o produto entregue (SOMMERVILLE, 2010).

De acordo com Sommerville (2010), existem seis estágios no processo de teste de aceitação, são eles:

1. *Definir critérios de aceitação:* nesta etapa os critérios de aceitação devem ser acordados entre o cliente/usuário e o fornecedor do software. Idealmente, esta etapa deve ocorrer no início do projeto, antes da assinatura do contrato de desenvolvimento.
2. *Planejar testes de aceitação:* decidir sobre os recursos, tempo e orçamento para conduzir os testes de aceitação, bem como, definir um cronograma. A cobertura exigida dos testes e os riscos para o processo de testes também devem ser discutidos neste estágio.
3. *Derivar testes de aceitação:* uma vez que os critérios de aceitação foram estabelecidos, eles precisam ser projetados como cenários de testes de aceitação para verificar se existe ou não um software aceitável.
4. *Executar testes de aceitação:* neste estágio os testes de aceitação são executados pelos usuários finais. Como estes testes, muitas vezes, são executados manualmente, pode ser necessário o treinamento dos usuários antes da execução dos testes.
5. *Negociar resultados de teste*: após a execução dos testes de aceitação, o cliente deverá repassar ao fornecedor os problemas encontrados durante a execução dos testes. Fornecedor e cliente irão discutir a fim de chegar à conclusão se o software é bom o suficiente para colocar em uso.
6. *Rejeitar/aceitar software:* esse estágio envolve a reunião entre os fornecedores de software e os clientes para decidir se o software deve ser aceito. Se o software não for bom suficiente para o uso, então será necessário um maior desenvolvimento para corrigir os problemas identificados. Após concluída a correção dos problemas, a fase de testes de aceitação é repetida.

Portanto, é na fase de testes de aceitação que o cliente/usuário decide se o software entregue atende ou não às suas necessidades (TORCHIANO, RICCA e PENTA, 2007) (MELNIK e MAURER, 2005).

Por outro lado, nos métodos ágeis, como por exemplo *eXtreme Programming* (XP), a fase de testes de aceitação tem um significado diferente. Nestes métodos, o usuário é parte da equipe de desenvolvimento e fornece os requisitos para o software em termos de histórias de usuário. Adicionalmente, o usuário também tem como responsabilidade a definição dos testes de aceitação para cada uma das histórias. Como estes métodos utilizam o conceito de *Test-driven development* (TDD) e os testes de aceitação são automatizados, os testes de aceitação da história sendo desenvolvida devem ser aprovados para que o desenvolvimento do software seja continuado. Sendo assim, nos métodos ágeis, não há uma fase de testes de aceitação em separado (SOMMERVILLE, 2010).

2.2.3.1 TDD

O TDD é um modelo de processo para o desenvolvimento de software no qual os testes são escritos antes do desenvolvimento do software. Este modelo tornou-se conhecido depois de Kent Beck apresentá-lo em seu livro *Extreme Programming Explained: Embrace Change* em 1999. Kent Beck resume o TDD em 5 passos, de acordo com o que segue (BECK, 2003):

1. Um novo caso de teste é escrito.;
2. Todos os casos de testes são executados até ver que um deles falhe;
3. Somente o código fonte necessário para fazer o caso de teste passar é escrito;
4. Todos os casos de teste são reexecutados e é verificado se todos eles passaram;
5. O código fonte é refatorado com o objetivo de remover duplicações.

2.2.3.2 *Acceptance test-driven development* (ATDD)

Em 2002, Ward Cunningham introduziu o *Framework for integrated tests* (Fit). O Fit é um *framework* para especificação e execução de testes de aceitação, no qual, os testes são especificados utilizando tabelas, denominadas *Fit tables*. Estas tabelas servem tanto para

representar os testes, quanto para reportar os resultados (MUGRIDGE e CUNNINGHAN, 2005).

As *Fit tables* podem ser inseridas no processo de desenvolvimento de software utilizando uma abordagem similar ao TDD. Nesta abordagem, os usuários escrevem os testes de aceitação para os requisitos de software desejados. Então, os programadores escrevem os códigos fonte que conectam as *Fit tables* especificadas pelos usuários com o código fonte do software a ser testado. O restante do processo é equivalente aos passos 2 a 5 do TDD. Este processo é denominado de *Acceptance test-driven development* (ATDD), pois os testes de aceitação são escritos antes do software (MUGRIDGE e CUNNINGHAN, 2005).

2.2.3.3 *Behavior-driven development* (BDD).

O BDD é uma abordagem de desenvolvimento ágil de software que incrementa o ATDD. Nesta abordagem são escritos cenários que englobam além dos testes de aceitação, descrições do comportamento esperado do software (ROSE, WYNNE e HELLESØY, 2015).

Os cenários do BDD são escritos antes do desenvolvimento do software através de histórias de usuário. Estas histórias são representadas por meio das *BDD languages*. Um exemplo de *BDD language* é a *Gherkin language*

A *Gherkin language* é uma linguagem de domínio que usa um conjunto de palavras-chave para dar estrutura e significado às histórias de usuário, permitindo sua execução de forma automatizada. Esta linguagem é composta basicamente por 3 comandos *Given*, *When* e *Then*. A palavra *Given* expressa as pré-condições para executar um teste, a palavra *When* expressa as condições ou um comportamento específico, e a palavra *Then* expressa as saídas e/ou mudanças esperadas a partir de um determinado comportamento (ROSE, WYNNE e HELLESØY, 2015).

3 REVISÃO SISTEMÁTICA DE LITERATURA

Uma Revisão Sistemática de Literatura (RSL) foi conduzida de acordo com as orientações propostas por Kitchenham (2007). O objetivo desta revisão sistemática é descobrir técnicas de testes de aceitação, identificar quais destas técnicas são automatizadas, quais ferramentas podem ser utilizadas para automatizá-las, com quais objetivos e por quem (público-alvo) estas técnicas foram empregadas em estudos anteriores.

O resultado desta revisão servirá como base para a seleção de um conjunto de técnicas que, devido aos seus atributos, podem ser utilizadas como ferramenta para especificação e comunicação de requisitos de software em editais de licitação. Esta RSL é classificada como uma revisão secundária de literatura. As etapas realizadas nesta RSL são descritas nas próximas seções.

3.1 QUESTÕES DE PESQUISA

As questões de pesquisa (QP) abordadas nesta RSL são:

- QP1: Quais são as técnicas de teste de aceitação disponíveis nos estudos selecionados?
- QP2: Há ferramentas para automatizar as técnicas identificadas na questão de pesquisa QP1? Quais são estas ferramentas?
- QP3: Nos estudos selecionados, quem é o público-alvo que utilizou as técnicas identificadas na questão de pesquisa QP1?
- QP4: Com quais objetivos foram empregadas as técnicas identificadas na questão de pesquisa de QP1?
- QP5: Nos estudos selecionados, quais técnicas de teste de aceitação foram utilizadas como artefato para comunicar requisitos de software?

O objetivo da QP1 é enumerar as técnicas de testes de aceitação existentes, mostrar um exemplo e uma breve explicação sobre cada uma delas. As demais questões de pesquisa abordadas por esta RSL estão correlacionadas com as técnicas enumeradas na resposta da questão de pesquisa QP1 e estão detalhadas a seguir.

A execução automatizada dos testes de aceitação pode tornar o processo de validação do software mais objetivo, pois quando os testes de aceitação são definidos logo no início do projeto pelos usuários em conjunto com a equipe de desenvolvimento do software, em uma linguagem comum, esses cenários de teste assumem o papel de um contrato formal entre as equipes (ANDREA, 2004). Então, na entrega do

produto, os cenários de teste de aceitação, definidos *a priori*, são executados e de forma transparente e objetiva obtém-se o resultado da aceitação ou rejeição do produto entregue. Como o objetivo deste trabalho é aplicar técnicas de testes de aceitação em licitações públicas, e são requisitos deste tipo processo administrativo transparência e objetividade na validação dos produtos e serviços entregues pelos fornecedores, busca-se, através da QP2, identificar quais das técnicas encontradas são automatizadas e quais são as ferramentas que as automatizam.

Os cenários de testes de aceitação podem estar inseridos em diferentes fases do desenvolvimento de software. Na fase de elicitação e documentação de requisitos, como artefato de especificação dos requisitos, no projeto e codificação do software, como artefato para comunicação de requisitos, na validação do software, como artefato para execução de testes de aceitação, e na manutenção do software, como artefato para execução de testes de regressão (HSIA, KUNG e SELL, 1997) (ARAÚJO, ROCHA, *et al.*, 2007) (SOEKEN, WILLE e DRECHSLER, 2012).

Visto que, cada uma destas fases do processo de desenvolvimento de software envolve papéis diferentes, por exemplo, usuários, analistas de negócio, analistas de sistemas, analistas de teste, engenheiros de software e gerentes de projetos, e que estes papéis possuem conhecimentos e experiências diferentes, logo, os cenários de teste de aceitação podem transpor todo o processo de desenvolvimento de software. Por conseguinte, espera-se que estes cenários possam ser mantidos e entendidos por todas estas pessoas (SOEKEN, WILLE e DRECHSLER, 2012). O objetivo das questões QP3 e QP4 é exatamente este. Na QP3 é mapeado qual é o público-alvo dos estudos, identificando as técnicas que já foram utilizadas em estudos anteriores tanto por usuários técnicos como por usuários não-técnicos. A QP4 explora com quais objetivos as técnicas foram utilizadas nos estudos anteriores, tentando identificar aquelas que tem maior cobertura de todo o processo de software. Os objetivos mapeados são: validação do software; documentação complementar da especificação de requisitos; especificação de requisitos; e testes de regressão.

Por fim, a QP5 identifica quais técnicas foram utilizadas nos estudos selecionados como um artefato para especificar requisitos de software, em substituição às formas mais tradicionais, como os cenários de casos de uso e histórias de usuários.

Para responder à primeira questão (QP1) da RSL, decidiu-se inicialmente que as técnicas de testes de aceitação seriam identificadas de acordo com a notação utilizada para especificar os cenários de teste. Uma

notação é uma coleção de símbolos relacionados em uma determinada estrutura que transmite um significado. Na sequência, definiu-se um processo com quatro atividades, que consistem na identificação, nomeação, classificação e validação das técnicas identificadas nos estudos selecionados. Abaixo são detalhadas estas quatro atividades:

1. Identificação: após a definição do conceito de técnica de testes de aceitação, foram extraídos dos estudos selecionados amostras de cenários de teste de aceitação.
2. Nomeação: na sequência, as amostras foram analisadas e os estudos que utilizavam a mesma técnica (notação) foram agrupados. Então, nomeou-se os grupos de acordo com a denominação da técnica dada nos estudos. Quando o nome dado para uma mesma técnica era diferente de um estudo para outro, o nome do grupo escolhido foi aquele citado pela maior parte dos estudos. Quando nenhum nome era apresentado, arbitrou-se um nome de acordo com as características da técnica.
3. Classificação: agrupados os estudos por técnica, para cada uma destas técnicas identificou-se o tipo de artefato utilizado para comunicar os cenários de teste de aceitação. Desta forma, de acordo com o tipo de artefato, foi atribuída uma classe à técnica. As classes estão descritas na Tabela 1.
4. Validação: para uma técnica ter validade em relação ao objetivo desta dissertação, é necessário que seja possível, com as informações disponíveis nos estudos selecionados, construir novos cenários de teste utilizando-a. Para tanto, foram definidos os fluxos dos casos de uso das funcionalidades sacar e simular financiamento de um caixa eletrônico. Em seguida, foram especificados os testes de aceitação para estes fluxos utilizando cada uma das técnicas identificadas. Especialistas avaliaram os cenários construídos para verificar sua exatidão em relação à notação proposta pela técnica de teste de aceitação.

Tabela 1 – Classes das técnicas de testes de aceitação de acordo com o tipo de artefato utilizado para especificar os cenários de teste.

Classificação	**Detalhes**
Linguagem Natural	Texto plano escrito em Linguagem Natural (LN) ou em uma Linguagem Específica de Domínio (do inglês, *Domain Specific Languauge* – DSL), próximo à linguagem falada, podendo utilizar anotações ou marcações.
Linguagem de Programação	Texto plano escrito em qualquer linguagem de programação, script ou linguagem de marcação, desde que a linguagem de marcação contenha comandos similares aos de uma linguagem de programação.
Tabelas	Um conjunto de dados organizados em colunas complementados ou não por pequenos parágrafos de texto explicativos.

Classificação	Detalhes
Diagramas	Cenários de testes de aceitação representados através de diferentes elementos gráficos que quando conectados corretamente possuem um significado e formam um diagrama.

Fonte: Elaborada pelo autor (2019).

Para responder as questões QP2 à QP5, foi utilizada uma abordagem em dois passos. O primeiro passo foi executado durante a leitura dos estudos selecionados, a cada técnica de teste de aceitação descoberta nesses estudos, as características apontadas para esta técnica que serviriam como base para responder as questões de pesquisa foram enumeradas. Então, em um segundo momento, foram construídas tabelas comparativas entre as técnicas encontradas e suas características. Os dados extraídos dos estudos selecionados para responder a estas questões estão detalhados na Seção 3.5.

3.2 PROCESSO DE BUSCA

Inicialmente, foi definida a chave de busca a ser executada nas bases de dados. A definição desta chave foi realizada considerando os critérios **população**, **intervenção** e **resultados** do PICOC (população, intervenção, comparação, resultados e contexto), assim como sugerido por Kitchenham (2007).

A **população** é o conjunto de elementos afetados pela intervenção. Sendo a **intervenção** os testes de aceitação, logo, a **população** será formada pelos processos da engenharia de software afetados pelos testes de aceitação, que são: a engenharia de requisitos, os testes e/ou a validação do software. No critério **intervenção**, foi empregado como sinônimo do termo *testes de aceitação* o termo *testes funcionais*, pois, esta denominação é utilizada em estudos mais antigos para se referir aos testes de aceitação. Os demais sinônimos de testes de aceitação utilizados na chave de busca foram extraídos de Melnik, Maurer e Chiasson (2006).

Os **resultados** compreendem aquilo que se espera encontrar nos estudos selecionados. De fato, é esperado que os estudos mostrem quais são as técnicas de testes de aceitação, suas ferramentas e como elas podem ser usadas para validar e especificar requisitos de software.

A chave de busca foi formada com base na intersecção dos critérios descritos nos parágrafos anteriores. Para cada um destes critérios foram definidos sinônimos que, por sua vez, foram conectados utilizando o operador booleano “OU” (“OR”), assim como apresentado na Tabela 2. Finalmente, os critérios de intervenção, população e resultados foram conectados pelo operador booleano “E” (“AND”), formando a chave de busca apresentada a seguir:

Intervenção AND População AND Resultados

A chave de busca resultante foi adaptada e executada nas seguintes bases de dados:

- ACM *Digital Library* (dl.acm.or);
- IEEEXplore *Digital Library* (ieeeexplore.ieee.org);
- SciVerse Scopus (scopus.com);
- ScienceDirect – Elsevier (sciencedirect.com);
- Springer (springer.com).

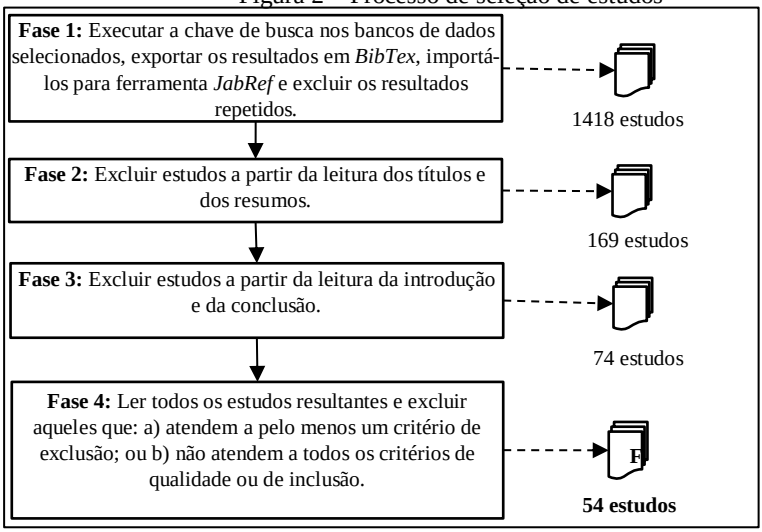
Um processo de seleção dividido em quatro fases foi aplicado aos resultados obtidos. A Figura 2 ilustra este processo e o número de estudos resultantes em cada fase.

Tabela 2 – Termos para os critérios intervenção, população e resultados.

Intervenção	acceptance test OR user test OR customer test OR functional test OR user scenarios OR user stories
População	software validation OR software testing OR requirements engineering
Resultados	technique OR tool OR notation OR requirement

Fonte: Elaborada pelo autor (2018).

Figura 2 – Processo de seleção de estudos



Fonte: Elaborada pelo autor (2018).

Na primeira fase, os resultados obtidos nas bases de dados foram exportados no formato BibTex[1] e importados em uma ferramenta denominada JabRef[2]. Utilizando esta ferramenta foram excluídos os registros duplicados, resultando 1418 estudos.

Na segunda e terceira fases, foram excluídos estudos a partir da leitura dos títulos e resumos, introdução e conclusão, respectivamente. Em uma última fase, os estudos resultantes foram lidos e aqueles que atendiam a pelo menos um dos critérios de exclusão ou que não atendiam a todos os critérios de inclusão e qualidade foram excluídos. Estes critérios são detalhados nas Seções 3.3 e 3.4. Ao final do processo, 54 estudos foram selecionados.

3.3 CRITÉRIOS DE INCLUSÃO E EXCLUSÃO

Os critérios para que um estudo seja considerado elegível para inclusão na RSL são:

- Deve abordar pelo menos uma técnica para especificar testes de aceitação;
- Além da QP1, deve responder pelo menos uma das demais questões de pesquisa;
- Ser avaliado com no mínimo 3 pontos no questionário de avaliação de qualidade descrito na Seção 3.4;
- Estar escrito em Inglês.

Todavia, caso o estudo atenda a qualquer um dos critérios de exclusão abaixo, não poderá compor o conjunto de estudo selecionados para RSL. Os critérios de exclusão são:

- Abordar técnica que não seja de teste de aceitação;
- Abordar a técnica de teste de aceitação, mas não realizar qualquer avaliação sobre a aplicabilidade desta técnica;
- O estudo ser um resumo estendido;
- Estudar somente o processo de teste de aceitação, mas não apresentar nenhuma técnica para especificar os cenários de teste neste processo;

[1] **BibTex** é um formato de arquivo utilizado para descrever e processar listas de referência bibliográficas.
[2] **JabRef** é uma ferramenta para processamento de referências bibliográficas.

3.4 CRITÉRIOS DE QUALIDADE

Os critérios de qualidade foram definidos com base em um questionário formado por cinco questões derivadas dos critérios de inclusão e exclusão de estudos detalhados na Seção 3.3, assim como orientado por Kitchenham (2007). Este questionário foi denominado Questionário de Avaliação de Qualidade, e o objetivo das questões que o compõe é identificar se um estudo primário atendente totalmente, parcialmente ou não atende aos critérios de qualidade estabelecidos. As questões que compõem o Questionário de Avaliação de Qualidade são apresentadas na Tabela 3.

A aplicação do questionário para avaliar a qualidade dos estudos, anteriormente selecionados através da análise do título e do resumo, foi conduzida por pares de revisores. Após a avaliação dos revisores, os estudos que obtiveram 0 pontos em qualquer critério de qualidade, ou menos que 3 pontos no somatório final, foram excluídos. A pontuação foi calculada da seguinte forma:

- Para cada resposta "Atende totalmente" foi atribuído 1 ponto ao estudo, todavia, para cada resposta "Não atende" foi atribuído 0 pontos para o estudo;
- Para estudos que atenderam parcialmente a um critério de qualidade foi atribuído 0,5 ponto por critério;
- Nos casos em que não houve concordância entre as respostas dos revisores, os itens em discordância foram discutidos até atingir o mesmo entendimento entre os dois revisores.

Tabela 3 – Questionário de Avaliação de Qualidade

QA1	O estudo é baseado em uma pesquisa fundamentada e com resultados validados, isto é, o estudo não é um relatório baseado em lições aprendidas ou na opinião de um *expert*?
QA2	Os objetivos da aplicação dos testes de aceitação no estudo podem ser enumerados?
QA3	O estudo pode ser repetido?
QA4	O estudo aborda pelo menos uma técnica de testes de aceitação? Sem pesquisar em outras fontes, é possível especificar um cenário de teste de aceitação qualquer utilizando a técnica abordado pelo estudo?
QA5	Pelo menos uma das seguintes características é avaliada: exatidão, testabilidade, capacidade de leitura, capacidade de escrita, entendimento, eficiência em esclarecer ou comunicar requisitos de software.

Fonte: Elaborada pelo autor (2018).

3.5 EXTRAÇÃO DE DADOS

Para responder tanto as questões de pesquisa quanto o questionário de avaliação de qualidade, foram extraídos dos estudos os dados abaixo:

1. Para cada estudo:

- Uma visão geral do estudo, respondendo as seguintes questões: (i) O que o estudo propõe? (ii) Qual é o problema a ser resolvido pelo estudo e a justificativa? (iii) Qual é a solução adotada? (iv) Quais são os principais resultados alcançados?
- Nome das técnicas de testes de aceitação utilizadas no estudo.

2. Para cada técnica identificada em um estudo:
    - Identificação dos artefatos utilizados para especificar os testes de aceitação, bem como um detalhamento de como este artefato é utilizado;
    - Um exemplo de especificação de teste de aceitação utilizando a técnica;
    - Para as técnicas que são automatizadas, o nome das ferramentas utilizadas no estudo. Entende-se por técnica automatizada aquela que permite a execução automática dos testes de aceitação.
    - Qual é o objetivo técnico da aplicação desta técnica no estudo: validar um software (VAL); complementar a especificação de requisitos, isto é, esclarecer requisitos (CLA); comunicar/especificar requisitos (ESPEC); executar testes de regressão (REG).
    - O estudo acredita que usuários não-técnicos podem utilizar a técnica proposta plenamente, tendo a capacidade de entender e especificar testes de aceitação a partir da utilização da técnica?

3.6 ANÁLISE DOS DADOS

Após a extração dos dados, os resultados foram organizados através de tabelas e/ou diagramas para mostrar:

- Uma lista de técnicas de testes de aceitação e o número de estudos que as utilizaram;
- A classificação das técnicas de acordo com o tipo de artefato utilizado para especificar os testes;
- A aplicação de todas as técnicas de testes de aceitação em um mesmo exemplo de cenário de caso de uso;
- A definição de quais são os pré-requisitos necessários, dentro do processo de desenvolvimento de software, para iniciar a fase de especificação dos cenários de teste das técnicas analisadas;

- Com base nos atributos de cada uma das técnicas: (i) quais técnicas possuem ferramentas disponíveis para implementá-las; (ii) com quais objetivos as técnicas já foram utilizadas em estudos anteriores; (iii) se há indicação para a utilização das técnicas como ferramenta para a comunicação de requisitos; e (iv) se os estudos anteriores apontam que as técnicas podem ser compreendidas por usuário não-técnicos.

3.7 RESULTADOS

Nesta seção são apresentados os resultados alcançados com a RSL, incluindo as respostas às questões de pesquisa. As referências bibliográficas dos 54 estudos selecionados que compõem a RSL estão listadas no APÊNDICE A – Estudos selecionados, onde, para cada estudo foi atribuído um número. Esta numeração é utilizada ao longo desta seção para referenciar os estudos. Foram considerados estudos disponibilizados até novembro de 2017 nas bases de dados listadas na seção 3.2.

3.7.1 QP1 – Quais são as técnicas de testes de aceitação existentes?

A partir da leitura dos trabalhos selecionados, foram identificadas 11 técnicas de testes de aceitação, listadas na Tabela 4. Na primeira coluna desta tabela está o código identificador atribuído a cada uma das técnicas, este código será utilizado ao longo deste trabalho para fazer referência a determinada técnica. Nas demais colunas, são apresentados o nome da técnica, a classe atribuída e os estudos que as abordaram.

Tabela 4 – Técnicas de testes de aceitação e suas classes

Código	Nome da técnica	Classe	Estudo(s)
T01	Fit/*Fit tables*	Tabelas	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 26, 44, 45, 49, 50, 51, 54]
T02	Planilha eletrônica com descrição de contexto	Tabelas	[18]
T03	Planilha eletrônica com validação através do estado dos objetos da aplicação	Tabelas	[19]
T04	Linguagem de Domínio Específico expressa em tabelas	Tabelas	[20]
T05	História de usuário com exemplos	Linguagem Natural	[21, 22, 23, 38, 39, 40, 52]
T06	História de usuário com exemplos e marcações	Linguagem Natural	[24, 25]
T07	Cenário de casos de uso com exemplos e anotações	Linguagem Natural	[27]
T08	BDD/*Gherkin language*	Linguagem Natural	[28, 29, 30, 31, 32, 33, 34, 35, 36, 37]

Código	Nome da técnica	Classe	Estudo(s)
T09	*EasyAccept*	Linguagem de Programação	[41, 42, 43, 53]
T10	Máquina de Estados Finitos	Diagrama	[46]
T11	US-UID	Diagrama	[47, 48]

Fonte: Elaborada pelo autor (2019).

No APÊNDICE D – Técnicas de teste de aceitação são descritas cada uma das classes de testes de aceitação a as técnicas atribuídas a elas, de acordo com o apresentado na Tabela 4.

3.7.2 QP2 – Quais são as ferramentas que automatizam as técnicas de teste de aceitação encontradas nos estudos selecionados?

A resposta para esta questão de pesquisa é apresentada na Tabela 5. Na primeira coluna desta tabela estão as técnicas de teste de aceitação identificadas através dos códigos definidos na primeira coluna da Tabela 4. A segunda coluna da Tabela 5 mostra o nome da ferramenta, a terceira coluna, DISP, indica se a ferramenta está disponível para utilização em ambiente remoto ou instalação local. As colunas ESPEC e EXEC indicam o nível de automação oferecido pela ferramenta para determinada técnica. ESPEC significa que a ferramenta provê funcionalidades para especificação de testes de aceitação e EXEC que a ferramenta possibilita a execução automatizada de testes de aceitação. Algumas ferramentas possuem observações na coluna ESPEC, pois além permitir a especificação dos testes, estas ferramentas possuem outras funcionalidades que conferem maior agilidade no desenvolvimento e reutilização do código de cola.

Tabela 5 – Técnicas de testes de aceitação e suas ferramentas

Técnica	Ferramenta	DISP	ESPEC	EXEC	Estudos
T01	FitNesse	✓	✓	✓	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 44, 49, 50, 51]
	FitClipse	✓	✓	✓	[12, 16]
	AutAT	-	-	✓	[12]
	ReFit	✓	✓	✓	[13, 51]
	FIT	✓	✓	✓	[1, 2, 3]
	Fitinium	✓	✓	✓	[49, 50]
	UCAT	-	✓	-	[17]
T02	Microsoft Excel	✓	✓	-	[18]
T03	Microsoft Excel	✓	✓	-	[19]
T04	Sem nome	-	✓	✓	[20]
T05	Processamento de linguagem natural	✓	✓	-	[38, 39, 40, 52]
	SOLIMVA	-	-	-	[39]
	Text Analyzer	✓	✓	-	[40]

Técnica	Ferramenta	DISP	ESPEC	EXEC	Estudos
	MEReq	-	✓	-	[21]
	TestMEReq	✓	✓	-	[22, 23]
T06	Concordion	✓	✓	✓	[24, 49, 50]
	AnnoTestWeb Run	✓	✓	✓	[25]
T07	Test-duo Framework: Acceplipse + ROBOT	-	✓	✓	[27]
	Acceplipse	-	✓	✓	[27]
T08	Cucumber	✓	✓	✓	[28, 31, 36, 49, 50]
	JBehave	✓	✓	✓	[28, 35]
	NBehave	✓	✓	✓	[28]
	EasyB	✓	✓	✓	[28]
	Jasmine	✓	✓	✓	[28]
	BEAST Tool	✓	Geração de código de cola	-	[29]
	F-TRec	✓	Reuso de código e cola	-	[30]
	Kirby	✓	Geração de código de cola	-	[31]
	BHive	✓	Geração de código de cola	-	[33]
	SnapMind	-	✓	✓	[37]
T09	EasyAccept	✓	✓	✓	[41, 42, 43, 53]
	FLOAppTest	✓	✓	✓	[43]
T10	-	-	-	-	-
T11	S3N4R10 / US-UID	✓	✓	✓	[47, 48]

Fonte: Elaborada pelo autor (2019).

3.7.3 QP3 – Nos estudos selecionados, quem é o público-alvo da utilização das técnicas de teste de aceitação?

As colunas “USR” e “TEC” da Tabela 6, indicam em quais estudos as técnicas foram utilizadas por usuários ou por técnicos, respectivamente. Há estudos nos quais as técnicas foram utilizadas tanto por não-técnicos (USR) quanto por técnicos (TEC), nestes casos o estudo aparece nas duas colunas da tabela.

Tabela 6 – Técnicas de teste de aceitação e público-alvo

Técnicas	USR	TEC
T01	[1, 7, 9, 11, 12, 17, 50, 51, 54]	[1, 2, 3, 5, 4, 54, 9, 10, 12, 13, 14, 15, 16, 17, 26, 45, 49, 44, 50, 51]
T02	[18]	[18]
T03	[19]	-

Técnicas	USR	TEC
T04	-	[20]
T05	[22, 23]	[21, 22, 23, 38, 39, 40, 52,]
T06	[24]	[24]
T07	-	[27]
T08	[29, 34, 35, 37]	[28, 29, 30, 31, 32, 33, 34, 35, 36, 37]
T09	[53]	[41, 42, 43, 53]
T10	-	[46]
T11	[47]	[48]

Fonte: Elaborada pelo autor (2019).

3.7.4 QP4 – Com quais objetivos foram empregadas as técnicas listadas na resposta de QP1?

A resposta para esta questão de pesquisa está apresentada na Tabela 7. A primeira coluna desta tabela identifica as técnicas de testes de aceitação e as colunas subsequentes identificam com qual objetivo as técnicas foram utilizadas. A coluna ESPEC indica os estudos que empregaram os testes de aceitação como artefato para especificar e comunicar requisitos de software para equipe de desenvolvedores. A coluna CLA indica os estudos que empregaram os testes de aceitação como um artefato para complementar a especificação de requisitos. A coluna VAL indica os estudos que empregaram testes de aceitação para validação do software. Por último, a coluna REG indica os estudos em que os testes de aceitação foram utilizados como artefato para a execução de testes de regressão.

Tabela 7 - Objetivo do emprego dos testes de aceitação nos estudos selecionados

Técnica	ESPEC	CLA	VAL	REG
T01	[1, 2, 3, 6, 7, 8, 9, 17, 44, 49, 50, 54]	[4, 5, 6, 7, 8, 9, 14, 15, 50, 54]	[1, 4, 6, 54, 8, 9, 10, 12, 13, 16, 17, 26, 44, 45, 49, 50, 51]	[6, 13, 44, 51]
T02	[18]	[18]	[18]	-
T03	[19]	-	[19]	-
T04	-	-	[20]	-
T05	[21, 22, 23]	[21, 39]	[21, 22, 23, 38, 39, 40, 52]	-
T06	[24]	[24]	[24]	-
T07	[27]	-	[27]	-
T08	[29, 31, 32, 33, 34, 36, 37]	[37]	[28, 29, 30, 32, 33, 34, 35, 36, 37]	[28]
T09	[41, 42, 53]	[42, 53]	[41, 42, 43, 53]	-
T10	-	-	[46]	-
T11	[47, 48]	-	[47, 48]	-

Fonte: Elaborada pelo autor (2019).

3.7.5 QP5 – Nos estudos selecionados, quais técnicas foram utilizadas como artefato para comunicar requisitos de software?

As colunas “CLA” e “ESPEC”, apresentadas na Tabela 7, respondem à questão QP5. Na coluna “CLA” estão os estudos que empregaram as técnicas de teste de aceitação em conjunto com outros artefatos de especificação de requisitos, nestes estudos as técnicas serviram como um complemento da especificação, contribuindo com a comunicação dos requisitos. Na coluna “SPEC” estão os estudos que utilizaram os cenários de teste com ferramenta para especificação e comunicação de requisitos, sem apoio de outros artefatos. Alguns estudos abordaram as duas possibilidades.

3.8 DISCUSSÃO

Nesta seção são discutidos alguns dos resultados alcançados com a RSL que irão contribuir para o desenvolvimento desta dissertação. Adicionalmente, são esclarecidas algumas das decisões tomadas durante o progresso da RSL.

3.8.1 Teste funcional versus Teste de Aceitação

O objetivo do teste de aceitação é validar se o software atende ou não às necessidades do usuário. Sendo assim, este teste se concentra nos processos de negócios e responde à pergunta: "A coisa certa foi construída?" (COHN, 2004).

Por outro lado, a finalidade de um teste funcional é verificar se o produto de software foi construído corretamente, independentemente de o software atender ou não às necessidades do usuário. Esse tipo de teste verifica todos os cenários possíveis, mesmo que seja improvável que alguns destes cenários existam "no mundo real". (EL-ATTAR e MILLER, 2009).

Embora alguns estudos indiquem que técnicas e ferramentas como os scripts do *Selenium*, *JAutomate* e *Canoo Web Test* podem ser utilizadas para realização da validação do *software*, estes estudos, na verdade, estão se referindo à validação interna do software, isto é, aos testes de sistema e não aos testes de aceitação. Algumas destas técnicas e suas ferramentas foram identificadas na última fase de seleção dos estudos da RSL. Os estudos que abordam estas técnicas foram excluídos por não atenderem aos critérios de seleção e qualidade, ou por atenderem a pelo menos um critério de exclusão. A Tabela 8 lista estas técnicas, as ferramentas e os

estudos que as referenciaram. As referências bibliográficas destes estudos estão listadas no APÊNDICE B – Estudos excluídos.

Tabela 8 – Técnicas de validação interna

Técnicas	Estudos	Ferramentas
Linguagens Formais	[1, 2, 3, 4, 5]	AGATHA [2]; TTF [3]; ATL [4]; A compatible Z tool [5].
Record/Replay scripts	[6, 7, 8, 9, 10, 11]	Selenium [6, 7, 11]; JAutomate [8]; A Record/Replay tool [9]; Marathon [10].
Scritps de testes utilizando linguagens de programação	[12, 13, 14, 15, 16]	Validation Framework [12]; Canoo Web Test Framework [13, 16]; Senario-based Validation Framework [14]; LBTest [16].
Cenários de teste gerados de Digramas UML	[17, 18, 19, 20]	UML-Checker [18]; Use Case Agent [20]

Fonte: Elaborada pelo autor (2019).

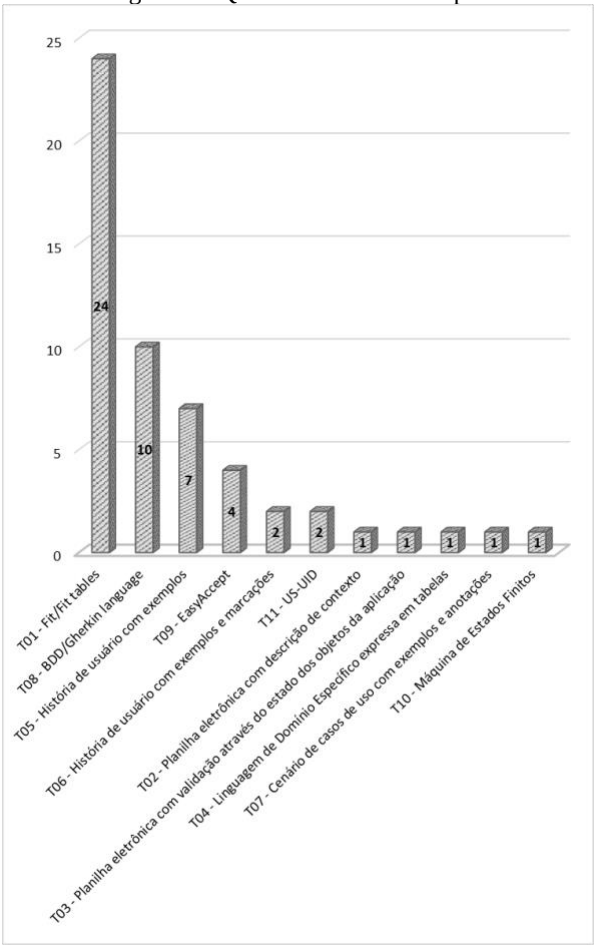
Adicionalmente, técnicas nos quais os cenários de teste de aceitação são construídos com base em elementos da Interface Gráfica do Usuário – *Graphical User Interface* (GUI) –, como por exemplo a técnica Record/Replay scripts, possuem limitações para serem aplicadas como técnicas de teste de aceitação. A principal limitação é o fato de que estas técnicas dependem de uma parte do software ou protótipo executável para produzir os scripts de testes. Além disto, qualquer alteração realizada na GUI, mesmo que não impacte nas funcionalidades do sistema, poderá interferir na execução dos testes, pois estes estão construídos com base nos elementos da GUI.

3.8.2 Ocorrência das técnicas de teste de aceitação

Foram identificadas 11 técnicas de teste de aceitação. O número de ocorrências das técnicas nos estudos selecionados está ilustrado no gráfico apresentado na Figura 3. A técnica com o maior número de ocorrência é o Fit/*Fit tables* (T01). Esta técnica foi abordada em 24 dos 54 estudos, totalizando 44% dos estudos. Este resultado está em consonância com o apontado por Park e Maurer (2008), que declarou que as tabelas FIT são a solução mais amplamente utilizada por organizações para especificar testes de aceitação.

A linguagem de domínio *Gherkin* (T08) foi abordada em 10 dos 54 estudos e as histórias de usuário com exemplos (T05) em 7 dos 54 estudos. Estas técnicas totalizaram, respectivamente, 18,5% e 13% dos estudos. Todavia, os estudos selecionados não revelaram dados que permitissem a comparação do percentual de estudos que abordaram determinada técnica e a popularidade desta técnica em um contexto industrial.

Figura 3 – Quantidade de estudos por técnica



Fonte: Elaborada pelo autor (2019).

3.8.3 Ferramentas para automação de testes de aceitação

Assim como mostrado na Tabela 5, há um grande número de ferramentas disponíveis que possibilitam a automatização das técnicas de teste de aceitação. É possível observar que, apesar da técnica mais referenciada nos estudos ser a T01 – Fit/Fit Tables, a técnica que possui maior número de ferramentas é a T08 – *Gherkin language*.

O grande número de ferramentas que automatizam a T08 é um indicador de que a técnica é amplamente utilizada e possui uma

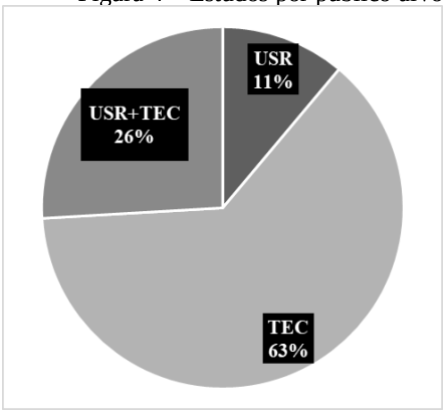
comunidade forte. Inclusive, pode-se observar nos estudos selecionados que há extensões para as ferramentas que permitem automatização do desenvolvimento de código de cola e gestão das *features* (LANDHÄUßER e GENAID, 2012) (KAMALAKAR, EDWARDS e DAO, 2013) (CARRERA, IGLESIAS e GARIJO, 2014) (CARTER e GARDNER, 2016).

3.8.4 Relação das técnicas de aceitação com o público-alvo dos estudos

A Tabela 6 apresenta o público-alvo dos estudos, enquanto que na Tabela 7 estão os objetivos destes estudos. Observe que um mesmo estudo pode ter sido dirigido aos usuários técnicos (TEC) e não-técnicos (USR), e também pode ter mais de um objetivo explorado entre os mapeados, que são: especificação de requisitos (ESPEC), complementação da especificação de requisitos (CLA), validação externa do software (VAL) e realização de testes de regressão (REG).

Fazendo uma análise do número de estudos por público-alvo, há um forte indício que a maioria dos estudos são voltados para perfis técnicos. O gráfico ilustrado na Figura 4 apresenta estes valores: 11% dos estudos selecionados envolveram somente usuários não-técnicos (USR), 63% dos estudos envolveram somente usuários técnicos (TEC), e 26% dos estudos envolveram usuários técnicos e não-técnicos (USR+TEC).

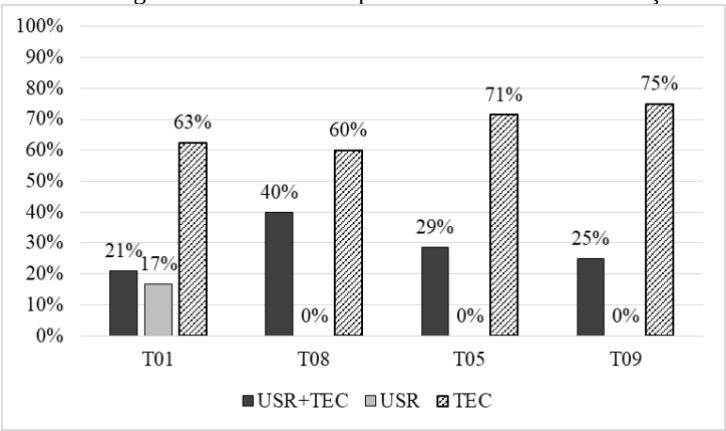
Figura 4 – Estudos por público-alvo



Fonte: Elaborada pelo autor (2019).

A Figura 5 ilustra um gráfico que mostra o percentual do público-alvo por técnica, levando em consideração somente as técnicas que foram abordadas por mais de 5 estudos (T01, T05, T08 e T09). Ao somar os valores das colunas "USR+TEC" com a coluna "USR" obtemos o percentual total de estudos que envolveram somente usuários não-técnicos e os que envolveram usuário técnicos e não-técnicos. Sendo assim, as técnicas de teste de aceitação que mais foram utilizadas por usuário não-técnicos nos estudos foram T01 – *Fit/Fit tables* e T08 – *Gherkin language*.

Figura 5 – Público-alvo por técnica de teste de aceitação.



Fonte: Elaborada pelo autor (2019).

3.8.5 Relação do público-alvo com o objetivo dos estudos

Nos estudos nos quais o público-alvo são os usuários não-técnicos, os objetivos mais recorrentes destes estudos são avaliar a capacidade desses usuários em especificar os cenários de teste de aceitação expressos por uma determinada técnica ou entendê-los para validar o sistema. Já os estudos nos quais o público-alvo é formado por técnicos, os objetivos geralmente se concentram em verificar a capacidade dos cenários de teste de aceitação na comunicação de requisitos de software e na validação do software. Por outro lado, os estudos que envolvem os dois grupos – usuários não-técnicos e técnicos – estão concentrados mais no objetivo de verificar a capacidade dos testes de aceitação em comunicar os requisitos de software.

3.8.6 Aderência das técnicas de testes de aceitação ao processo incremental de desenvolvimento de software

O modelo de processo incremental de desenvolvimento de software é composto pelas atividades de especificação, implementação e validação (SOMMERVILLE, 2010). Estas atividades são intercaladas, e não separadas, e apresentam um rápido *feedback* entre elas. Portanto, para cada incremento do software é realizada uma iteração do processo, onde um conjunto de requisitos é especificado, desenvolvido e validado. Ao final de cada iteração, uma versão executável do sistema é entregue. A Tabela 9 detalha cada uma das fases do processo incremental de desenvolvimento de software (SOMMERVILLE, 2010).

Tabela 9 – Fases do processo incremental de desenvolvimento de software

Fase	**Detalhamento**
Especificação	Os requisitos do sistema são detalhados com os *stakeholders*. Os requisitos são documentados para que possam ser implementados. O resultado desta etapa é o *design* do produto a ser desenvolvido.
Implementação	O *design* do produto é traduzido para códigos fonte, são executados os testes unitários, testes de integração e testes de sistema.
Validação	O software desenvolvido é validado pelos usuários para verificar se atendem aos requisitos definidos no *design* do produto. Nesta etapa são executados os testes de aceitação.

Fonte: Elaborada pelo autor (2018).

Alguns dos estudos selecionados explicitaram o modelo de processo de software no qual foram aplicadas as técnicas de teste de aceitação, esta relação entre modelo, técnica e estudos pode ser observada na Tabela 10.

Tabela 10 – Associação entre técnicas de teste de aceitação, modelos de processos e estudos.

Técnica	**Modelos de processo e metodologias de desenvolvimento de software**			
	Cascata tradicional	**Incremental tradicional**	**Incremental com ATDD**	**Incremental com BDD**
T01	[26]	-	[1, 2, 3, 4, 5, 44, 49, 50]	-
T02	-	-	[18]	-
T03	-	-	-	-
T04	-	[20]	-	-
T05	[39]	-	[22]	-
T06	-	-	[24]	-
T07	-	-	-	-
T08	[35]	-	-	[28, 29, 30, 31, 33, 34, 36, 37]
T09	-	-	[41, 42, 43]	-
T10	[46]	-	-	-
T11	-	-	[47, 48]	-

Fonte: Elaborada pelo autor (2019).

Baseado nos dados apresentados na Tabela 10 e nas informações extraídas das características e da visão geral das técnicas de testes de aceitação, há um forte indício que todas as técnicas podem ser utilizadas desde o início do processo incremental de desenvolvimento de software, com exceção da técnica T07. A técnica T07, cenários de casos de uso com anotações e exemplos, depende da documentação de todos os fluxos dos casos de uso da aplicação para ser implementada.

3.8.7 Testes de aceitação como artefato para a especificação de requisitos funcionais de software

Embora o objetivo principal de todas as técnicas de teste de aceitação é a validação externa do software, os artefatos produzidos para especificar estes testes também podem ser utilizados com outros objetivos, por exemplo:

- Documentação de requisitos de software: a utilização dos testes de aceitação como artefato para documentação de requisitos contribui para um melhor entendimento e comunicação das necessidades do usuário, pois estas necessidades estão representadas de forma precisa através dos testes de aceitação (TALBY, NAKAR, *et al.*, 2005) (SOEKEN, WILLE e DRECHSLER, 2012) (LATORRE, 2013);
- Cenários de teste de regressão: os testes de aceitação, já especificados em outras iterações, podem ser executados como testes de regressão para assegurar que as demais funcionalidades do sistema não foram afetadas após a alteração ou a inclusão de novas funcionalidades (TALBY, NAKAR, *et al.*, 2005);
- Identificação de requisitos: os testes de aceitação podem ser utilizados para melhorar o resultado das atividades de identificação de requisitos, pois, a especificação dos testes obriga o usuário/cliente informar mais detalhes sobre as funcionalidades desejadas, contribuindo para a prevenção de requisitos ambíguos, incompletos e/ou inconsistentes (WANDERLEY, SILVA e ARAÚJO, 2015) (LATORRE, 2013).

A utilização dos cenários de teste de aceitação como artefato para documentar requisitos de software foi avaliada por alguns dos estudos selecionados. As tabelas do *Fit* (T01), por exemplo, contribuem para a redução da interpretação errônea e ambiguidade da documentação de software, todavia, alguns estudos apontam que para documentar um

requisito de software por completo utilizando as *Fit tables* é necessário completá-las com uma descrição textual, como por exemplo, uma história de usuário. Além disto, alguns experimentos envolvendo a aplicação desta técnica mostram que usuários não-técnicos necessitam da assistência de um profissional técnico para elaborar as especificações dos cenários de aceitação.

Por outro lado, as técnicas de teste de aceitação que utilizam tanto as histórias de usuário escritas em linguagem natural para especificar os requisitos do sistema quanto os cenários de teste de aceitação possuem uma melhor receptividade dos usuários pela facilidade de compreensão e desenvolvimento destas histórias. Porém, os problemas em especificar os requisitos utilizando estas técnicas de teste de aceitação acabam sendo os mesmos da utilização de linguagem natural, pois, as histórias são escritas *a priori*, influenciando na qualidade dos testes.

Porém, há técnicas que possuem características similares às histórias de usuário escritas em linguagem natural, mas que são incrementadas por certo formalismo através da utilização de Linguagens Específicas de Domínio. Nestas técnicas os testes de aceitação são expressados de forma declarativa, apresentados em tabelas, como em T04 - Linguagem de Domínio Específico expressa em tabelas, ou através de texto plano, como em *T08 – BDD/Gherkin Language*. Por descreverem não só o critério de aceitação, como acontece com as *Fit tables*, mas por também complementar o critério de aceitação com o comportamento do sistema, os requisitos de sistema expressados através dos testes escritos nestas técnicas tendem a ser mais detalhados, eliminando a necessidade de descrições complementares.

Outra técnica de teste de aceitação utilizada para especificar requisitos de software é o US-UID (T11). Esta técnica expressa cenários de teste de aceitação através diagramas que representam exemplos de entrada e saída esperadas. Identifica-se nos estudos selecionados apenas dois estudos que avaliam a capacidade de usuários não técnicos em criar e compreender cenários de teste de aceitação utilizando esta técnica. Os resultados destes estudos são positivos, mostrando que os usuários não-técnicos têm capacidade de compreender e especificar os cenários de teste. Porém, ambos estudos são dos mesmos autores, além de não haver estudos que validam a aplicação desta técnica como ferramenta para documentação de requisitos.

As técnicas T03, Planilha eletrônica - estado dos objetos, e T09, *EasyAccept*, também foram aplicadas para documentar requisitos, porém, estas técnicas requerem assistência de um projetista. Na técnica T03 os cenários de teste são fortemente acoplados com a arquitetura do sistema,

pois as validações são baseadas em estados dos objetos, sendo necessário conhecimento prévio da estrutura de classes do projeto para especificar os testes. Na técnica T09 o problema está na execução dos testes, que precisa ser realizada por um profissional técnico. Apesar de uma boa técnica poder precisar da assistência de um analista de sistemas, técnicas de teste de aceitação que necessitam de apoio de *designers* para serem especificadas ou executadas não são consideradas técnicas adequadas, tanto para especificação de requisitos, quanto para realização de testes de aceitação. A necessidade de conhecer o código ou estrutura de classes do sistema fere o conceito de teste de caixa-preta, além disto, não há transparência entre a especificação dos testes e sua execução.

4 PROPOSTA

Idealmente, a aplicação de testes de aceitação, como ferramenta para auxiliar a especificação de requisitos funcionais em editais de licitação, assegurará à administração pública o recebimento do software contratado com todas as funcionalidades descritas no edital ou em instrumento próprio previsto por ele. Adicionalmente, acredita-se que a aplicação desta prática nas contratações de produtos software resultará em uma definição mais clara, completa, e explícita dos requisitos funcionais do sistema desejado, diferentemente do que ocorre atualmente com os contratos de terceirização de software nos quais os requisitos são especificados puramente em linguagem natural sem a definição de critérios e cenários de aceitação. Espera-se que, quando aplicada corretamente, esta prática reduza significativamente os riscos de que o produto de software entregue por um fornecedor à administração pública não atenda às necessidades dos seus usuários, não entregue valor para a organização e resulte em prejuízo ao erário.

Todavia, a aplicação de cenários de testes de aceitação como especificação de requisitos em editais de licitação requer uma análise da aderência desta solução aos procedimentos legais e técnicos inerentes aos processos licitatórios. Pois, além da necessidade de assegurar que os cenários de teste de aceitação são capazes de comunicar os requisitos de software, é necessário validá-los como artefato para a especificação de um objeto em um processo de licitação.

Sendo assim, para atingir aos objetivos propostos, este capítulo descreve os critérios técnicos e legais a serem atendidos por uma técnica de testes de aceitação para que ela possa ser utilizada como ferramenta para auxiliar a especificação de requisito funcionais de um software em editais de licitação. Estes critérios serão utilizados para selecionar entre as técnicas de teste de aceitação encontradas na RSL aquelas que serão submetidas a um experimento que permitirá avaliar a aplicabilidade destas técnicas em editais de licitação como ferramenta para auxiliar na especificação de requisitos funcionais de um software.

4.1 CRITÉRIOS PARA SELEÇÃO DAS TÉCNICAS DE TESTE DE ACEITAÇÃO

Na RSL apresentada no capítulo 3, foram identificadas 11 técnicas diferentes para a especificação de testes de aceitação. Para que os cenários de teste escritos em um determinada técnica de teste de aceitação possam compor a solução proposta por esta dissertação, é necessário que a técnica

atenda a um conjunto de critérios. Estes critérios estabelecem quais requisitos uma determinada técnica deve atender para ser utilizada como artefato de especificação de funcionalidades de um software em editais de licitação. Os critérios definidos foram divididos em dois grupos: critérios técnicos e critérios legais.

4.1.1 Critérios técnicos

Estudos anteriores listaram características esperadas de boas técnicas e *frameworks* para condução de teste de aceitação. Andrea (2007) definiu um conjunto de requisitos esperados de uma técnica de teste de aceitação e Talby, Nakar, et al., (2005) sugeriu uma lista de padrões e anti-padrões (o que não fazer) que devem ser atendidos ao se desenvolver um *framework* para a execução de testes de aceitação. A partir destes estudos, foram definidos os seguintes critérios técnicos para a seleção de uma técnica de teste de aceitação:

1. A **notação** deve ser **simples**;
2. A técnica não deve possuir pré-requisitos, podendo ser empregada desde a **elicitação dos requisitos** até a validação do software;
3. Os cenários de testes devem ser capazes de **expressar requisitos funcionais**;
4. A técnica deve produzir **testes automatizados**;
5. A técnica deve ser **independente do projeto** (*design*) do sistema;
6. A técnica deve ser **independente da interface gráfica do usuário**;
7. A técnica deve permitir o **desenvolvimento rápido** e **reutilização dos códigos de cola**.

4.1.1.1 Notação simples

De acordo com os deveres e habilidades definidos para os analistas de negócio descritos no BABOK (*Business Analysts Body of Knowledge*), não se espera de um analista de negócio a realização de quaisquer atividades que possam ser consideradas tecnicamente complexas, por exemplo, a formulação de métodos formais ou uso linguagens complexas para especificar e validar requisitos de software (INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS, 2015). Adicionalmente, cenários de testes de aceitação devem ser entendidos tanto por usuário

não-técnicos quantos por especialistas, e ambos devem ter a capacidade de julgá-los, avaliando a sua completude e exatidão (ANDREA, 2004).

Sendo assim, uma notação de cenários de teste de aceitação deve evitar qualquer complexidade técnica, devendo ser compreendida tanto por usuário não-técnicos como pela equipe de desenvolvimento, aliás, a maioria dos casos de teste é simples e não requer cálculos, condições, loops, recursão e assim por diante. Por exemplo, a utilização de artefatos já conhecidos, como as planilhas eletrônicas, facilita a especificação e comunicação dos requisitos entre os *stakeholders* do projeto (ANDREA, 2004) (TALBY, NAKAR, *et al.*, 2005).

4.1.1.2 Elicitação de requisitos

O processo de desenvolvimento dos testes de aceitação deve iniciar juntamente com a análise de requisitos, pois a especificação dos testes de aceitação contribui para o desdobramento e detalhamento dos requisitos desejados pelos usuários (EL-ATTAR e MILLER, 2009). Sendo assim, uma boa técnica de testes de aceitação não deve depender do *design* do produto e deve ser capaz de comunicar as necessidades dos usuários, preenchendo a lacuna entre as fases de análise e *design* do produto de software (ANDREA, 2004).

4.1.1.3 Expressar requisitos funcionais

A técnica de teste de aceitação deve ser capaz de expressar requisitos de software, ou pelo menos, auxiliar na comunicação destes requisitos, de forma clara, correta, objetiva, completa e sem ambiguidade (SOEKEN, WILLE e DRECHSLER, 2012). Espera-se, portanto, que os cenários de teste de aceitação para um conjunto de requisitos de uma aplicação sejam suficientemente corretos, pois falhas na especificação destes testes podem mascarar não conformidades do sistema em teste (ANDREA, 2004).

4.1.1.4 Testes automatizados

A execução de testes de aceitação manualmente é um processo propenso a erros e incômodo. É desejável desenvolver testes de aceitação executáveis para permitir que sua execução seja realizada sem esforço, com precisão, e facilmente repetíveis (TALBY, NAKAR, *et al.*, 2005).

4.1.1.5 Independente do projeto (*design*) do sistema

Os testes de aceitação devem ser especificados logo no início do projeto de desenvolvimento do software pelos usuários, e se necessário, com ajuda de analistas de negócio e/ou de sistemas. Durante as fases iniciais do desenvolvimento, é comum que uma quantidade significativa de requisitos e detalhes de *design* estejam faltando, ou estejam disponíveis apenas em um nível abstrato. Portanto, é muito importante que uma técnica de testes de aceitação para auxiliar a especificação de requisitos possa ser aplicada sem a disponibilidade de tais informações detalhadas, enquanto se utiliza qualquer informação abstrata disponível (ANDREA, 2004).

4.1.1.6 Independente da interface gráfica do usuário

Ferramentas do tipo *Record and Replay* devem ser evitadas na hora de especificar os testes de aceitação. Além de precisar de protótipos ou de uma parte executável do sistema em testes, estas ferramentas criam cenários de teste não legíveis e que precisam ser regravados a qualquer mudança na interface gráfica (TALBY, NAKAR, *et al.*, 2005). Adicionalmente, nem todas as camadas de uma aplicação devem ser testadas de forma tão completa. Por exemplo, pode-se ignorar a camada de interface do usuário, que não possui lógica de negócios, para economizar tempo e recursos (TALBY, NAKAR, *et al.*, 2005).

4.1.1.7 Códigos de cola reutilizáveis

A atividade de escrita dos códigos de cola para os cenários de testes de aceitação não pode ser um gargalo do processo de desenvolvimento do software, pois, do contrário, a automatização dos testes passará a ser considerada opcional e os testes rapidamente se tornarão incompletos e obsoletos (ANDREA, 2004).

Visto que os testes de aceitação de uma mesma aplicação compartilham componentes comuns, por exemplo, passos repetidos entre vários cenários e os setups (configurações) dos testes, espera-se que a técnica de teste de aceitação permita reutilizar os passos já implementados em um código de cola para compor outros cenários de testes de aceitação (ANDREA, 2004).

4.1.2 Critérios legais

Os critérios legais de seleção de uma técnica de teste de aceitação são os requisitos de negócio, previstos em legislação ou em normas técnicas, que determinam os critérios a serem atendidos pela técnica para que ela possa ser utilizada em contratos de terceirização de software como instrumento para comunicar ou esclarecer requisitos funcionais de um sistema. São os critérios legais:

1. A técnica e suas ferramentas devem ser acessíveis a todos aqueles que tem interesse em estabelecer contrato com a administração pública (BRASIL, 1993);
2. A técnica e suas ferramentas não podem limitar ou restringir participação dos interessados no processo licitatório (BRASIL, 1993);
3. Os critérios de aceitação devem ser parâmetros objetivos para verificar se um bem ou serviço recebido está em conformidade com os requisitos especificados (BRASIL, 2014);
4. A execução dos testes deve ser transparente e igual para todos os participantes (BRASIL, 1993).

5 SELEÇÃO DAS TÉCNICAS DE TESTE DE ACEITAÇÃO

Na RSL, apresentada no Capítulo 3, foram encontradas 11 técnicas de testes de aceitação. Para cada uma destas técnicas foi levantada uma série de informações, tais quais, seus atributos, aplicações e exemplos de especificação. O objetivo deste capítulo é, com base nestas informações apresentadas nos resultados da RSL, selecionar uma ou mais técnicas de teste de aceitação que atendem aos critérios técnicos e legais definidos no Capítulo 4.

5.1 AVALIAÇÃO DOS CRITÉRIOS POR TÉCNICA DE TESTE DE ACEITAÇÃO

A seleção das técnicas de teste de aceitação parte da avaliação de quais dos critérios técnicos e legais, definidos no Capítulo 4, são atendidos por uma determinada técnica. Sendo assim, serão selecionadas as técnicas que atendem a todos os critérios.

Para realizar a seleção, as técnicas e os critérios foram dispostos em uma tabela, assim como apresentado na Tabela 11. A primeira e a segunda colunas desta tabela mostram respectivamente o código e o nome da técnica de teste de aceitação, de acordo com o definido na seção 3.7.1. A terceira coluna apresenta os critérios técnicos e a quarta coluna os critérios legais. A coluna dos critérios técnicos é subdividida em 7 colunas, numeradas de 1 a 7, cada uma destas colunas corresponde ao critério técnico de mesmo número detalhado na Subseção 4.1.1. A coluna critérios legais é subdividida em 4 colunas, numeradas de 1 a 4, cada uma destas colunas corresponde ao critério legal de mesmo número detalhado na Subseção 4.1.2.

Assim, com base nas informações apuradas na RSL apresentada no Capítulo 3, foram marcados com o símbolo "✓" na Tabela 11 os critérios que são atendidos por cada uma das técnicas. Todavia, não foi possível estabelecer com base nestas informações se alguns critérios são atendidos ou não por determinada técnica de teste de aceitação, portanto, estes critérios foram marcados com "?". Os critérios que não puderam ser respondidos com base nos resultados obtidos da RSL são:

- Critério técnico 1: a técnica deve utilizar uma notação fácil, que possa ser entendida e especificada por técnicos e não-técnicos. Apesar de a resposta para a QP3, apresentada na subseção 3.7.3, responder parcialmente este critério, nem todas as técnicas foram avaliadas pelos estudos selecionados quanto a capacidade de

usuários não-técnicos especificar e interpretar os cenários de teste;

- Critério técnico 3: a técnica deve ser capaz de comunicar requisitos de software;
- Critério técnico 7: a técnica deve permitir o desenvolvimento rápido dos códigos de cola;
- Critério legal 2: a técnica deve ser acessível a todos, fácil de entender, com ferramentas gratuitas e comunidade ativa.

Sendo assim, desconsiderando os critérios marcados com "?", somente 5 das 11 técnicas atenderam totalmente aos demais critérios de seleção, e foram grifadas em negrito na Tabela 11, são elas: T01, T06, T08, T09 e T11.

Tabela 11 – Critérios de seleção atendidos por cada técnica de teste de aceitação

#	Técnica	Critérios técnicos							Critérios legais			
		1	2	3	4	5	6	7	1	2	3	4
T01	**Fit/*Fit tables***	**?**	**✓**	**?**	**✓**	**✓**	**✓**	**?**	**✓**	**?**	**✓**	**✓**
T02	Planilha eletrônica com descrição de contexto	?	-	?	-	-	✓	?	✓	?	✓	-
T03	Planilha eletrônica com validação através do estado dos objetos da aplicação	?	-	?	-	-	✓	?	✓	?	✓	-
T04	Linguagem de Domínio Específico expressa em tabelas	?	✓	?	-	✓	✓	?	✓	?	✓	-
T05	História de usuário com exemplos	?	✓	?	-	✓	✓	?	✓	?	✓	-
T06	**História de usuário com exemplos e marcações**	**?**	**✓**	**?**	**✓**	**✓**	**✓**	**?**	**✓**	**?**	**✓**	**✓**
T07	Cenário de casos de uso com exemplos e anotações	?	-	?	-	✓	✓	?	-	?	✓	-
T08	**BDD/*Gherkin Language***	**?**	**✓**	**?**	**✓**	**✓**	**✓**	**?**	**✓**	**?**	**✓**	**✓**
T09	***EasyAccept***	**?**	**✓**	**?**	**✓**	**✓**	**✓**	**?**	**✓**	**?**	**✓**	**✓**
T10	Máquina de Estados Finitos	?	✓	?	-	✓	✓	?	✓	?	✓	-
T11	**US-UID**	**?**	**✓**	**?**	**✓**	**✓**	**✓**	**?**	**✓**	**?**	**✓**	**✓**

Fonte: Elaborada pelo autor (2019).

Porém, a avaliação dos critérios marcados com "?" é mandatória para a seleção de uma ou mais técnicas de teste de aceitação que serão aplicadas como ferramenta para auxiliar na especificação de requisitos funcionais de software em editais de licitação. Como não foi possível responder estes critérios com os resultados obtidos na RSL, uma série de experimentos foi planejada para contemplar estas lacunas da RSL.

Nas próximas seções são apresentados três experimentos. Estes experimentos foram definidos, planejados e projetados de acordo com as orientações propostas por Wholin, Runeson e Höst (2012) e Juristo e Moreno (2001). O objetivo geral dos experimentos é responder aos critérios marcados pelo ponto de interrogação "?" na Tabela 11 para as

técnicas T01, T06, T08, T09 e T11, que são as técnicas que atendem a todos os demais critérios de seleção. Portanto, os experimentos permitirão avaliar:

- A notação utilizada pela técnica é fácil? São avaliadas a legibilidade dos cenários de teste de aceitação e a capacidade de usuários não-técnicos para especificá-los utilizando as técnicas pré-selecionadas;
- A técnica é capaz de comunicar requisitos? É avaliado se uma técnica consegue comunicar todos os detalhes conhecidos de um requisito funcional;
- Os códigos de cola são reutilizáveis? São avaliados o tempo para o desenvolvimento do código de cola e o percentual de reutilização de código;
- Se empregada em um processo licitatório, a técnica de teste de aceitação seria acessível aos interessados da licitação, não limitando a concorrência? Além da avaliação da técnica quanto a sua legibilidade por não-técnicos, será avaliado também a capacidade de técnicos implementarem os códigos de cola para conectar o software em teste ao conjunto de cenários de teste de aceitação previamente definidos, bem como o esforço médio necessário em minutos para implementação de códigos de cola por cenário de teste.

A perspectiva é que através dos resultados dos experimentos seja possível selecionar uma ou mais técnicas de teste de aceitação que possam ser aplicadas em editais de licitação como ferramenta para auxiliar na especificação dos requisitos funcionais.

5.2 EXPERIMENTO 1: INTERPRETAÇÃO E ESPECIFICAÇÃO DE CENÁRIOS DE TESTE DE ACEITAÇÃO POR USUÁRIOS NÃO-TÉCNICOS

O objetivo deste experimento é avaliar se usuários não-técnicos, todos com o mesmo treinamento, conseguem interpretar e especificar cenários de teste de aceitação utilizando as técnicas selecionadas na seção 5.1. A perspectiva é que usuários não-técnicos possam utilizar uma destas técnicas para expressar os requisitos esperados de um software em contratos de terceirização. O objeto deste estudo são os casos de uso sobre o saque em um caixa eletrônico e o cálculo das parcelas mensais de um financiamento, especificados no APÊNDICE C – Cenários de Casos de Uso, e também um requisito hipotético de um software de e-commerce

para gestão de programas de fidelidade que está especificado em linguagem natural no próprio questionário entregue aos participantes.

5.2.1 Participantes do experimento

Os participantes deste experimento são usuários de software, trinta e três deles não possuem formação em Ciência da Computação, Sistemas de Informação, ou áreas afins, um participante possui formação em Sistemas de Informação[3]. Todos eles possuem, no mínimo, ensino superior completo e estão atuando nas áreas em que são graduados. A média de experiência de atuação profissional dos participantes é de 10,2 anos.

5.2.2 Material utilizado para o experimento

O experimento foi conduzido com base em 3 requisitos: sacar dinheiro de um caixa eletrônico, simular o valor das parcelas mensais de um financiamento e calcular o número de pontos ganhos em um programa de fidelidade de acordo com o valor da compra realizada em uma loja virtual.

Para participar do experimento, os participantes receberam um e-mail com instruções básicas e um questionário. No questionário foi disponibilizado o seguinte conteúdo:

- Orientações para a realização do experimento e preenchimento do questionário;
- Doze questões separadas em três grupos: perfil, compreensão e aplicação.
- Endereço da internet para acesso a dois vídeos. O primeiro vídeo é igual para todos os participantes, e explica os conceitos de teste de aceitação. O segundo vídeo explica umas das 5 técnicas de teste de aceitação selecionadas na seção 5.1. Este vídeo varia de um questionário para outro de acordo com o grupo ao qual está associado o participante. A distribuição dos participantes em grupos está detalhada no projeto do experimento na subseção 5.2.4.
- Endereço da internet para um documento que contém exemplos da aplicação da técnica de teste de aceitação. A técnica apresentada nos exemplos é a mesma abordada no segundo

[3] Este aspecto será justificado na subseção 5.2.7.1.

vídeo. Os exemplos foram construídos com base nos fluxos de eventos dos casos de uso que fazem parte dos requisitos objeto deste experimento, especificados no APÊNDICE C – Cenários de Casos de Uso. Estes exemplos também são utilizados como referência para a realização dos outros dois experimentos que envolvem a série proposta nesta dissertação.

Tanto o e-mail encaminhado aos participantes quanto o questionário são mostrados no APÊNDICE N – Materiais do experimento 1.

5.2.3 Questões de pesquisa

Considerando as técnicas T01 (*Fit/Fit tables*), T06 (História de usuário com exemplos e marcações), T08 (*BDD/Gherkin Language*), T09 (*EasyAccept*) e T11 (US-UID), este experimento propõe a resolução das seguintes questões de pesquisa:

- RQ1. Com quais destas técnicas usuários não-técnicos conseguem entender os requisitos funcionais e regras de negócio de um software representados por meio de cenários de teste de aceitação?
- RQ2. Qual é a percepção dos participantes sobre a dificuldade que eles tiveram para entender os requisitos e regras de negócio comunicados através de uma destas técnicas de teste de aceitação?
- RQ3. Com quais destas técnicas usuários não-técnicos conseguem especificar os requisitos funcionais e regras de negócio de um software utilizando cenários de teste de aceitação?
- RQ4. Qual é a percepção dos participantes sobre a dificuldade de especificar cenários de teste de aceitação utilizando uma destas técnicas para um dado requisito de software?

5.2.4 Projeto do experimento

O experimento foi dividido em duas partes. Na primeira parte do experimento foram utilizados dois objetos e cinco tratamentos. Os objetos são os requisitos **(SR1)** "Sacar dinheiro em um caixa eletrônico" e **(SR2)** "Calcular o valor das parcelas mensais de um financiamento". Estes requisitos estão descritos no APÊNDICE C – Cenários de Casos de Uso. Os tratamentos são:

- **(Fit01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T01 - Fit/*Fit tables*;
- **(UserStories01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T06 - História de usuário com exemplos e marcações;
- **(BDD01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T08 - BDD/*Gherkin Language*;
- **(EasyAccept01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T09 - *EasyAccept*;
- **(US-UID01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T11 - US-UID.

Na segunda parte do experimento, foram utilizados um objeto e cinco tratamentos. O objeto é um requisito do módulo do programa de fidelidade de um software para gestão de lojas virtuais, e consiste em: **(SR3)** o sistema deverá atribuir 1 ponto para cada real gasto pelo cliente referente ao valor total da compra menos o frete. Os tratamentos são:

- **(Fit02)** O participante deve especificar o requisito solicitado através de cenários de teste de aceitação escritos utilizando a técnica T01 - Fit/*Fit tables*;
- **(UserStories02)** O participante deve especificar o requisito solicitado através de cenários de teste de aceitação escritos utilizando a técnica T06 - História de usuário com exemplos e marcações;
- **(BDD02)** O participante deve especificar o requisito solicitado através de cenários de teste de aceitação escritos utilizando a técnica T08 - BDD/*Gherkin Language*;
- **(EasyAccept02)** O participante deve especificar o requisito solicitado através de cenários de teste de aceitação escritos utilizando a técnica T09 - *EasyAccept*;
- **(US-UID02)** O participante deve especificar o requisito solicitado através de cenários de teste de aceitação escritos utilizando a técnica T11 - US-UID.

Os participantes foram divididos, aleatoriamente, em 5 grupos, nomeados de A à E. A cada grupo foi atribuído um tratamento, sendo que o tratamento utilizado para a parte 2 do experimento foi equivalente ao utilizado para a parte 1, isto é, cada grupo trabalhou somente com uma

das técnicas de teste de aceitação. Os questionários foram respondidos por cada um dos membros deste grupo, individualmente e sem qualquer suporte adicional. A Tabela 12 resume o projeto do experimento, nesta tabela é ilustrado qual tratamento foi empregado para cada um dos grupos para cada um dos requisitos nas duas partes do experimento.

Tabela 12 – Projeto do experimento 1

Grupo	Parte 1 – Interpretação		Parte 2 – Especificação	Número de participantes
	SR1	SR2	SR3	
Grupo A	(Fit01)	(Fit01)	(Fit02)	9
Grupo B	(BDD01)	(BDD01)	(BDD02)	6
Grupo C	(UserStories01)	(UserStories01)	(UserStories02)	7
Grupo D	(EasyAccept01)	(EasyAccept01)	(EasyAccept02)	5
Grupo E	(US-UID01)	(US-UID01)	(US-UID02)	7

Fonte: Elaborado pelo autor (2019).

5.2.5 Treinamento

Os participantes foram treinados através de videoaulas disponibilizadas no *YouTube*. O endereço da internet para acesso às videoaulas foi informado aos participantes no próprio questionário.

A primeira videoaula explica o conceito de teste de aceitação e qual é a motivação para a aplicação de testes de aceitação como artefato para especificação de requisitos em contratos de terceirização de software. Todos os participantes tiverem acesso a este vídeo, com duração de 6 minutos e 14 segundos.

Já a segunda videoaula é diferente para cada um dos grupos, pois o objetivo desta videoaula é explicar a técnica de teste de aceitação que será utilizada pelo grupo. Há uma pequena variação de duração do vídeo de uma técnica para o de outra, mas o tempo médio dos vídeos é de 8 minutos e 45 segundos. Sendo assim, todos os participantes do experimento receberam o mesmo treinamento.

5.2.6 Procedimento adotado

O experimento foi conduzido em ambiente virtual através da troca de mensagens eletrônicas por redes sociais e por e-mail. Inicialmente, através das redes sociais, foi disponibilizado um convite à participação no experimento destinados a pessoas que não são da área de tecnologia da informação e são usuários de software. Os interessados deveriam responder ao convite informando um e-mail por onde seriam realizadas as futuras interações. Nesta etapa 55 pessoas demonstraram interesse.

Na sequência, o conjunto de interessados foi dividido em cinco grupos, e para cada grupo foi atribuída uma técnica de teste de aceitação, de acordo com o definido no projeto do experimento, apresentado na subseção 5.2.4. Então, foi encaminhado aos grupos o questionário com os materiais necessários à execução do experimento.

De posse dos materiais necessários, os participantes iniciaram a primeira parte do experimento. Nesta parte, os participantes receberam, através de vídeos, treinamento sobre teste de aceitação e sobre a técnica de teste de aceitação atribuída ao seu grupo. Além disto, também tiveram acesso a exemplos de cenário de testes de aceitação utilizando a referida técnica.

Realizado o treinamento e contemplados os exemplos, os participantes foram convidados a responder se compreenderam o conceito de testes de aceitação, e quais são as funcionalidades, e suas respectivas regras de negócio, especificadas através de testes de aceitação nos exemplos apresentados. O objetivo desta parte do experimento é avaliar se os participantes conseguem interpretar os testes especificados através de uma determinada técnica.

Na segunda parte, foi solicitado aos participantes que eles especificassem um requisito de software utilizando determinada técnica de teste de aceitação. O requisito a ser especificado foi apresentado no primeiro vídeo e descrito no próprio questionário em linguagem natural. O objetivo desta etapa do experimento é avaliar se os participantes conseguem especificar testes de aceitação utilizando uma determinada técnica.

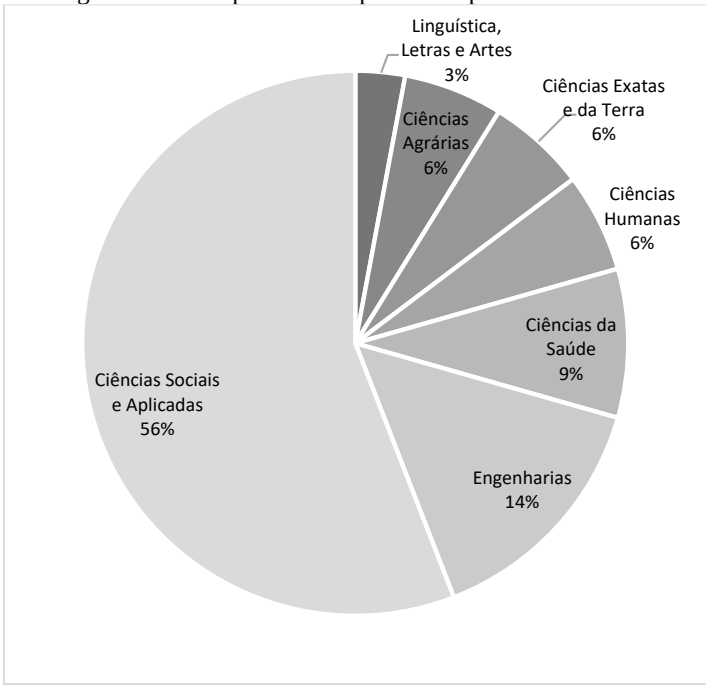
5.2.7 Resultados

Nesta subseção são apresentados e discutidos os resultados dos experimentos. Os dados dos experimentos estão disponíveis em www.leb.inf.ufsc.br/index.php/mestrado/ernani.

5.2.7.1 Perfil dos participantes

O gráfico ilustrado na Figura 6 mostra o percentual de participantes por área de conhecimento de acordo com a formação. A área de conhecimento mais presente é das Ciências Sociais e Aplicadas, com 56% dos participantes. Nesta área, a formação predominante dos participantes é bacharelado em Direito.

Figura 6 – Participantes do experimento por área de conhecimento



Fonte: Elaborada pelo autor (2019).

De acordo com o projeto e o procedimento estabelecido para o experimento, apresentados respectivamente nas subseções 5.2.4 e 5.2.6, os interessados em participar do experimento foram divididos em 5 grupos, com 11 membros cada. A cada grupo foi atribuído um tratamento, isto é, cada grupo utilizou uma das cinco técnicas de teste de aceitação. Todavia, dos 55 interessados, somente 34 (62,8%) participantes responderam ao questionário no prazo estabelecido. Na Tabela 13, são apresentados por grupo: a técnica de teste de aceitação utilizada, o número de questionários respondidos e a média em anos de experiência profissional dos participantes.

Na Tabela 13, ao lado do número de participantes que utilizaram a técnica T09 – *EasyAccept*, há um asterisco. O objetivo deste asterisco é chamar atenção para o seguinte fato: apesar de 5 pessoas terem respondido ao questionário sobre esta técnica, são considerados para efeitos de análise dos dados somente as respostas de 4 dos 5 questionários devolvidos pelos participantes. Um dos questionários foi eliminado, pois, foi respondido por um técnico. Esta participação extraordinária foi

necessária devido à constatação realizada durante o decurso do experimento de que os usuário não-técnicos não estavam conseguindo responder ao questionário que avaliava o *EasyAccept*. Desta forma, com o intuito de validar os materiais desenvolvidos para o treinamento dos usuários não-técnicos, um técnico, não especialista em testes de aceitação e que nunca teve contato com qualquer uma das técnicas disponíveis, graduado em Sistemas de Informação e com especialização em gestão de projetos, foi convidado a responder ao questionário sobre T09. Visto que este participante respondeu com sucesso todas as questões, demonstrado habilidade clara para entender e especificar requisitos de teste utilizando esta técnica, há um forte indício que os materiais produzidos para o experimento eram válidos. Mais tarde, um usuário não-técnico também conseguiu atingir níveis satisfatórios de compreensão e capacidade de especificação.

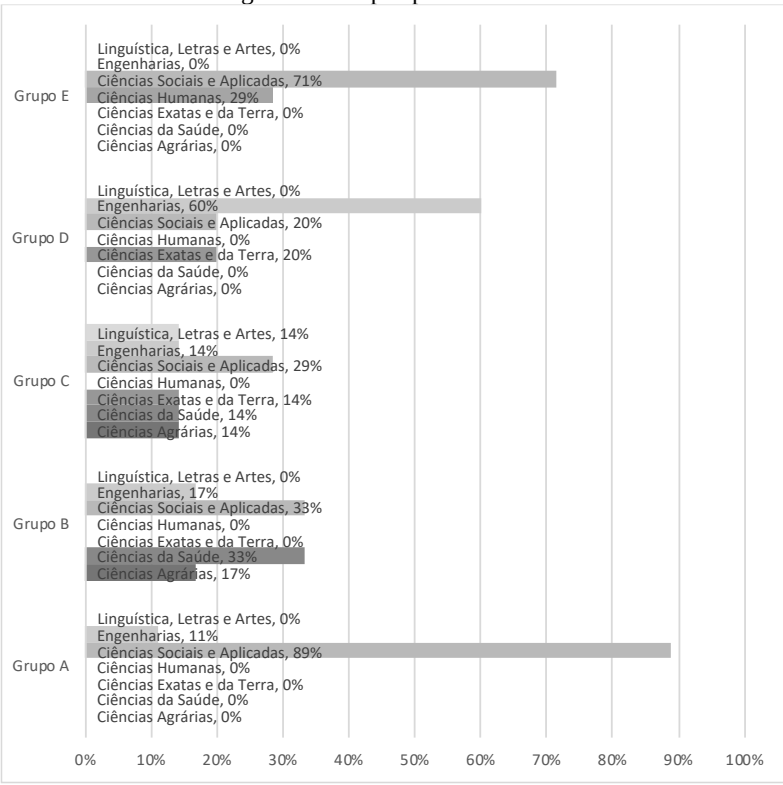
Tabela 13 – Participantes por grupo

Grupo	**Técnica de teste de aceitação**	**Participantes efetivos**	**Média da experiência profissional em anos**
Grupo A	T01 – Fit/*Fit tables*	9	12,11
Grupo B	T08 – BDD/*Gherkin Language*	6	4,12
Grupo C	T06 – Histórias de usuário com exemplos e marcações	7	16,28
Grupo D	T09 – *EasyAccept*	5*	7,4
Grupo E	T11 – US-UID	7	9

Fonte: Elaborada pelo autor (2019).

Considerando somente os participantes que responderam ao questionário no prazo determinado, a composição dos grupos quanto a área de conhecimento dos participantes está apresentada no gráfico ilustrado na Figura 7. Este gráfico apresenta o percentual de participantes por área de conhecimento para cada um dos grupos. A distribuição dos participantes nos grupos foi realizada de forma aleatória, pois como o autor conhecia a maioria dos participantes, uma distribuição não aleatória poderia influenciar os resultados. O Grupo C é o que possui a formação mais heterogênea, o que é positivo para o estudo, pois a população são os usuários de produtos de software transacionais que já participaram de projetos de definição ou validação de requisitos funcionais. Por outro lado, os Grupos A e E são os mais homogêneos. Em áreas de conhecimento como a das engenharias os participantes já tiveram contato com disciplinas de programação, o que leva a acreditar que participantes desta área de conhecimento tendem a ter menos dificuldade, sendo assim, um grupo mais homogêneo, formando em sua maioria por engenheiros, pode apresentar resultados tendenciosos.

Figura 7 – Grupos por área de conhecimento



Fonte: Elaborada pelo autor (2019).

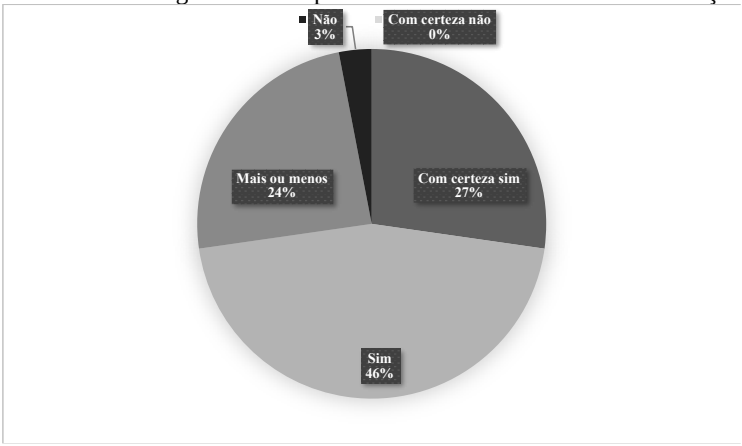
5.2.7.2 Compreensão do conceito de teste de aceitação

Após assistirem aos vídeos, os participantes foram questionados se entenderam o conceito de teste de aceitação. Cada participante conferiu uma nota na escala de 1 a 5, onde 1 representa com certeza sim e 5 com certeza não. Observe o gráfico ilustrado na Figura 8, a maioria absoluta dos participantes (73%) respondeu que com certeza sim ou sim, entenderem o conceito de teste de aceitação.

Visto que a perspectiva do experimento é a utilização dos cenários de teste de aceitação como ferramenta para auxiliar a especificação dos requisitos de um software em contratos de terceirização, é necessário que além de assimilar o conceito de teste de aceitação, os usuário não-técnicos também devem ser capazes de compreender e construir cenários de teste de aceitação. Portanto, nas etapas posteriores do experimento, foram

avaliadas as capacidades dos participantes para entender e especificar cenários de teste de aceitação.

Figura 8 – Compreensão do conceito de teste de aceitação



Fonte: Elaborada pelo autor (2019).

5.2.7.3 Legibilidade dos cenários de teste de aceitação

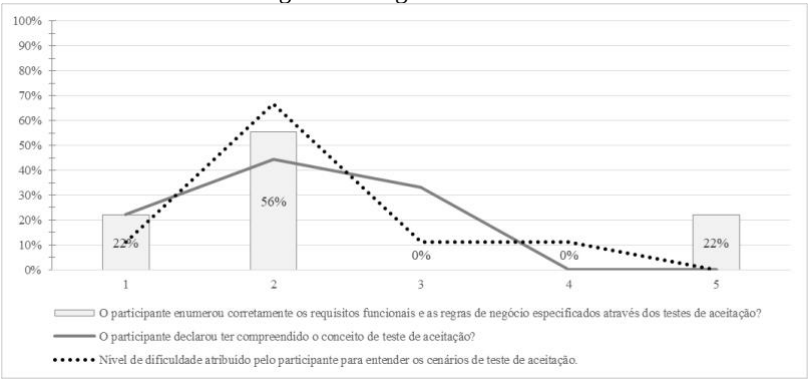
Foram disponibilizados aos participantes documentos com os cenários de teste de aceitação de dois requisitos de um software bancário. O objetivo é avaliar se os usuários não-técnicos são capazes de identificar nestes documentos quais são as funcionalidades e regras de negócio esperadas para os requisitos. A técnica de teste de aceitação utilizada para escrever o cenário de teste nos documentos varia de acordo com o grupo ao qual o participante do experimento está relacionado. A formação dos grupos e a distribuição das técnicas de teste de aceitação são apresentadas nas Tabelas Tabela 13 e Tabela 12, respectivamente. Nas próximas subseções são apresentados os resultados obtidos por técnica de teste de aceitação (grupo do experimento).

5.2.7.3.1 T01 – Fit/Fit tables – Grupo A

Na Figura 9 é ilustrado o gráfico de análise da legibilidade dos cenários de teste de aceitação escritos utilizando *Fit tables.* Neste gráfico são apresentados os valores para três variáveis: (i) legibilidade dos testes; (ii) compreensão do conceito de testes de aceitação, e (iii) nível de

dificuldade atribuído pelo participante para entender os cenários de teste de aceitação.

Figura 9 – Legibilidade das Fit *tables*



Fonte: Elaborada pelo autor (2019)

No gráfico ilustrado na Figura 9, o eixo das ordenadas é o percentual de participantes do experimento, enquanto o eixo das abscissas é a resposta que os participantes atribuíram a uma pergunta. A linha contínua do gráfico ilustrado na Figura 9 apresenta, na escala de 1-5, o nível de compreensão do conceito de teste de aceitação declarado pelo participante. O valor 1 representa que o participante declarou que com certeza entendeu o conceito de teste de aceitação e valor 5 representa que o participante declarou que com certeza não entendeu o conceito de teste de aceitação. Neste mesmo gráfico, representado pela linha pontilhada, está o nível de dificuldade que o participante atribui para o entendimento dos cenários de teste de aceitação representados através das *Fit tables*. Esta variável também está valorada em uma escala de 1 a 5, onde o valor 1 representa que o participante julgou muito fácil entender os cenários de teste de aceitação expressados através de *Fit tables*, enquanto que, o valor 5 representa que o participante julgou muito difícil entender estes cenários de teste de aceitação.

Por fim, as barras representam o percentual de participantes que:

1. Acertaram totalmente: identificaram corretamente todas as funcionalidade e regras de negócio;
2. Acertaram: identificaram corretamente todas as funcionalidade e regras de negócio, porém, confundiram os conceitos de funcionalidade e regra de negócio;

3. Acertaram parcialmente: identificaram a maior parte das funcionalidade e regras de negócio e não incluíram na resposta nenhuma informação errada.
4. Erraram: apesar de terem identificado algumas funcionalidades e regras de negócio corretamente, os participantes incluíram em sua resposta informações erradas ou controversas.
5. Erraram totalmente: o participante deixou a questão em branco ou respondeu com informações totalmente incorretas.

Sendo assim, o gráfico ilustrado na Figura 9 mostra que 78% dos participantes acertaram totalmente ou acertaram a resposta da pergunta do questionário que lhes solicitava identificar as funcionalidades e regras de negócio especificadas com o auxílio das *Fit tables.* Os outros 22% dos participantes erraram totalmente a resposta.

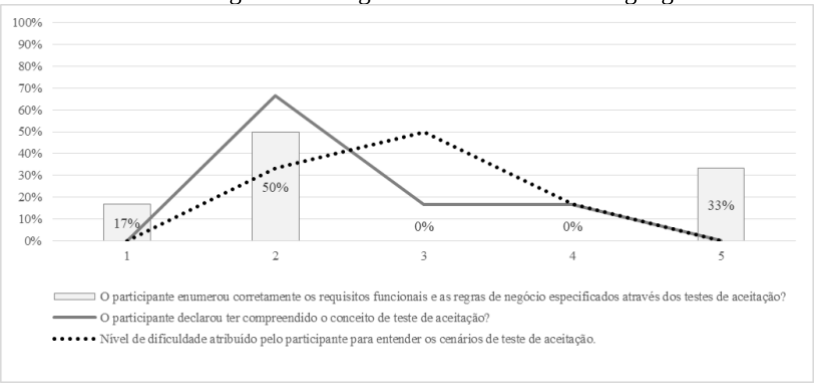
A opinião dos participantes quanto à dificuldade para compreender os cenários de teste de aceitação, expressos através das *Fit tables*, está em consonância com a pergunta do questionário para identificação das funcionalidades e regras de negócio especificadas com o auxílio desta técnica. A maioria dos participantes do grupo, 78%, responderam que consideraram muito fácil (11%) ou fácil (67%) compreender os cenários de teste representados através das *Fit tables.*

5.2.7.3.2 T08 – BDD/Gherkin language – Grupo B

Na Figura 10 é ilustrado o gráfico de análise da legibilidade dos cenários de teste de aceitação escritos utilizando *Gherkin language*. Este gráfico possui as mesmas variáveis que o gráfico ilustrado na Figura 9, consequentemente, são aplicadas a elas o mesmo esquema de atribuição de valores e a mesma representação gráfica que foram aplicados para os cenários escritos utilizando *Fit tables*.

Portanto, o gráfico ilustrado na Figura 10 mostra que 67% dos participantes *acertaram totalmente* ou *acertaram* a resposta da pergunta do questionário que lhes solicitava identificar as funcionalidades e regras de negócio especificadas com o auxílio da *Gherkin Language.* Os outros 33% dos participantes *erraram totalmente* a resposta.

Figura 10 – Legibilidade da Gherkin Language



Fonte: Elaborada pelo autor (2019).

Apesar da maioria dos participantes ter acertado a essa resposta, a percepção deles quanto à dificuldade para compreender os cenários de teste de aceitação não acompanha a mesma proporção. A metade dos participantes do grupo (50%) responderam que não estão certos se é fácil ou difícil compreender os cenários de teste escritos na *Gherkin Language*, 33% declaram ser fácil e 17% difícil.

5.2.7.3.3 T06 – Histórias de usuário com exemplos e marcações – Grupo C

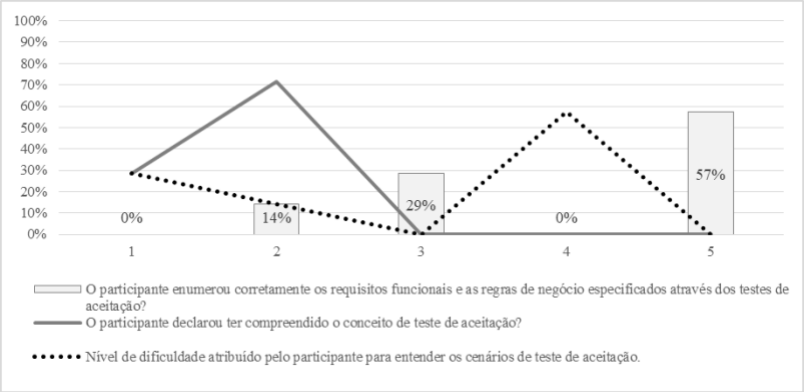
Na Figura 11 é ilustrado o gráfico de análise da legibilidade dos cenários de teste de aceitação escritos utilizando Histórias de Usuário com exemplos e marcações. Este gráfico possui as mesmas variáveis que o gráfico ilustrado na Figura 9 e, consequentemente, são aplicadas a elas o mesmo esquema de atribuição de valores e a mesma representação gráfica que foram aplicados para os cenários escritos utilizando *Fit tables*.

Diferente das demais técnicas, o gráfico ilustrado na Figura 11 mostra que a maioria dos participantes (57%) erraram totalmente a resposta da pergunta do questionário que lhes solicitava identificar as funcionalidades e regras de negócio especificadas com o auxílio das Histórias de usuário com exemplos e marcações. Os mesmos 57% dos participantes também apontaram que consideraram a técnica difícil.

Ressalta-se, no entanto, que apesar dos participantes terem considerado e técnica difícil e a maioria deles não ter identificado com sucesso as funcionalidades e regras de negócio apresentadas no exemplo disponibilizado no questionário, todos os participantes responderam que

compreenderam o conceito de teste de aceitação. Dada a proximidade das Histórias de usuário com exemplos e marcações com a linguagem natural, a única explicação para este resultado é o fato desta apresentar os mesmos problemas daquela: ambiguidade e excesso de informações desnecessárias.

Figura 11 – Legibilidade das Histórias de usuário com exemplos e marcações.



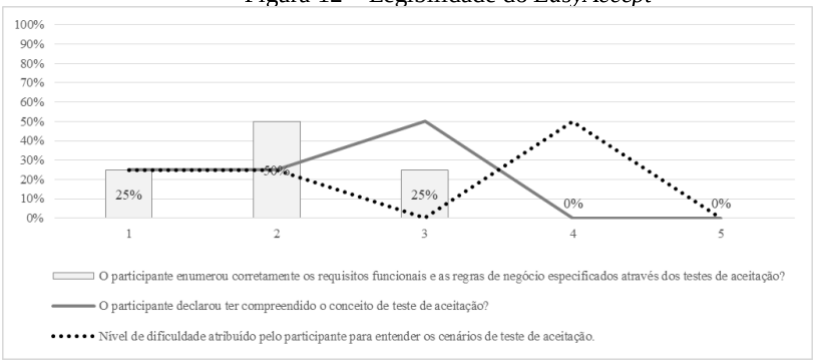
Fonte: Elaborada pelo autor (2019).

5.2.7.3.4 T09 – EasyAccept – Grupo D

Na Figura 12 é ilustrado o gráfico de análise da legibilidade dos cenários de teste de aceitação escritos utilizando o *EasyAccept*. Este gráfico possui as mesmas variáveis que gráfico ilustrado na Figura 9, consequentemente, são aplicadas a elas o mesmo esquema de atribuição de valores e a mesma representação gráfica, aplicados para os cenários escritos utilizando o *EasyAccept.*

Assim como mostra o gráfico ilustrado na Figura 12, 75% dos participantes do Grupo D *acertaram totalmente* ou *acertaram* a resposta da pergunta do questionário que lhes solicitava identificar as funcionalidades e regras de negócio especificadas com o auxílio do *EasyAccept.* Quanto à percepção dos participantes sobre a dificuldade do entendimento da técnica, metade do grupo (50%) julgou que os cenários de teste de aceitação produzidos utilizando esta técnica são *muito fáceis* ou *fáceis* de entender, a outra metade julgou que os cenários são *difíceis* de entender.

Figura 12 – Legibilidade do *EasyAccept*

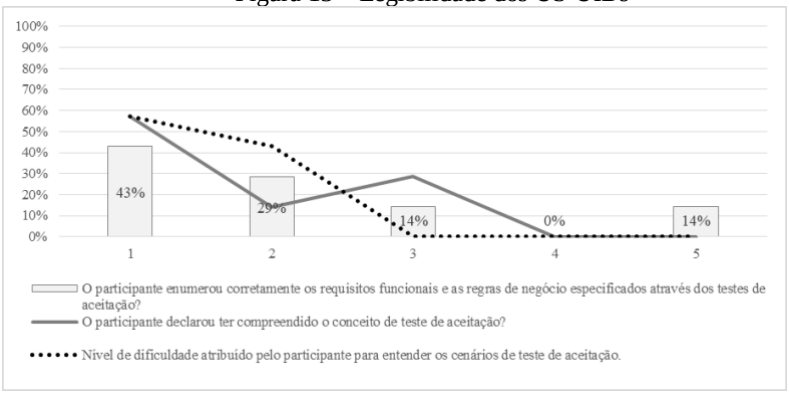


Fonte: Elaborada pelo autor (2019).

5.2.7.3.5 T11 – US-UID – Grupo E

Na Figura 13 é ilustrado o gráfico de análise da legibilidade dos cenários de teste de aceitação escritos utilizando os US-UIDs. Este gráfico possui as mesmas variáveis que gráfico ilustrado na Figura 9, consequentemente, são aplicadas a elas o mesmo esquema de atribuição de valores e a mesma representação gráfica que foram aplicados para os cenários escritos utilizando *Fit tables*.

Figura 13 – Legibilidade dos US-UIDs



Fonte: Elaborada pelo autor (2019).

O gráfico ilustrado na Figura 10 mostra que 72% dos participantes *acertaram totalmente* ou *acertaram* a resposta da pergunta do

questionário que lhes solicitava identificar as funcionalidades e regras de negócio especificadas com o auxílio dos US-UIDs. Os outros 28% dos participantes estavam divididos igualmente entre os que *erraram totalmente* a resposta e os que *acertaram parcialmente*. Todos os participantes julgaram que os cenários de teste de aceitação produzidos utilizando esta técnica são *muito fáceis* (57%) ou *fáceis* (43%) de entender.

5.2.7.4 Especificação de cenários de teste de aceitação

Na segunda parte do experimento, foi solicitado aos participantes para especificar o cenário de teste de aceitação para um dado requisito. Para escrever a especificação o participante deveria utilizar a técnica de teste de aceitação atribuída ao seu grupo, esta técnica é a mesma que o participante utilizou na primeira parte deste experimento. A formação dos grupos e a distribuição das técnicas de teste de aceitação são apresentadas nas Tabelas Tabela 13 e Tabela 12, respectivamente. O objetivo é avaliar se os usuários não-técnicos são capazes de especificar cenários de teste de aceitação para um dado requisito de software. Nas próximas subseções são apresentados os resultados obtidos por técnica de teste de aceitação (grupo do experimento).

5.2.7.4.1 T01 – Fit/Fit tables – Grupo A

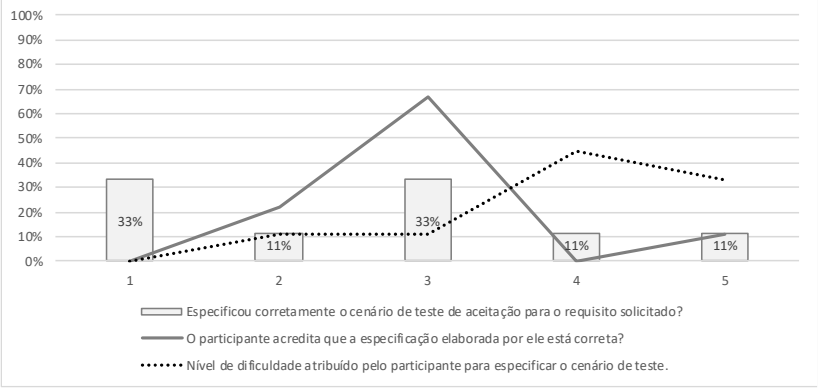
O gráfico ilustrado na Figura 14 apresenta dados relacionados à especificação de cenários de teste de aceitação por usuário não-técnicos utilizando *Fit tables*. Neste gráfico são apresentados os valores para três variáveis: (i) exatidão do cenário de teste de aceitação especificado, (ii) a percepção do participante quanto à exatidão da especificação do cenário de teste de aceitação escrita por ele, e (iii) a opinião do participante sobre o grau de dificuldade para especificar o cenário de teste utilizando *Fit tables*.

A linha pontilhada do gráfico ilustrado na Figura 14 apresenta, na escala de 1-5, a opinião do participante sobre a dificuldade para especificar cenários de teste utilizando *Fit tables*. O valor 1 representa que o participante acredita que é muito fácil especificar cenários de teste utilizando *Fit tables*, enquanto que o valor 5 representa que o participante acredita ser muito difícil escrever estas especificações. Neste mesmo gráfico, as barras representam o percentual de participantes que:

1. Acertaram totalmente: especificaram corretamente um cenário de teste de aceitação utilizando determinada técnica;

2. Acertaram: especificaram corretamente um cenário de teste de aceitação, porém, cometeram erros na notação da técnica que não atrapalham o significado semântico do cenário;
3. Acertaram parcialmente: especificaram o cenário de teste de aceitação de maneira incompleta e não adicionaram nenhuma informação incorreta.
4. Erraram: apesar de ter especificado o cenário correto ou parcialmente correto, os participantes incluíram na sua reposta informações erradas ou controversas.
5. Erraram totalmente: o participante deixou a questão em branco ou respondeu com informações totalmente incorretas.

Figura 14 – Especificação de cenários de teste de aceitação por usuários não-técnico utilizando *Fit tables*.



Fonte: Elaborada pelo autor (2019).

Por fim, a linha contínua, apresentada no gráfico ilustrado na Figura 14, simboliza a percepção do usuário sobre a própria especificação do cenário de teste de aceitação. O usuário atribui a sua especificação uma nota de 1 a 5, onde 1 significa que o participante acredita que o cenário especificado está totalmente correto, e 5 significa que o participante acredita que o cenário especificado está totalmente incorreto.

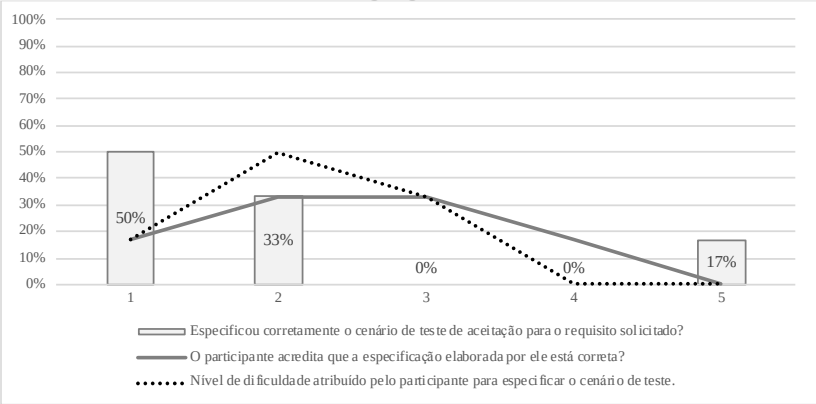
Sendo assim, o gráfico ilustrado na Figura 14 mostra que 44% dos participantes conseguiram especificar corretamente um cenário de teste de aceitação utilizando *Fit tables*, 22% dos participantes *erraram* ou *erraram totalmente* a resposta. Os outros 33 % *acertaram parcialmente*. Quando os participantes foram questionados se eles achavam que a especificação desenvolvida por eles estava correta, 67% deles responderam não estarem certos sobre a exatidão das suas especificações.

Além disto, a maioria dos participantes (78%) acredita que é *difícil* (44%) ou *muito difícil* (33%) especificar os cenários de teste de aceitação utilizando as *Fit tables.*

5.2.7.4.2 T08 – BDD/Gherkin Language – Grupo B

Na Figura 15 é ilustrado o gráfico para análise dos dados respectivos à atividade de especificação do cenário de teste de aceitação por usuários não-técnicos utilizando a *Gherkin language*. Este gráfico possui as mesmas variáveis que o gráfico ilustrado na Figura 14, consequentemente, são aplicadas a elas o mesmo esquema de atribuição de valores e a mesma representação gráfica que foram aplicados para os cenários escritos utilizando *Fit tables*.

Figura 15 – Especificação de cenários de teste de aceitação por usuários não-técnico utilizando BDD/*Gherkin language*.



Fonte: Elaborada pelo autor (2019).

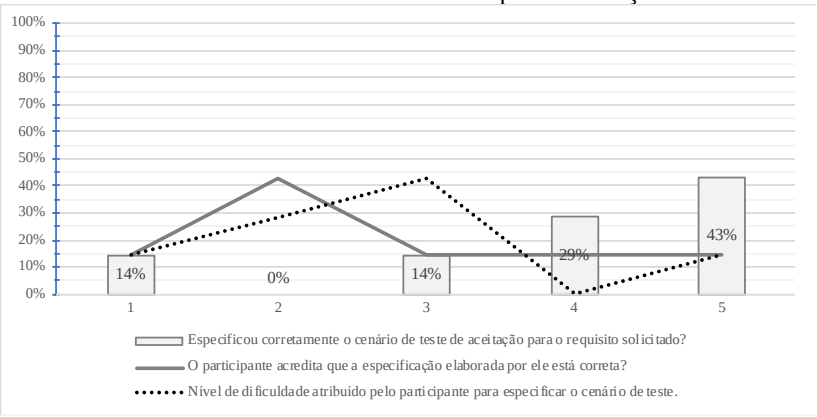
Observa-se neste gráfico, que um número significativo de participantes, 83%, obtiveram sucesso em suas especificações, enquanto somente 17% dos participantes falharam. Inclusive, os mesmos 17% que falharam na especificação, acreditam que o cenário de teste especificado estava incorreto. Além disto, verifica-se também que a maioria dos participantes (67%) considerou *muito fácil* ou *fácil* especificar o cenário de teste para o requisito solicitado utilizando esta técnica.

5.2.7.4.3 T06 – Histórias de usuário com exemplos e marcações – Grupo C

O gráfico ilustrado na Figura 16 apresenta os dados para a atividade de especificação de cenário de teste realizada pelo Grupo C utilizando a técnica Histórias de Usuário com exemplos e marcações. Este gráfico possui as mesmas variáveis que gráfico ilustrado na Figura 14, consequentemente, são aplicadas a elas o mesmo esquema de atribuição de valores e a mesma representação gráfica que foram aplicados para os cenários escritos utilizando *Fit tables*.

Assim como na primeira parte do experimento, onde os resultados obtidos demonstraram que os usuários não-técnicos não conseguiram entender os cenários de teste de aceitação escritos utilizando Histórias de usuário com exemplos e marcações, o mesmo desempenho foi observado na especificação dos cenários, onde a maioria dos usuários, 72%, não conseguiu especificar o cenário de teste de aceitação. Além disto, verifica-se uma avaliação distorcida dos usuários do seu entendimento sobre técnica, pois, 57% dos usuários acreditaram que suas especificações estavam corretas e 43% deles avaliaram a elaboração da especificação dos cenários de teste de aceitação utilizando a técnica como *fácil* ou *muito fácil*.

Figura 16 – Especificação de cenários de teste de aceitação por usuários não-técnico utilizando Histórias de usuário com exemplos e marcações.

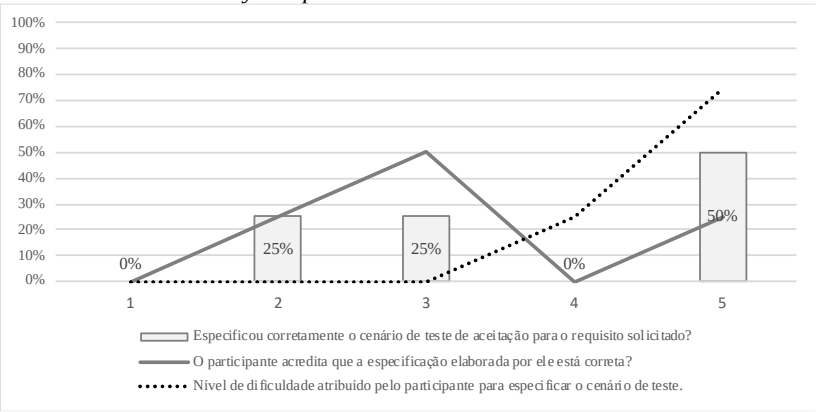


Fonte: Elaborada pelo autor (2019).

5.2.7.4.4 T09 – EasyAccept – Grupo D

Na Figura 17 é ilustrado o gráfico para análise dos dados respectivos à atividade de especificação do cenário de teste de aceitação por usuários não-técnicos utilizado o *EasyAccept*. Este gráfico possui as mesmas variáveis que gráfico ilustrado na Figura 14, consequentemente, são aplicadas a elas o mesmo esquema de atribuição de valores e a mesma representação gráfica que foram aplicados para os cenários escritos utilizando *Fit tables*.

Figura 17 – Especificação de cenários de teste de aceitação por usuários não-técnico utilizando o *EasyAccept*.



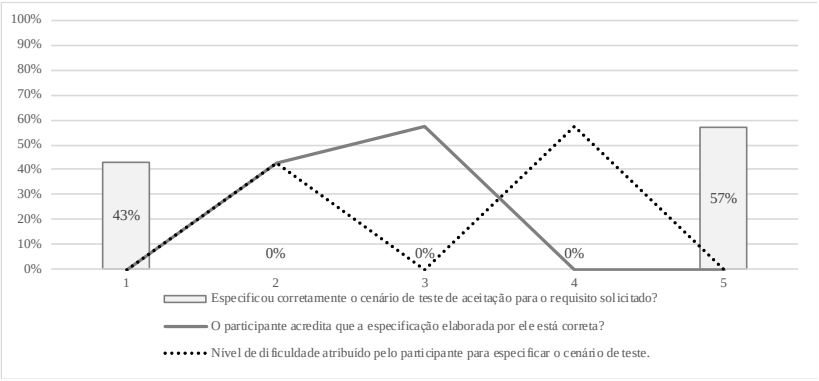
Fonte: Elaborada pelo autor (2019).

Ao contrário da primeira parte do experimento, na qual o *EasyAccept* alcançou bons índices quanto ao entendimento dos cenários de teste de aceitação, na segunda parte do experimento metade dos usuários não conseguiu especificar o cenário de teste utilizando esta técnica. Todos os participantes acharam difícil (25%) ou muito difícil (75%) especificar os testes de aceitação utilizando esta técnica. Além disto, metade dos participantes não conseguiu julgar se o teste especificado estava correto ou não, e a outra metade ficou dividida, 25% acreditam que o teste especificado está correto e os outros 25% acreditam que está errado.

5.2.7.4.5 T11 – US-UID – Grupo E

Os dados obtidos a partir das respostas dos participantes do Grupo E, na segunda parte do experimento, estão apresentados no gráfico ilustrado na Figura 18. Este gráfico possui as mesmas variáveis que gráfico ilustrado na Figura 14, consequentemente, são aplicadas a elas o mesmo esquema de atribuição de valores e a mesma representação gráfica, estes detalhes estão descritos na subseção 5.2.7.4.1 para as *Fit tables*, e também se aplicam para a análise dos cenários especificados pelos participantes utilizando os US-UIDs.

Figura 18 – Especificação de cenários de teste de aceitação por usuários não-técnico utilizando US-UIDs.



Fonte: Elaborada pelo autor (2019).

O percentual de participantes que especificou corretamente os cenários de teste utilizando US-UIDs, 43%, é o mesmo valor dos participantes que julgaram ser fácil especificar os cenários de teste utilizando esta técnica, e também é o mesmo valor dos participantes que acreditam que os cenários de teste especificados estão corretos. Da mesma forma 57% dos participantes especificaram o cenário de teste incorretamente, e este é o mesmo número de participantes que consideram a técnica difícil e que não souberam determinar se o cenário de teste especificado estava correto ou não.

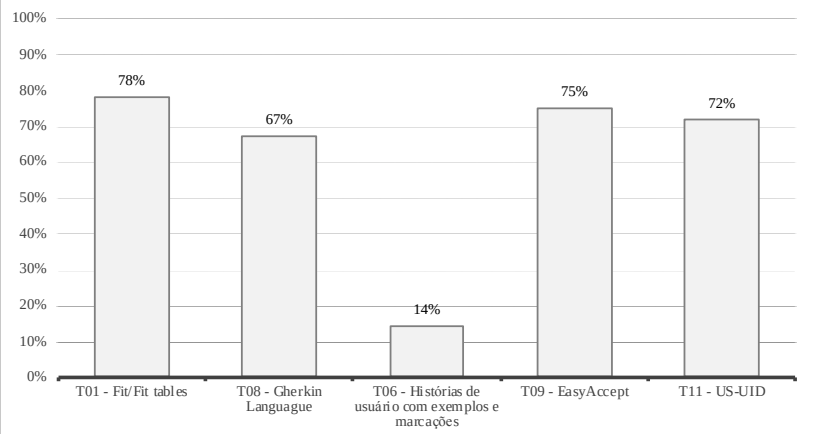
5.2.7.5 Discussão

Nesta subseção são discutidos os resultados obtidos através do experimento 1. O objetivo desta seção é responder, fundamentado nos resultados descritos nas subseções anteriores, as questões de pesquisa.

5.2.7.5.1 RQ1 – Técnicas de teste de aceitação que usuários não-técnicos conseguem entender

Todas as técnicas utilizadas no experimento foram entendidas por pelo menos um usuário não-técnico. Todavia, algumas delas demonstraram-se mais fáceis. A Figura 19 ilustra um gráfico de barras comparativo do percentual de participantes que enumerou corretamente as funcionalidades e regras de negócio especificadas com auxílio dos cenários de teste de aceitação. A comparação entre as técnicas foi realizada utilizando a proporção entre o número de respostas corretas e o número de participantes que responderam ao questionário para cada uma das técnicas. Com exceção da técnica T06 – Histórias de usuário com exemplos e marcações, as demais técnicas apresentaram resultados similares, com percentuais que variam de 67% a 78%.

Figura 19 – Percentual de participantes que identificaram corretamente as funcionalidades e regras de negócio por técnica.



Fonte: Elaborado pelo autor (2019).

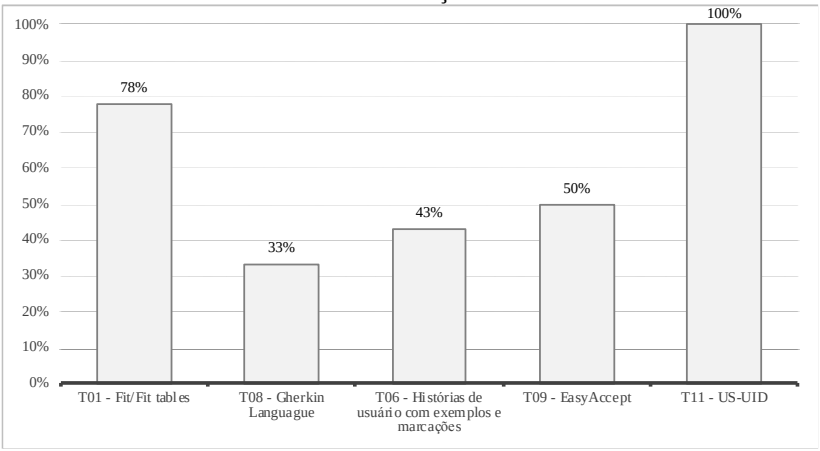
Portanto, apesar do tamanho da amostra não ser suficiente para produzir evidências estatísticas, os resultados obtidos através das medidas

descritivas incorrem na noção de que usuários não-técnicos entendem com mais facilidade os cenários de teste escritos utilizando as técnicas T01, T08, T09 e T11.

5.2.7.5.2 RQ2 – Opinião dos participantes sobre a dificuldade para entender os cenários de teste de aceitação

O gráfico ilustrado na Figura 20 apresenta o percentual de participantes por técnica que considerou *fácil* ou *muito fácil* entender os cenários de teste de aceitação. Tanto a técnica T01 – Fit/*Fit tables*, quanto a técnica T11 – US-UID, foram bem recebidas pelos participantes, que as julgaram fácil de entender. Estas duas técnicas também tiveram bons resultados na análise do percentual de participantes que enumerou corretamente as funcionalidades e regras de negócio especificadas com auxílio dos cenários de teste de aceitação, detalhada na Subseção 5.2.7.5.1.

Figura 20 – Percentual por técnica dos participantes que acharam muito fácil ou fácil entender os cenários de teste de aceitação.



Fonte: Elaborada pelo autor (2019).

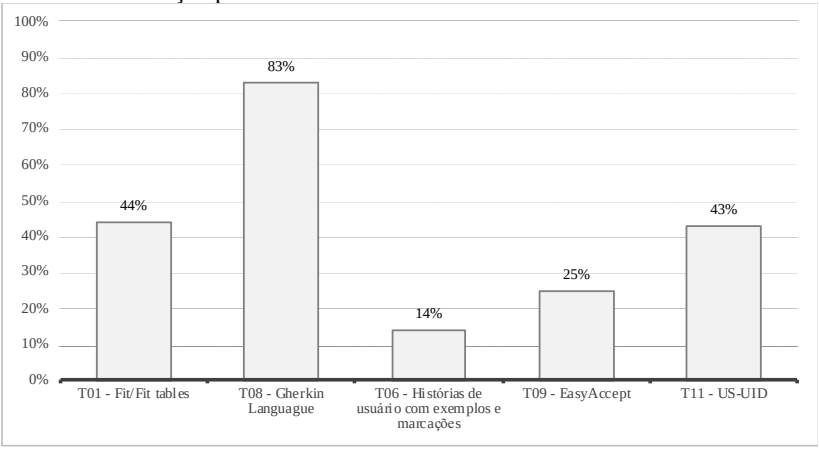
Todavia, esperava-se um resultado diferente para as técnicas T06 e T08. No caso da técnica T06, 43% dos participantes julgaram fácil entender os cenários especificados por esta técnica, porém, somente 14% dos participantes identificaram corretamente as funcionalidades e regras de negocia apresentadas no exemplo disponibilizado no questionário.

Em contrapartida, com a técnica T08 ocorreu o contrário, enquanto que, somente 33% dos participantes julgaram *fácil* ou *muito fácil* entender o cenário de teste de aceitação escrito utilizando esta técnica, 67% dos participantes identificaram corretamente as funcionalidades e regras de negócio apresentadas no exemplo disponibilizado no questionário.

5.2.7.5.3 RQ3 – Técnicas de teste de aceitação que usuários não-técnicos conseguem utilizar para especificar cenários de teste

O gráfico ilustrado na Figura 21 apresenta o percentual de participantes por técnica que conseguiram especificar o cenário de teste de aceitação corretamente. A T08 – BDD/*Gherkin language* foi a técnica que alcançou a maior proporção de especificações totalmente corretas ou corretas dentre as técnicas avaliadas por este estudo, 83% dos participantes obtiveram êxito na especificação. As técnicas T01 – Fit/Fit tables e T11 – US-UID registraram, respectivamente, 44% e 43%.

Figura 21 – Percentual de participantes que especificaram corretamente o cenário de teste de aceitação por técnica.



Fonte: Elaborada pelo autor (2019).

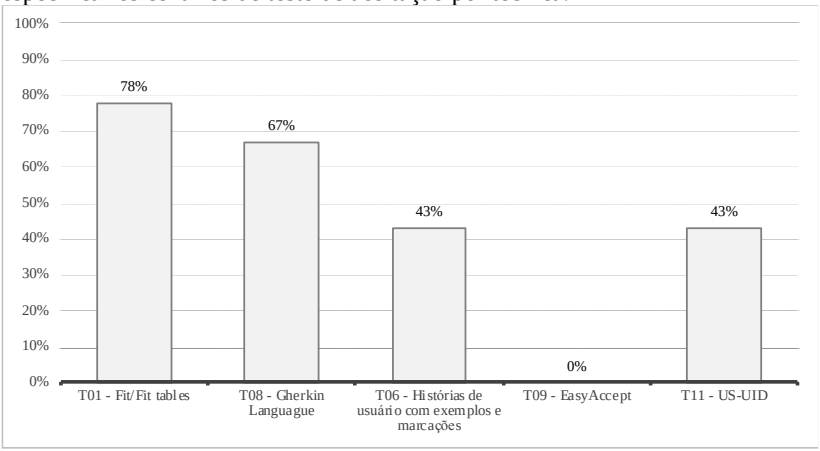
Somente 14% dos participantes que utilizaram a técnica Histórias de Usuário com exemplos e marcações (T06) e 25% dos que utilizaram a técnica *EasyAccept* (T09) conseguiram especificar o cenário de teste de aceitação corretamente. Atribui-se estes resultados as seguintes causas: a técnica T06, como já discutido anteriormente, é uma técnica de representação de cenários de teste de aceitação em Linguagem Natural

(LN), não possui qualquer estrutura definida, nem é auxiliada por uma DSL, logo, os mesmos problemas encontrados ao especificar requisitos utilizando LN são os mesmos incidentes da especificação de cenários de teste de aceitação utilizando esta técnica, por outro lado, a técnica T09 possui sintaxe demasiadamente técnica, similar a uma linguagem de programação, o que torna-se muito complexo para usuários não-técnicos.

5.2.7.5.4 RQ4 – Opinião dos participantes sobre a dificuldade para especificar os cenários de teste de aceitação

A Figura 22 ilustra um gráfico que apresenta o percentual de participantes, por técnica, que consideraram *muito fácil* ou *fácil* especificar o cenário de teste de aceitação. A maioria dos participantes que utilizaram as técnicas T01 – Fit/*Fit tables* e T08 – BDD/*Gherkin language* opinaram ser *muito fácil* ou *fácil* especificar o cenário de teste de aceitação utilizando estas técnicas; T01 acumulou um percentual de 78% e T08 de 67%. Estas duas técnicas também atingiram bons resultados quanto à exatidão das especificações dos cenários de teste produzidos pelos participantes, registrando uma proporção de acertos por grupo de 44% para T01 e 83% para T02, assim como apresentado na Subseção 5.2.7.5.3.

Figura 22 – Percentual dos participantes que acharam muito fácil ou fácil especificar os cenários de teste de aceitação por técnica.



Fonte: Elaborada pelo autor (2019).

Por outro lado, especificar um cenário de teste de aceitação utilizando as técnicas T06 – Histórias de usuário com exemplos e marcações, T09 – *EasyAccept* e T11 – US-UID não foi considerada uma atividade fácil ou muito fácil pela maioria dos participantes dos grupos que as utilizaram. Especificar os cenários utilizando estas técnicas, de acordo com o ilustrado no gráfico da Figura 22, foi considerado muito fácil ou fácil por 43% dos participantes de cada grupo. Comparando o resultado da avaliação da exatidão das especificações com a opinião dos participantes sobre a dificuldade para especificar os cenários, há um forte indício que os dados a respeito da técnica T11 são equivalentes, pois a proporção de participantes que achou fácil é igual a dos que especificou com sucesso. Em contrapartida, a técnica T06 apresentou um comportamento diferente, enquanto que 43% dos participantes considerou muito fácil ou fácil especificar os cenários de teste utilizando esta técnica, somente 14% o fizeram corretamente.

5.2.7.5.5 Técnicas que possuem uma notação simples

Um dos critérios de seleção das técnicas de teste de aceitação para utilização em editais de licitação é que a notação seja simples. Entende-se por notação simples aquela em que usuários não-técnicos conseguem entender os cenários de teste e também conseguem especificar novos cenários utilizando-as. O experimento 1, mostrado nesta seção, avaliou a notação das técnicas pré-selecionadas na seção 5 sob dois aspectos: (i) a opinião dos participantes sobre o nível de dificuldade para entender e especificar cenários de teste utilizando determinada técnica, e (ii) a avaliação das respostas dos participantes, quando lhes foi solicitado explicar o significado das especificações de cenários de teste dadas como exemplo no questionário e quando lhes foi solicitado para especificar um determinado cenário de teste de aceitação. A Tabela 14 apresenta os dados obtidos na avaliação de cada um destes aspectos.

As perguntas de pesquisa RQ2 e RQ4 retratam a opinião dos participantes sobre a dificuldade em compreender e especificar os cenários de teste de aceitação escritos utilizando determinada técnica. Portanto, para compor os valores da coluna "Opinião dos participantes (i)", da Tabela 14, foram somados os percentuais das respostas com valor igual a *muito fácil* ou *fácil* destas questões de pesquisa. Da mesma forma, foram obtidos os valores para a coluna "Avaliação das respostas (ii)", onde foram somados os percentuais das respostas avaliadas como *totalmente corretas* ou *corretas*, de acordo com o apresentado nas respostas das questões de pesquisa RQ1 e RQ3.

Ao final, as técnicas cuja a média aritmética dos valores apresentados nas colunas identificadas por (i) e (ii) é maior que a média total de todas as técnicas (52,5) foram consideradas fáceis de entender e especificar por usuários não técnicos. Estas técnicas estão grifadas em negrito na Tabela 14, e são elas: T01 (Fit/*Fit tables*), T08 (BDD/*Gherkin language*) e T11 (US-UID).

Tabela 14 – Avaliação das técnicas de teste de aceitação quanto a facilidade para entender e especificar cenários de teste.

Técnica	**Opinião dos participantes (i)**		**Avaliação das respostas (ii)**		**Média**
T01 - Fit tables	**78**	**78**	**78**	**44**	**69,5**
T08 - BDD / Gherkin Language	**33**	**67**	**67**	**83**	**62,5**
T06 - Histórias de usuário com exemplos e marcações	43	43	14	14	28,5
T09 - EasyAccept	50	0	75	25	37,5
T11 - US-UID	**100**	**43**	**72**	**43**	**64,5**
				Total	**52,5**

Fonte: Elaborada pelo autor (2019).

5.2.7.5.6 Contribuições espontâneas dos participantes

No questionário foi disponibilizado um campo, não obrigatório, para que os participantes apresentassem comentários sobre a técnica de teste de aceitação ou sobre experiências anteriores com testes de aceitação. Entre os comentários apresentados, pode-se destacar:

- Dos 34 participantes do experimento, cinco relataram ter participado de processos de validação de software nas organizações onde trabalham. Em todos os casos a validação foi realizada manualmente a partir da execução de cenários definidos pelo próprio usuário durante a fase de validação, sem a utilização de qualquer técnica.
- Entre os 5 participantes que tiveram experiência prévia com validação de software, dois deles apontaram que, nos projetos que participaram, durante a fase de validação do software, constatou-se que o produto entregue não atendia as suas necessidades. Ambos registraram que nestes projetos os requisitos foram escritos utilizando linguagem natural e sem a definição de cenários de aceitação. A principal falha apontada por um destes participante é que: “os usuários do sistema

acreditavam estar implícitas no escopo do projeto algumas funcionalidades que são inerentes ao negócio".

- Dos 34 participantes, três acreditam que seria interessante que os usuário não-técnicos tivessem suporte de técnicos para especificar os cenários de teste de aceitação.

5.2.8 Ameaças para a validade

Desde o planejamento do experimento até a sua execução, algumas ameaças para a validade deste experimento foram identificadas. Por exemplo, os requisitos de software utilizados como objeto do experimento não são exemplos provenientes de editais de licitação. Dado que os participantes do experimento são de áreas de conhecimento distintas, optou-se por utilizar como objeto do experimento uma área de negócio comum, pois, caso o domínio do objeto do experimento não fosse de conhecimento de todos os participantes, esta barreira poderia atrapalhar os resultados, cujo objetivo é avaliar a técnica de teste de aceitação. Seria interessante, portanto, repetir o experimento separando a amostra por extratos, e para cada extrato utilizar exemplos reais de licitação que são da área de conhecimento dos participantes.

Outro fato a ser observado é a composição da amostra. Os grupos possuem percentuais distintos de participantes de diferentes áreas de conhecimento, além disto, não há participantes de todas as áreas de conhecimento. Esta ameaça a validade do estudo pode ser mitigada da mesma forma como foi apresentado para a questão do objeto, através da estratificação das amostras.

Adicionalmente, os participantes realizaram o experimento sem qualquer supervisão. Portanto, não há certeza de que os participantes não utilizaram materiais além dos disponibilizados, ou que ainda, não consultaram outros indivíduos da área de tecnologia da informação.

Por último, o tamanho da amostra. Dado que a população são todos os usuários de software sem conhecimento técnico que participam de projetos de desenvolvimento ou melhoria de software, a amostra utilizada neste experimento é muito pequena, o que não permite uma generalização estatística dos resultados, somente uma análise de tendência.

5.3 EXPERIMENTO 2: COMUNICAÇÃO DE REQUISITOS DE SOFTWARE ATRAVÉS DE TESTES DE ACEITAÇÃO

Estudos anteriores, listados na coluna "ESPEC" da Tabela 7, apresentada na seção 3.7.4, confirmaram que é possível a utilização de

cenários de teste de aceitação como ferramenta para comunicação dos requisitos de um software para equipes de desenvolvimento. O objetivo deste experimento é confirmar esta aplicação dos testes de aceitação para as técnicas selecionadas na seção 5.2. A perspectiva é adotar uma destas técnicas para comunicar os requisitos esperados de um software em contratos de terceirização. O objeto deste estudo são os casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

5.3.1 Participantes do experimento

Os participantes deste experimento foram 24 estudantes da disciplina de Programação I do curso de Bacharelado em Sistemas de Informação da Faculdade Estácio de Florianópolis. Os estudantes já fizeram outras disciplinas nas áreas de Programação e Engenharia de Software. Todos eles têm experiência prévia em programação e a maioria deles já trabalha na área de tecnologia da informação. Além disto, os participantes nunca tiveram contato com nenhuma das técnicas de teste de aceitação que foram utilizadas no experimento. Apesar do experimento ter sido conduzido como uma atividade obrigatória da disciplina, os participantes não foram avaliados pelos artefatos produzidos, mas sim pela participação no estudo.

5.3.2 Material utilizado para o experimento

O experimento foi conduzido com base em 2 requisitos: **(SR1)** "Sacar dinheiro de um caixa eletrônico" e **(SR2)** "Calcular o valor das parcelas mensais de um financiamento". Os fluxos para os casos de uso que realizam estes requisitos estão detalhados no APÊNDICE C – Cenários de Casos de Uso.

Os participantes receberam um questionário com 11 perguntas e um documento com a especificação dos requisitos SR1 e SR2. Os estudantes foram divididos em 6 grupos, sendo que, o primeiro grupo – denominado grupo de controle – recebeu a especificação dos fluxos de caso de uso que realizam os requisitos, detalhados no APÊNDICE C – Cenários de Casos de Uso. Os demais grupos, nomeado de A à E, receberam a especificação dos testes de aceitação para os fluxos dos casos de uso detalhados no APÊNDICE C – Cenários de Casos de Uso.

As especificações dos testes de aceitação para os requisitos SR1 e SR2, utilizando cada uma das técnicas de teste de aceitação tratadas neste experimento, estão disponíveis nos apêndices desta dissertação de acordo com a Tabela 15.

Tabela 15 – Especificação dos testes de aceitação por técnica

Técnica	Apêndice
T01 – Fit/*Fit tables*	APÊNDICE E – Exemplo de cenários de teste escritos utilizando Fit Tables
T06 - História de usuário com exemplos e marcações	APÊNDICE I – Exemplos de cenários de teste escritos utilizando Histórias de Usuário com anotações
T08 – BDD/*Gherkin Language*	APÊNDICE J – Exemplos de cenários de teste escritos utilizando *Gherkin Language*
T09 - *EasyAccpet*	APÊNDICE K – Exemplos de cenários de teste escritos utilizando o *EasyAccept*
T11 – US-UID	APÊNDICE L – Exemplos de cenários de teste escritos utilizando US-UID

Fonte: Elaborada pelo autor (2019).

5.3.3 Questões de pesquisa

Considerando as técnicas T01 (Fit/*Fit tables*), T06 (História de usuário com exemplos e marcações), T08 (BDD/*Gherkin Language*), T09 (*EasyAccept*) e T11 (US-UID), este experimento propõe a resolução da seguinte questão de pesquisa:

- RQ1. Quais destas técnicas podem ser utilizadas como artefato para comunicar requisitos funcionais de um software para equipes de desenvolvimento?

5.3.4 Projeto do experimento

Foram utilizados dois objetos e seis tratamentos. Os objetos são os requisitos **(SR1)** "Sacar dinheiro em um caixa eletrônico" e **(SR2)** "Calcular o valor das parcelas mensais de um financiamento". Os fluxos de eventos destes requisitos estão descritos no APÊNDICE C – Cenários de Casos de Uso. Para fins de comparação com práticas para a especificação de requisitos amplamente utilizadas por empresas desenvolvedores de software, também foi incluído neste experimento um grupo de controle. Este grupo recebeu ao invés dos requisitos especificados através das técnicas de teste de aceitação selecionadas na Seção 5.2, a descrição dos fluxos dos casos de uso a serem implementados. Portanto, os tratamentos são:

- **(UseCaseFlows)** Requisitos de software especificados através de fluxos de eventos dos casos de uso;
- **(Fit01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T01 - Fit/*Fit tables*;

- **(UserStories01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T06 - História de usuário com exemplos e marcações;
- **(BDD01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T08 - BDD/*Gherkin Language*;
- **(EasyAccept01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T09 - *EasyAccept*;
- **(US-UID01)** Requisitos de software especificados através de cenários de teste de aceitação escritos utilizando a técnica T11 - US-UID.

Os participantes foram divididos, aleatoriamente, em 6 grupos. O primeiro grupo foi nomeado "Grupo de controle", os demais são identificados das letras de A à E. A cada grupo foi atribuído um tratamento, isto é, como os requisitos esperados do software foram apresentados aos desenvolvedores. A Tabela 16 resume o projeto do experimento.

Tabela 16 – Projeto do experimento 2

Grupo	SR1	SR2	Número de participantes
Grupo de controle	(UseCaseFlows)	(UseCaseFlows)	4
Grupo A	(Fit01)	(Fit01)	4
Grupo B	(BDD01)	(BDD01)	4
Grupo C	(UserStories01)	(UserStories01)	4
Grupo D	(EasyAccept01)	(EasyAccept01)	4
Grupo E	(US-UID01)	(US-UID01)	4

Fonte: Elaborada pelo autor (2018).

5.3.5 Treinamento

Antes da realização do experimento os participantes foram treinados nos seguintes assuntos:

- Uma palestra de 30 minutos sobre teste de aceitação, TDD, ATDD e BDD;
- Uma palestra de 2 horas e 30 minutos englobando os seguintes assuntos: *Fit tables*, *Gherkin Language*, Histórias de usuário com exemplos e marcações, *EasyAccept* e US-UID.

5.3.6 Procedimento adotado

O experimento foi conduzido em duas partes. Na primeira parte, foi introduzido aos participantes o objetivo do experimento e eles foram aleatoriamente separados em 6 grupos. Na sequência, cada membro do grupo respondeu ao questionário. Então, foram realizadas as palestras de treinamento descritas na subseção 5.3.5.

Na segunda parte, foram entregues aos grupos a especificação das funcionalidades do software a ser desenvolvido utilizando uma das técnicas de teste de aceitação selecionadas na seção 5.1 e respeitando a distribuição de acordo com o projeto do experimento, apresentado na Tabela 16. Os grupos ficaram livres para escolher as ferramentas e tecnologias para desenvolver os requisitos solicitados. Os testes não precisavam ser automatizados. No final da segunda parte, os grupos apresentaram as soluções implementadas e elas foram validadas seguindo os mesmos critérios definidos nos testes de aceitação.

5.3.7 Resultados

Nesta subseção são apresentados e discutidos os resultados do experimento 2. Os dados do experimento estão disponíveis em www.leb.inf.ufsc.br/index.php/mestrado/ernani.

5.3.7.1 Perfil dos participantes

Os participantes deste experimento são estudantes do 3º período do curso de bacharelado em Sistemas de Informação. O experimento foi conduzido como uma atividade obrigatório da disciplina de Programação I. Por ser um curso noturno, o público-alvo são estudantes que já trabalham na área de tecnologia da informação e estão buscando atualização e um diploma de curso superior. Sendo assim, 72% destes estudantes são empregados efetivos ou empresários, 16% são estagiários e os outros 12% não trabalham ou trabalham em outras áreas diferentes da de tecnologia da informação. A média de experiência profissional dos estudantes que trabalham na área de tecnologia da informação é de 3,2 anos.

Com exceção das especificações de requisitos através dos cenários de casos de uso utilizadas pelo grupo de controle do experimento, os participantes nunca tinham tido contato com qualquer uma das técnicas de teste de aceitação abordadas por este estudo. Os participantes com mais experiência foram divididos igualitariamente em cada um dos grupos.

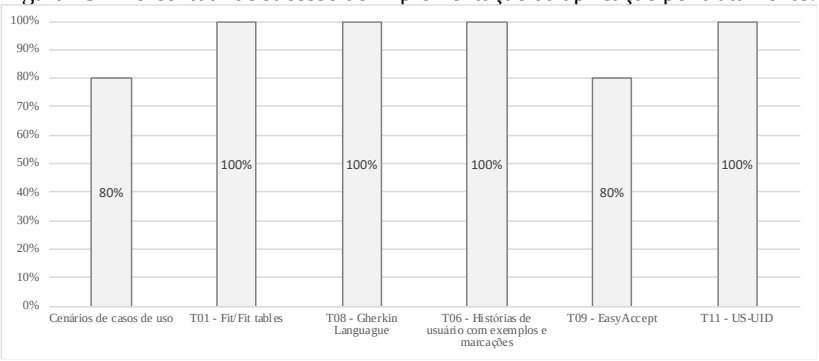
5.3.7.2 Implementação da aplicação

Inicialmente os participantes responderam ao questionário para análise do perfil, e na sequência foram submetidos ao treinamento mandatório sobre as técnicas de teste de aceitação. Foram explicadas cada uma das técnicas e apresentados exemplos.

Então, em um segundo momento, os participantes foram divididos em grupos, sendo que, aqueles com mais experiência foram divididos em igual quantidade entre os grupos, e os demais, foram divididos de forma aleatória. Cada um dos grupos recebeu uma especificação de requisitos utilizado um formato diferente. O formato variava de acordo com o tratamento estabelecido para o grupo do projeto do experimento, apresentado na subseção 5.3.4. Um grupo recebeu os requisitos expressados através de cenários de caso de uso, e foi denominado, grupo de controle. Os demais grupos receberam requisitos especificados através de técnicas de teste de aceitação. Ao total, cada grupo recebeu 5 casos de teste de aceitação para implementação de dois cenários: sacar dinheiro em uma caixa eletrônico e simular as parcelas mensais de um financiamento.

A Figura 23 apresenta os resultados obtidos por cada um dos grupos de acordo com o tratamento utilizado. Os grupos que utilizaram as técnicas T01, T08, T06 e T11, implementaram corretamente 100% dos cenários solicitados utilizando os testes de aceitação como fermenta para obtenção de informação dos requisitos. Já os grupos que utilizaram tanto os cenários de casos de uso quanto os testes de aceitação escritos utilizando o *EasyAccpet* (T09), conseguiram implementar 4 dos 5 cenários solicitados.

Figura 23 – Percentual de sucesso de implementação da aplicação por tratamento.



Fonte: Elaborada pelo autor (2019).

5.3.7.3 RQ1 - Técnicas que podem ser utilizadas como artefato para comunicar requisitos funcionais de um software para equipes de desenvolvimento

Como visto na Subseção 5.3.7.2, todas as formas de representação dos requisitos de software tiveram um bom desempenho. O fato de o grupo de controle não ter implementado com sucesso o software para realização de 100% dos cenários de caso de uso, enquanto que, os grupos que utilizaram as técnicas de teste de aceitação T01, T08, T06 e T11 o fizeram, vem de encontro ao que os estudos anteriores, listados na coluna "ESPEC" da Tabela 7 da Subseção 3.7.4, já haviam destacado: os testes de aceitação auxiliam com sucesso a comunicação dos requisitos de software entre usuários e equipes de desenvolvimento, preenchendo a lacuna entre a especificação de requisitos e o produto entregue. Portanto, pode-se afirmar que este experimento confirmou que as técnicas T01, T08, T06 e T11 podem ser utilizadas como ferramenta para auxiliar a especificação de requisitos funcionais de um software.

5.3.8 Ameaças para a validade

Dado que estudos anteriores já avaliaram positivamente a aplicabilidade dos testes de aceitação como ferramenta para apoiar a especificação dos requisitos funcionais de um software, a expectativa deste experimento é confirmar que todas as técnicas selecionadas podem ser utilizadas com este fim. Os resultados alcançados com este experimento são compatíveis com aqueles alcançados pelos estudos listados na coluna ESPEC da Tabela 7, porém, as seguintes ameaças para a validade do estudo devem ser consideradas:

- Os participantes desconheciam tanto o conceito de testes de aceitação, quanto as técnicas apresentadas: a condução do experimento com um grupo de desenvolvedores com experiência em realização de testes automatizados, bem como, familiaridade com as técnicas abordadas, poderia aprimorar os resultados deste experimento;
- Os requisitos utilizados como objeto do experimento são do domínio dos desenvolvedores, o que pode mascarar falhas na especificação dos requisitos do software. Por outro lado, se os desenvolvedores não tiverem conhecimento da área de negócio na qual está inserida a solução de software em desenvolvimento, os resultados também poderiam ser influenciados pelo

desconhecimento. Portanto, seria interessante trabalhar com desenvolvedores que compartilhem determinado domínio conhecimento, porém com requisitos que ainda não existem ou que os participantes nunca tiveram contato como usuários;

- As validações do código-fonte e da aplicação produzidos pelos participantes foram realizadas pelo professor da disciplina, para deixar o processo de validação dos resultados mais objetivo seria interessantes a participação de um especialista da área de negócio e o emprego de testes automatizados.
- Por fim, o tamanho da amostra analisada é muito pequeno, o que limita a capacidade de testes estatísticos. Além disto, a amostra representa somente um extrato da população, que são desenvolvedores de software. Em um próximo experimento seria interessante o envolvimento de outros papéis, como testadores, analistas de sistema, arquitetos de aplicação e de dados.

5.4 EXPERIMENTO 3: ESFORÇO NECESSÁRIO PARA O DESENVOLVIMENTO DE CÓDIGOS DE COLA

A automatização dos testes de aceitação depende do desenvolvimento de códigos de cola para possibilitar a execução dos cenários especificados. Estes códigos de cola são trechos de código escritos em uma linguagem de programação e estabelecem a conexão dos cenários de teste com o código fonte do software em teste. Espera-se que o desenvolvimento destes códigos e cola não ofereçam alto custo para a organização, desencorajando a realização automatizada dos testes de aceitação.

O objetivo deste experimento é avaliar o esforço, em termos de tempo, para o desenvolvimento do código de cola para um conjunto de testes de aceitação especificados em uma das técnicas selecionadas na seção 5.1. Para tanto, programadores foram convidados a implementar estes códigos para um software e seus respectivos cenários de teste.

A perspectiva é adotar uma destas técnicas para validar requisitos esperados de um software em contratos de terceirização. O objeto deste estudo são os casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

5.4.1 Participantes do experimento

Participaram deste experimento nove desenvolvedores de software do Tribunal de Justiça do Estado de Santa Catarina, um desenvolvedor de software da Universidade Federal de Santa Catarina (UFSC) e um aluno do curso de bacharelado em Ciência da Computação da UFSC.

5.4.2 Material utilizado para o experimento

Um questionário contendo orientações necessárias para a execução do experimento e 17 perguntas divididas em três seções: perfil, compreensão e aplicação. Além disto, o questionário disponibiliza um endereço de internet para a Wiki de um projeto do GitHub. Esta Wiki contém:

- Um guia para obtenção e configuração de um projeto Java, que é o código fonte da aplicação que será testada;
- Orientações sobre instalação das ferramentas necessárias para a execução dos testes de aceitação;
- Procedimento para obtenção dos cenários de teste de aceitação;
- Procedimento para a execução dos testes de aceitação.

5.4.3 Questões de pesquisa

Considerando as técnicas T01 (Fit/*Fit tables*), T06 (História de usuário com exemplos e marcações), T08 (BDD/*Gherkin Language*), T09 (*EasyAccept*) e T11 (US-UID), este experimento propõe a resolução da seguinte questão de pesquisa:

- RQ1. Quanto tempo é necessário em média para codificar o código de cola para um caso de uso utilizando cada uma destas técnicas de teste de aceitação?

5.4.4 Projeto do experimento

Os objetos do experimento são os cenários de teste de aceitação para os fluxos de eventos dos casos de uso “Sacar dinheiro em um caixa eletrônico” e “Calcular parcelas mensais de um financiamento” detalhados no APÊNDICE C – Cenários de Casos de Uso, e sua respectiva implementação. Os tratamentos são:

- **(Fit01)** Cenários de teste de aceitação escritos utilizando a técnica T01 - Fit/*Fit tables*;

- **(UserStories01)** Cenários de teste de aceitação escritos utilizando a técnica T06 - História de usuário com exemplos e marcações;
- **(BDD01)** Cenários de teste de aceitação escritos utilizando a técnica T08 - BDD/*Gherkin Language*;
- **(EasyAccept01)** Cenários de teste de aceitação escritos utilizando a técnica T09 - *EasyAccept*;
- **(US-UID01)** Cenários de teste de aceitação escritos utilizando a técnica T11 - US-UID.

A Tabela 17 resume o projeto do experimento.

Tabela 17 – Projeto do experimento3

Grupo	Software	Número de participantes
Grupo A	(Fit01)	2
Grupo B	(BDD01)	2
Grupo C	(UserStories01)	3
Grupo D	(EasyAccept01)	2
Grupo E	(US-UID01)	2

Fonte: Elaborada pelo autor (2018).

5.4.5 Treinamento

Os participantes não receberam treinamento.

5.4.6 Procedimento adotado

O experimento foi conduzido em ambiente virtual através da troca de mensagens eletrônicas. Inicialmente, foram encaminhados e-mails a possíveis interessados, convidando-os a participar de um experimento para avaliar o esforço, em termos de tempo, para o desenvolvimento de código de cola para execução de testes de aceitação automatizados de um software escrito na linguagem Java. Os interessados deveriam responder ao convite informando um e-mail por onde seriam realizadas as futuras interações. Nesta etapa 11 pessoas demonstraram interesse.

Na sequência, o conjunto de interessados foi dividido em cinco grupos, e para cada grupo foi atribuída uma técnica de teste de aceitação, de acordo com o definido no projeto do experimento, apresentado na subseção 5.4.4. Então, foi encaminhado aos grupos o questionário com os materiais necessários a execução do experimento.

De posse dos materiais necessários, os participantes iniciaram o experimento. Realizaram uma leitura completa do questionário,

responderam as questões a respeito do perfil do participante e compreensão do conceito de testes de aceitação.

Na sequência, os participantes acessaram a Wiki do projeto, instalaram as ferramentas necessárias, obtiveram os códigos fonte do software e os cenários de teste de aceitação. Então os participantes executaram os testes de aceitação e, visto que os códigos de cola não estavam implementados no projeto, visualizaram os erros da execução dos testes.

Em seguida, os desenvolvedores passaram a implementar os códigos de cola e executar os cenários de teste. Este ciclo foi repetido até que todos os cenários fossem testados com sucesso. Por fim, os participantes responderam as demais perguntas do questionário.

5.4.7 Resultados

Nesta subseção são apresentados e discutidos os resultados do experimento 3. Os dados do experimento estão disponíveis em www.leb.inf.ufsc.br/index.php/mestrado/ernani.

5.4.7.1 Perfil dos participantes

Os participantes deste experimento são todos técnicos que, dada as características dos ambientes de trabalho em que atuam, acabam por realizar atividades em todas as fases do desenvolvimento de um software, desde o levantamento e documentação de requisitos, passando pela codificação do software, testes e terminando na implantação e suporte do produto aos usuários finais. A experiência profissional média dos participantes é de 9 anos. Entre os participantes, somente um deles não atua no mercado de trabalho, todavia, é bolsista do Laboratório de Engenharia de Software e Banco de Dados (LEB) da UFSC, e desenvolve atividades de pesquisa que incluem o desenvolvimento de uma ferramenta para a especificação e execução de testes de aceitação.

5.4.7.2 RQ1 - Tempo médio necessário para codificar o código de cola para um cenário de teste de aceitação

Antes de iniciar a codificação dos códigos de cola, os participantes tiveram acesso a um questionário, no qual responderam perguntas sobre o seu perfil e compreensão de testes de aceitação. Além disto, o próprio questionário continha um link de acesso a uma *Wiki* do repositório GitHub criado para cada uma das técnicas de teste de aceitação. Esta *Wiki*

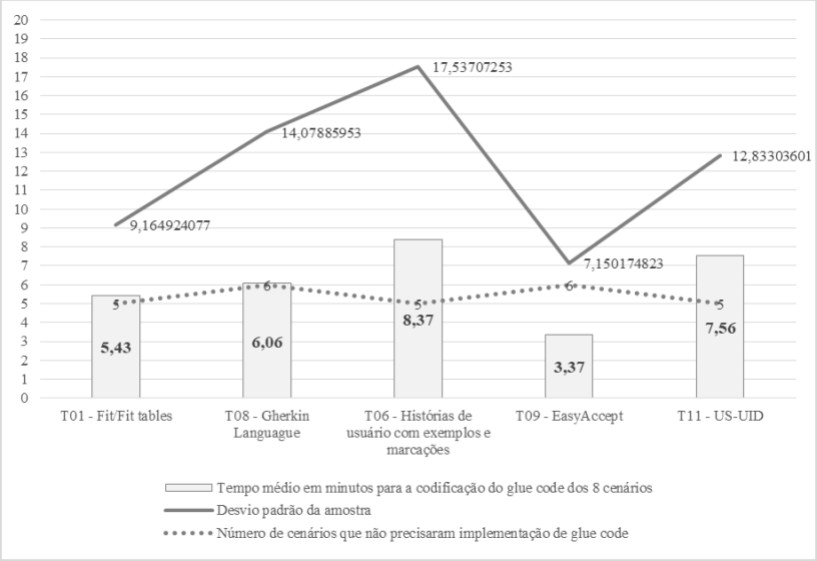
contém orientações para a realização do experimento, informações de como obter o código fonte do software a ser testado e os cenários de testes de aceitação, além disto, mostra como executar os testes e indica materiais que ensinam a implementar os códigos de cola.

Na sequência, após a leitura destes materiais, os participantes implementaram o código de cola. Todos os participantes realizaram todas as atividades solicitadas com sucesso. A Figura 24 ilustra um gráfico que apresenta, através das barras, o tempo médio por cenário de teste para a implementação dos códigos de cola para cada uma das técnicas, através da linha contínua, o desvio padrão da amostra, e, por meio da linha pontilhada, o número de cenários que não precisaram de implementação de código de cola, pois, já foram atendidos por implementações realizadas para outros cenários.

A técnica que exigiu o menor tempo por cenário foi a T09 – *EasyAccept*, com 3,37 minutos. As técnicas T01 – Fit/*Fit tables*, T08 – BDD/*Gherkin language* e T11 – US-UID, apresentaram um tempo médio de 5,43 minutos, 6,06 minutos e 7,56 minutos, por cenário, respectivamente. A técnica que apresentou o maior esforço em termos de tempo para o desenvolvimento dos códigos de cola foi T06 – Histórias de usuário com exemplo e marcações, totalizando uma média de 8,37 minutos por cenário.

Assim como ilustra a linha pontilhada no gráfico ilustrado na Figura 24, todas as técnicas demonstraram um bom nível de reaproveitamento de código de cola. Para as técnicas T01, T06 e T11, dos 8 cenários de teste e aceitação, não foi necessário desenvolver códigos de cola para 5 deles, pois foram atendidos pelos códigos implementados para atender aos outros 3. Para as técnicas T08 e T09, o reaproveitamento foi ainda maior, dos 8 cenários de teste de aceitação, 6 deles não necessitaram de implementação de novos códigos de cola, pois já haviam sido atendidos pelo código implementado para os outros 2 cenários. O reflexo deste reaproveitamento de cenários pode ser visto no desvio padrão amostral, ilustrado pela linha contínua no gráfico, do tempo para a implementação do código de cola, pois a média do tempo de implementação de cada cenário para uma mesma técnica variou muito, por exemplo, enquanto alguns cenários demoraram 20 minutos, outros tantos demoraram 0 minutos para terem o código de cola implementado.

Figura 24 – Tempo médio para implementação dos códigos de cola por técnica.



Fonte: Elaborada pelo autor (2019).

5.4.7.3 Desenvolvimento do código de cola

O esforço necessário para o desenvolvimento do código de cola para os cenários de teste de aceitação pode ser fator limitante da escolha de uma técnica de teste de aceitação ao invés de outra. Isto acontece porque a partir do momento que o código de cola é deixado de ser implementado e os testes de aceitação passam a não ser executados automaticamente, eles tornam-se desatualizados e obsoletos.

Todas as técnicas de teste de aceitação avaliadas neste estudo apresentaram uma boa reutilização de *steps* dos testes de aceitação. O resultado é que para um conjunto de 8 cenários de teste, o desenvolvimento de código de cola para 2 ou 3 cenários foi suficiente para atender a todos os demais.

O esforço médio em minutos necessário para o desenvolvimento do código de cola para cada um dos 8 cenários utilizando as técnicas de teste de aceitação selecionadas na Seção 5.1 é de 6,15 minutos. Sendo assim, há um forte indício que as técnicas que estão abaixo deste tempo médio são aquelas que apresentam menor esforço, e são elas: T09 – *EasyAccept*, com esforço médio 3,37 minutos por cenário, T01 – Fit/*Fit*

tables, com esforço médio de 5,43 minutos por cenário e T08 – BDD/*Gherkin language* com esforço médio de 6,06 minutos por cenário.

5.4.8 Ameaças para a validade

Desde o planejamento do experimento até a sua execução, algumas vulnerabilidades foram encontradas. O não conhecimento do software em teste, bem como das técnicas de teste de aceitação, influenciam negativamente os resultados, pois, menos tempo poderia ter sido investido no desenvolvimento dos códigos de cola caso os desenvolvedores tivessem experiência anterior com as técnicas de aceitação e com o software em teste. Este problema poderia ser reduzido através de treinamento prático das técnicas de teste de aceitação e também apresentação da arquitetura e principais funcionalidades/métodos do software em teste.

O domínio da aplicação era comum a todos, porém, os requisitos não são aqueles encontrados em editais de licitação. Caso outros requisitos fossem empregados, com grau de complexidade maior, os resultados referentes ao tempo necessário para codificar os códigos de cola poderia ter sido impactado. A repetição deste experimento com requisitos mais complexos poderia avaliar este impacto.

Cada técnica utiliza diferentes ferramentas, e cada uma destas ferramentas funcionalidades próprias. Algumas destas ferramentas possuem funcionalidades que contribuem com o desenvolvimento dos códigos de cola, como por exemplo, a geração de um modelo de código. Estas ferramentas aumentam a produtividade do desenvolvimento destes códigos, consequentemente, desenvolver códigos de cola para as técnicas apoiadas por estas ferramentas torna-se mais eficiente do que aquelas que exigem esforço 100% manual. Para dirimir esta ameaça, buscou-se utilizar as ferramentas mais apontadas na revisão sistemática que suportam a geração de código de cola. Porém, o nível de geração do código de uma ferramenta para a outra não é o mesmo.

Por fim, o tamanho da amostra é muito pequeno para a generalização dos resultados através de testes estatísticos, é necessário repetir este experimento com mais indivíduos. Além disto, a utilização de requisitos mais complexos também contribuiria para melhor avaliação das técnicas selecionadas.

5.5 SELEÇÃO DAS TÉCNICAS DE TESTE DE ACEITAÇÃO

Os experimentos conduzidos neste capítulo permitem elaborar uma análise das técnicas de teste de aceitação selecionadas na Seção 5.1 para avaliar quais delas atendem aos critérios de seleção estabelecidos na Seção 4.1 que não puderam ser respondidos através da RSL apresentada no Capítulo 3.

A Tabela 18 apresenta grifadas em negrito as técnicas de teste de aceitação que atenderam a todos os critérios técnicos e legais, definidos respectivamente nas subseções 4.1.1 e 4.1.2. Para avaliar se uma técnica atende ou não a um critério foi realizada uma avaliação com base nos dados obtidos através da RSL apresentada no Capítulo 3 e dos experimentos planejados nas seções 5.2, 5.3 e 5.4. Na Tabela 18 as colunas sombreadas correspondem aos critérios que foram avaliados com base nos resultados dos experimentos, as demais colunas referem-se aos critérios já avaliados anteriormente com base na RSL.

O critério técnico 1 e o critério legal 2 foram avaliados de acordo com o exposto na subseção 5.2.7.5.5, onde as técnicas que obtiveram médias maior que 60 foram consideradas técnicas fáceis de entender e especificar, assim como ilustra a Tabela 14. Porém, para o critério legal 2 foi retirada a técnica T11 (US-UID), pois, a documentação disponível das ferramentas não é suficiente para a utilização da técnica, o que restringe seu acesso ao público em geral. A avaliação para o critério técnico 3 está detalhada na Seção 5.3.7.3 e para o critério técnico 7 na Seção 5.4.7.3.

Tabela 18 – Seleção das técnicas de teste de aceitação para aplicação em um exemplo de edital.

#	Técnica	Critérios técnicos							Critérios legais			
		1	2	3	4	5	6	7	1	2	3	4
T01	**Fit/*Fit tables***	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
T06	História de usuário com exemplos e marcações	-	✓	✓	✓	✓	✓	-	✓	-	✓	✓
T08	**BDD/*Gherkin Language***	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
T09	*EasyAccept*	-	✓	✓	✓	✓	✓	✓	✓	-	✓	✓
T11	US-UID	✓	✓	✓	✓	✓	✓	-	✓	-	✓	✓

Fonte: Elaborada pelo autor (2019).

6 AVALIAÇÃO DAS TÉCNICAS DE TESTES DE ACEITAÇÃO SELECIONADAS

Nesta seção é apresentada a definição, o projeto, o planejamento e os resultados de um experimento, seguindo as orientações propostas por Wholin, Runeson, et al., (2012) e Juristo e Moreno (2001). O objetivo deste experimento é comparar a aplicabilidade das *Fit Tables* e da *Gherkin language* como artefato para auxiliar a especificação de requisitos em editais de licitação. A perspectiva deste experimento é adotar uma destas técnicas para especificar requisitos de software tanto em editais de licitação quanto em contratos de terceirização de desenvolvimento de software. Os participantes deste experimento são estudantes do último ano do curso de graduação em Ciência da Computação e o objeto é um conjunto de requisitos de um software para a gestão de pessoas. Este software provê funcionalidades tais quais registros funcionais e processamento das folhas de pagamento dos empregados de uma organização pública.

6.1 PARTICIPANTES DO EXPERIMENTO

Participaram do experimento 18 estudantes da disciplina de Tópicos Especiais em Aplicações Tecnológicas I. Os participantes estão no último ano do curso de bacharelado em Ciência da Computação da Universidade Federal de Santa Catarina, já cursaram as disciplinas de Programação e Engenharia de Software, têm conhecimento médio a avançado em programação, e, antes do experimento, nunca tiveram contato com as técnicas *Fit tables* e *Gherkin language*. A maior parte dos participantes são estagiários de empresas que trabalham com desenvolvimento de software.

Embora o experimento tenha sido conduzido como uma atividade obrigatória da disciplina, os participantes não foram avaliados pelos artefatos produzidos, mas, pelo envolvimento no experimento. Além disto, os participantes foram avisados que a atividade se tratava de um experimento para comparar a aplicação de duas técnicas de teste de aceitação como artefato para auxiliar a comunicação de requisitos em editais de licitação.

6.2 MATERIAL UTILIZADO PARA O EXPERIMENTO

O material foi disponibilizado em um repositório na internet. Foi dada permissão a cada um dos participantes para acessar este repositório,

que continha: a descrição de um software para gestão de pessoas, instruções para configurar o ambiente de desenvolvimento para realizar novas implementações neste software, uma planilha eletrônica e um questionário.

O objeto do experimento é um conjunto de quatro requisitos do software de gestão de pessoas. A planilha eletrônica foi utilizada pelos participantes para anotar o tempo gasto em cada uma das tarefas do experimento. O questionário é um conjunto de 24 perguntas para investigar o perfil dos participantes e realizar uma análise qualitativa das tarefas realizadas por eles. Este questionário pode ser visualizado no APÊNDICE O – Questionário do Experimento 4

As ferramentas escolhidas serem utilizadas neste experimento foram aquelas mais referenciadas pelos estudos. O ambiente de desenvolvimento foi configurado pelos próprios participantes que receberam um tutorial orientando passo-a-passo esta atividade. O conteúdo do tutorial contém orientações para:

- Instalar e configurar o *Java Enterprise Edition*;
- Instalar e configurar uma versão *standalone* do *FitNesse* e suas dependências. O *FitNesse* é uma ferramenta que automatiza a execução dos testes de aceitação escritos através das *Fit tables;*
- Instalar e configurar o *Eclipse,* ambiente integrado de desenvolvimento Java;
- Instalar e configurar o *plug-in* do *Cucumber* no Eclipse. O *Cucumber* é uma ferramenta para especificar e automatizar a execução dos testes de aceitação escritos utilizando a *Gherkin language*.
- Um exemplo do tipo *quick start* para validar que o ambiente de desenvolvimento está corretamente configurado.

6.3 QUESTÕES DE PESQUISA

Considerando que as *Fit tables* e a *Gherkin language* são técnicas de teste de aceitação disponíveis e que atendem aos critérios necessários para sua aplicação como ferramenta para auxiliar na especificação de requisitos de software em editais de licitação, este experimento busca responder as seguintes questões de pesquisa:

- **RQ1.** Qual destas duas técnicas é mais fácil de aprender?
- **RQ2.** Qual destas duas técnicas requer menos esforço (em termos de tempo) para especificar cenários de teste de aceitação?

- **RQ3.** Qual destas duas técnicas é a melhor para comunicar requisitos de um software?

Uma vez formuladas as questões de pesquisa, é possível transformá-las em hipóteses nulas a serem testadas no experimento:

- **H_{0a}** A exatidão dos cenários de teste de aceitação especificados pelos participantes que compareceram a uma palestra de três horas sobre *Fit tables* e *Gherkin language* é a mesma para as duas técnicas de teste de aceitação.
- **H_{0b}** O esforço para especificar cenários de teste de aceitação é o mesmo para ambas as técnicas.
- **H_{0c}** A exatidão das funcionalidades do software implementadas pelos participantes com base nas especificações expressadas tanto por *Fit tables* quanto por *Gherkin language* é a mesma para as duas técnicas de teste de aceitação.

Por outro lado, as hipóteses alternativas são:

- **H_{1a}** A exatidão dos cenários de teste de aceitação especificados pelos participantes que compareceram a uma palestra de três horas sobre *Fit tables* e *Gherkin language* é diferente de uma técnica de teste de aceitação para outra.
- **H_{1b}** O esforço para especificar cenários de teste de aceitação é diferente de uma técnica para outra.
- **H_{1c}** A exatidão das funcionalidades do software implementadas pelos participantes com base nas especificações expressadas tanto por *Fit tables* quanto por *Gherkin language* é diferente de uma técnica de teste de aceitação para outra.

As variáveis dependentes deste experimento são:

- **ATS#** – O teste de aceitação para o requisito # foi especificado: **{corretamente, incorretamente**};
- **ATST#** – Os participantes precisam de **{?}** minutos para especificar os cenários de teste de aceitação do requisito #;
- **AR#** – O código-fonte entregue, implementado com base nos cenários de teste de aceitação do requisito # é executável e foi aceito pelo Analista de Negócio: **{sim, não};**

Na definição das variáveis dependentes, o símbolo "#" representa um dos requisitos de software identificado de SR1 a SR4 na Tabela 19, os valores entre chaves "{}" são os possíveis valores atribuídos para a variável dependente, e o símbolo "?" representa um valor inteiro.

O número de testes de aceitação que foram especificados corretamente (ATS#) é obtido a partir da avaliação dos artefatos entregues pelos participantes. O tempo gasto para especificar os cenários de teste

para cada requisito (ATST#) é obtido a partir dos dados anotados pelos participantes em uma planilha eletrônica entregue ao final do experimento. O número de requisitos corretamente implementado (AR#) é obtido a partir da execução e avaliação do código fonte implementado pelos participantes. A avaliação do código fonte foi realizada pelo autor desta dissertação que exerceu o papel de analista de negócio. Para realizar a avaliação foram executados os softwares entregues pelos participantes, e o analista de negócios decidiu se aceitava ou não o produto entregue. Se o analista de negócio aceitar a funcionalidade entregue, o requisito codificado é considerado correto. Caso contrário, é considerado incorreto.

Tabela 19 – Requisitos a serem implementados no sistema de gestão de pessoas

#	**Requisito do software**
SR1	O sistema deve permitir retificar informações da ficha funcional dos colaboradores.
SR2	O sistema deve calcular os valores a serem pagos aos colaboradores a título de Vantagem Pessoal Nominalmente Identificada (VPNI).
SR3	O sistema deve permitir excluir registros da ficha funcional dos colaboradores.
SR4	O sistema deve calcular a média mensal dos valores a serem pagos aos colaboradores a título de Vantagem Pessoal Nominalmente Identificada (VPNI) por cargo.

Fonte: Elaborada pelo autor (2018).

6.4 PROJETO DO EXPERIMENTO

O experimento foi dividido em duas partes. Na primeira parte os participantes especificaram os cenários de teste de aceitação, enquanto na segunda parte os participantes implementaram novos requisitos no software de gestão de pessoas usando como fonte de informação os requisitos funcionais especificados com o auxílio de testes de aceitação.

Nas duas partes do experimento, foram utilizados quatro objetos e dois tratamentos. Os objetos são os novos requisitos a serem implementados no software para gestão de pessoas, de acordo com a Tabela 19. Os tratamentos são:

(F) Requisitos de software especificados com o auxílio de cenários de teste de aceitação escritos através de *Fit tables*.

(G) Requisitos de software especificados com o auxílio de cenários de teste de aceitação escritos utilizando *Gherkin language*.

Os participantes foram divididos em dois grupos, identificados pelas letras A e B. Na parte 1 do experimento, o grupo A especificou os cenários de teste de aceitação para dois dos requisitos que compõem o objeto do experimento utilizando *Fit tables*, enquanto o grupo B especificou os cenários de teste de aceitação para outros dois requisitos que compõem o objeto do experimento utilizando *Gherkin language*. A Tabela 20 apresenta estes grupos e quais tratamentos foram utilizados por

eles para especificar cada um dos requisitos de software. No cabeçalho desta tabela, SR1, SR2, SR3 e SR4 são abreviações para os objetos listados na Tabela 19.

Na segunda parte do experimento, os dois requisitos especificados pelo grupo A, com o auxílio dos cenários de teste de aceitação, foram enviados aos participantes do grupo B, e *vice-versa*. Então, como mostra a Tabela 21, o grupo A codificou no software de gestão de pessoas os requisitos SR3 e SR4, estes requisitos foram especificados pelo grupo B, utilizando *Gherkin language*, na primeira parte deste experimento. Por sua vez, o grupo B codificou no software de gestão de pessoas os requisitos SR1 e SR2, estes requisitos foram especificados pelo grupo A, utilizando *Fit tables*, na primeira parte deste experimento. Esta troca de especificações entre os participantes dos grupos permitiu simular um cenário mais próximo ao real.

Tabela 20 – Projeto de experimentos da Parte 1 – Especificação de cenários de teste de aceitação.

Participantes	**Objetos e tratamentos**			
	SR1	***SR2***	***SR3***	***SR4***
Grupo A	(F)	(F)	-	-
Grupo B	-	-	(G)	(G)

Fonte: Elaborada pelo autor (2018).

Todavia, antes de realizar a troca dos cenários entre os participantes dos grupos, foi verificada a exatidão e consistência de cada um dos cenários produzidos pelos participantes. Cenários de teste de aceitação que apresentavam problemas de especificação, eram substituídos por cenários corretos para os mesmos requisitos. Esta intervenção foi necessária para prevenir a geração de falsos negativos durante o processo da validação de software entregue pelas equipes, pois é com base nesta validação que será verificado qual das técnicas de teste de aceitação comunica melhor os requisitos de um software.

Tabela 21 – Projeto da segunda parte do experimento: implementação dos novos requisitos.

Participantes	**Objetos e tratamentos**			
	SR1	***SR2***	***SR3***	***SR4***
Grupo A	-	-	(G)	(G)
Grupo B	(F)	(F)	-	-

Fonte: Elaborado pelo autor (2018).

6.5 TREINAMENTO

Os participantes foram treinados nos seguintes assuntos:

- Uma palestra de meia hora sobre testes de aceitação, TDD, ATDD e BDD;
- Uma palestra de uma hora e meia sobre *Fit tables* e *FitNesse*, esta palestra abrangeu os conceitos de *Fit tables*, a configuração do *FitNesse* (*framework* para especificação e execução dos testes de aceitação), e o desenvolvimento de exercícios como atividades práticas;
- Uma palestra de uma hora e meia sobre a *Gherkin language* e o *Cucumber*, nesta palestra foram abordados os conceitos e sintaxe da *Gherkin language*, além de trabalhar a configuração do *plug-in Cucumber* para o Eclipse e a resolução de exercícios práticos para fixação.

6.6 PROCEDIMENTO

O experimento foi conduzido de acordo com o descrito nesta subseção. Inicialmente, foi proferido aos participantes uma breve introdução da definição e dos objetivos deste experimento, em seguida, os participantes foram divididos de forma aleatória em dois grupos e, então, receberam o material descrito na Subseção 0.

Em seguida, as seguintes etapas foram executadas:

(1) Os participantes tiveram 20 minutos para verificar se o ambiente de software necessário para realizar a especificação dos testes de aceitação estava funcionando corretamente. Este ambiente foi previamente configurado durante a sessão de treinamento. Nesta etapa, os participantes também responderam as seis primeiras questões do questionário.

(2) Os participantes leram um documento que apresenta uma visão geral do software para gestão de pessoas, na sequência, receberam um outro documento com a descrição em linguagem natural de dois novos requisitos a serem especificados com o auxílio dos cenários de teste de aceitação.

(3) Para cada requisito, os participantes:
(3.a) preencheram a hora de início desta atividade em suas planilhas eletrônicas;

(3.b) tiveram que entender o requisito descritos em linguagem natural, e se tivessem qualquer dúvida poderiam consultar o analista de negócio disponível;
(3.c) tiveram que especificar o requisito utilizando a técnica de teste de aceitação atribuída a eles;
(3.d) e quando terminado, os participantes tinham que marcar o horário de parada na folha de tempo.

(4) Em seguida, os participantes tiveram que responder as próximas oito perguntas do questionário e enviar as especificações dos cenários de teste de aceitação produzidos para um repositório da web. Os artefatos foram identificados por um ID numérico aleatório e apenas os pesquisadores sabiam quem os enviou.
(5) Um especialista em testes de aceitação avaliou todos os artefatos carregados e marcou os que eram inconsistentes ou incompletos. Essa marca era visível apenas para os pesquisadores.
(6) No sexto passo, a segunda parte do nosso experimento foi iniciada. Os participantes tiveram 20 minutos para verificar se o ambiente para desenvolver as próximas tarefas estava funcionando. Depois disso, eles tiveram que baixar as especificações dos cenários de teste de aceitação produzidos por um participante do outro grupo.
(7) Em seguida, os participantes tiveram que responder duas perguntas no questionário relacionadas à sua visão sobre os artefatos baixados.
(8) Os pesquisadores tiveram que verificar quais participantes baixaram os testes de aceitação que, de acordo com a avaliação do especialista, estavam incorretos. Em seguida, eles trocaram os testes incorretos de aceitação por testes corretos.
(9) Para cada cenário de teste de aceitação (requisito), os participantes:
(9.a) tiveram que preencher o horário de início em sua planilha eletrônica;
(9.b) tiveram que entender por si mesmos o teste de aceitação;
(9.c) tiveram que desenvolver o novo requisito para o software de gestão de pessoas expresso através dos cenários de teste de aceitação;
(9.d) quando concluída esta atividade, deveriam anotar o tempo de parada em sua planilha eletrônica.
(10) Por fim, os participantes tiveram que responder as próximas oito perguntas do questionário e enviar o código fonte produzido para um repositório web. Os artefatos foram identificados por um ID

numérico aleatório, e apenas os pesquisadores sabiam quem os enviou.

Este procedimento foi realizado em duas sessões. A primeira com três horas e meia de duração e a segunda com duas horas de duração.

6.7 RESULTADOS E ANÁLISE DOS DADOS

Nesta seção são apresentados e discutidos os resultados obtidos no experimento. Os dados e gráficos do experimento estão disponíveis em www.leb.inf.ufsc.br/index.php/xp2018/.

6.7.1 Consistência e exatidão dos cenários de teste de aceitação

A Tabela 22 é a tabela de contingência para as variáveis dependentes ATS*SR1*, ATS*SR2*, ATS*SR3* e ATS*SR4*, estas variáveis estão detalhadas no projeto do experimento na Subseção 6.4. A primeira linha desta tabela mostra o número de especificações elaboradas pelos participantes utilizando as *Fit tables* como técnica de teste de aceitação: 17 tarefas foram concluídas com sucesso e apenas 1 tarefa falhou. A segunda linha mostra o número de especificações elaboradas pelos participantes utilizando *Gherkin language*: 13 tarefas foram concluídas com sucesso e 5 tarefas falharam. Cenários de teste para requisitos distintos especificados pelo mesmo participante foram considerados como medidas independentes.

Tabela 22 – Tabela de contingência para especificação dos cenários de teste de aceitação.

	Os cenários de teste de aceitação foram especificados:	
Tratamento	**Corretamente**	**Incorretamente**
Fit Tables (T)	17	1
Gherkin Language (G)	13	5

Fonte: Elaborada pelo autor (2019).

Foi aplicado o *Fisher's exact test* nos dados apresentados na Tabela 22. Este teste retornou um valor-p de 0,1774. Portanto, a hipótese nula é aceita, a associação entre linhas (tratamentos) e colunas (resultados) é considerado não estatisticamente significativo.

Embora não se possa obter uma resposta para RQ1 através do *Fisher's exact test*, acredita-se que sessões de treinamento extras e mais exercícios práticos possam diminuir o número de especificações incorretas, pois, os erros identificados nos cenários de teste especificados pelos participantes em ambas as técnicas são erros básicos.

Adicionalmente, combinando estes resultados com os resultados do experimento apresentado na seção 5.2 o número de especificação corretas e incorretas de uma técnica para outra tende-se a se igualar. Portanto, há um forte indício que a complexidade para aprender as duas técnicas de teste de aceitação por desenvolvedores de software e por usuários não-técnicos é a mesma, já que nos dois experimentos os participantes receberam as mesmas palestras e produziram cenários de teste de aceitação com um nível similar de qualidade utilizando ambas as técnicas.

6.7.2 Esforço, em termos de tempo, para especificar os cenários de teste de aceitação

A Tabela 23 apresenta o tempo, em minutos, gasto pelos participantes para especificar os cenários de teste de aceitação para cada um dos requisitos. Os valores sublinhados referem-se aos cenários que foram especificados incorretamente. Cenários de teste para requisitos distintos especificados pelo mesmo participante foram considerados como medidas independentes.

Tabela 23 – Tempo gasto para especificar os testes de aceitação para os novos requisitos do software para gestão de pessoas.

Tratamento	Tempo gasto para especificar os cenários de teste de aceitação em minutos
Fit tables (F)	{20, 21, 21, 23, 35, 36, 40, 45, 50, 65, 66, 77, 79, 88, 108, 120, 126, 135}
Gherkin Language (G)	{15, 16, 17, 15, 39, 30, 30, 30, 45, 35, 60, 28, 40, 40, 70, 57, 84, 75}

Fonte: Elaborada pelo autor (2019).

Foi realizado o teste de normalidade de Shapiro-Wilk para verificar se os dados coletados do experimento para os dois tratamentos têm uma distribuição normal. Em seguida, realizou-se dois testes t, o primeiro considerando todos os dados, que retornou um valor p de 0,0291, e o segundo excluindo os dados sublinhados e obtivemos um valor p de 0,0334. Para um intervalo de 95% de confiança, a hipótese nula é rejeitada. Portanto, há uma diferença, em termos do tempo médio gasto para especificar testes de aceitação entre as *Fit tables* e a *Gherkin language*.

Respondendo a RQ2, o esforço, em termos de tempo médio, para especificar cenários de teste de aceitação utilizando *Gherkin language* (40 minutos) é menor do que usando *Fit tables* (64 minutos).

6.7.3 Aplicabilidade dos cenários de teste de aceitação como artefato para auxiliar na especificação de requisitos funcionais de um software

A Tabela 24 é a tabela de contingência para as variáveis dependentes ATS*SR1*, ATS*SR2*, ATS*SR3* e ATS*SR4*, estas variáveis estão detalhadas no projeto do experimento na subseção 6.4. A primeira linha desta tabela mostra o número de tarefas implementadas com base nos requisitos funcionais especificados com auxílio das *Fit tables*: 13 tarefas foram concluídas com êxito e 5 tarefas falharam. A segunda linha mostra a mesma informação para as tarefas implementadas com base nos requisitos funcionais especificados com auxílio da *Gherkin language*: 10 tarefas foram concluídas com sucesso e 8 falharam.

Aplicou-se o *Fisher's exact test* nos dados apresentados na Tabela 24. Este teste retornou um valor p de 0,4887. Portanto, a hipótese nula é aceita, não há evidências estatísticas significantes de que um dos tratamentos aplicados na especificação dos requisitos do software tenha tido melhor resultado do que o outro.

Então, respondendo a RQ3, não se pode supor, com base no resultado do *Fisher's exact test*, que uma técnica é melhor que outra para comunicar requisitos. Além disso, acredita-se que da mesma forma que um treinamento mais efetivo poderia melhorar as especificações produzidas pelos participantes, o mesmo efeito positivo também poderia refletir na capacidade dos participantes em entender melhor os requisitos especificados com o auxílio dos cenários de teste de aceitação.

Adicionalmente, observa-se que nos resultados do experimento 2, apresentado na Seção 5.3.7, tanto as *Fit tables* quanto a *Gherkin language* obtiveram 100% de êxito na comunicação dos requisitos. Todavia, naquele experimento o domínio dos requisitos era comum a todos os participantes – um software para caixa eletrônico –, enquanto que, neste experimento, o domínio era desconhecido pelos participantes.

Tabela 24 – Tabela de contingência para o desenvolvimento dos requisitos comunicados através dos cenários de teste de aceitação.

	O código-fonte entregue implementado com base nos cenários de teste de aceitação é executável e foi aceito pelo analista de negócios:	
Tratamento	**Sim**	**Não**
Fit Tables (T)	13	5
Gherkin language (G)	10	8

Fonte: Elaborada pelo autor (2018).

Por outro lado, apesar das evidências apresentadas nos dois experimentos demonstrarem que as técnicas são equivalentes para auxiliar na especificação dos requisitos funcionais de um software, acredita-se que, para um melhor aproveitamento das *Fit tables*, é necessário adicionar às tabelas uma pequena descrição textual, apresentando uma visão geral do requisito funcional a ser implementado e o objetivo do cenário de teste de aceitação. Estudos anteriores já mostraram a tendência destas tabelas serem "fracas em detalhes" sobre o software (MELNIK, MAURER e CHIASSON, 2006), adicionalmente, observa-se que os participantes do experimento, mesmo sem qualquer orientação sobre o assunto, também sentiram esta necessidade, pois a maioria das especificações escritas por eles, utilizando esta técnica, continham um pequeno parágrafo explicativo sobre o requisito desejado e o objetivo do cenário de teste de aceitação.

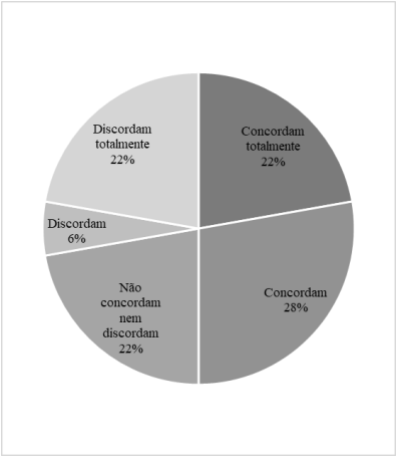
6.7.4 Resultados do questionário

Nesta seção, são discutidas quatro questões do questionário aplicado durante o experimento. As questões consistem em afirmações as quais os participantes deveriam atribuir uma nota de 1 a 5, de acordo com o seu entendimento, onde 1 significa que o participante *concorda totalmente* com a afirmação, 2 que o participante *concorda*, 3 que o participante *não está certo se concorda ou discorda*, 4 que o participante *discorda* e 5 significa que o participante *discorda totalmente* com a informação. As questões 1 (Q1) e 3 (Q3) foram aplicadas ao grupo A, enquanto as questões 2 (Q2) e 4 (Q4) foram aplicadas ao grupo B.

6.7.4.1 (Q1) **Não** tive dificuldade em especificar cenários de teste de aceitação usando *Fit tables.*

De acordo com o gráfico ilustrado na Figura 25, metade dos participantes concordam totalmente ou concordam com esta afirmação, 22% não têm certeza e 28% discordam ou discordam totalmente. Por outro lado, enquanto metade dos participantes deste experimento não tiveram dificuldade para especificar os cenários de teste de aceitação utilizando *Fit tables*, no experimento realizado com usuários não-técnicos, apresentado na Subseção 5.2.7.4, 78% dos participantes apontaram que encontraram dificuldade para especificar cenários de teste de aceitação utilizando as *Fit tables.*

Figura 25 – Não tive dificuldade em especificar cenários de teste de aceitação utilizando *Fit tables.*



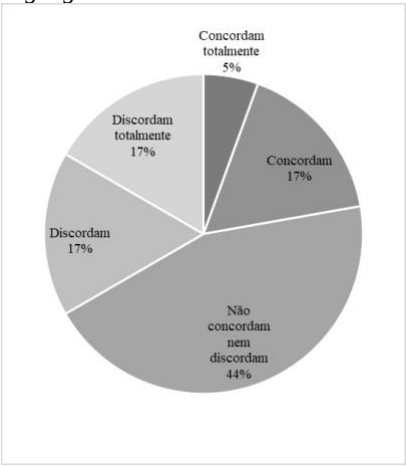
Fonte: Elaborada pelo autor (2018).

Adicionalmente, nos dois experimentos o número de cenários especificados corretamente, foi maior do que aqueles especificados incorretamente. Porém, neste experimento o percentual de sucesso foi maior do que no experimento apresentado na seção subseção 5.2.7.4. Estes dados mostram que, durante o processo de documentação dos requisitos de um software escritos com o auxílio das *Fit tables*, o apoio de um técnico aos usuários na execução desta atividade pode reduzir os riscos de erros, bem como tornar a atividade mais fácil para os usuários não-técnicos.

6.7.4.2 (Q2) **Não** tive dificuldade em especificar cenários de teste de aceitação usando o *Gherkin language*.

Assim como ilustra o gráfico apresentado na Figura 26, 34% dos participantes *discordam totalmente* ou *discordam* com esta afirmação, este percentual é maior que os 22% dos participantes que *concordam totalmente* ou *concordam*. O restante dos participantes (44%) não tem certeza sobre essa afirmação.

Figura 26 – Não tive dificuldade em especificar cenários de teste de aceitação utilizando *Gherkin language*.



Fonte: Elaborada pelo autor (2018).

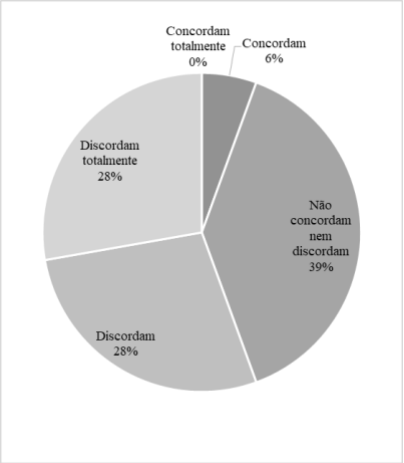
Como a *Gherkin language* é uma DSL para especificação do comportamento do sistema, utilizada no BDD, e possui sintaxe muito próxima da linguagem natural, era esperado que os participantes não reportassem dificuldade em especificar os cenários utilizando esta técnica. Este resultado é diferente do obtido no experimento 1, apresentado na Subseção 5.2.7.3.2, onde os usuário não-técnicos reportaram não encontrar dificuldade para especificar cenários de teste de aceitação utilizando *Gherkin language.* Todavia, nos dois experimentos os participantes obtiveram, em sua maioria, sucesso na elaboração das especificações.

Portanto, ao comparar os resultados do experimento apresentado na Seção 5.2, com os resultados apresentados por este experimento, podemos constatar que a percepção quanto à dificuldade em especificar requisitos através das *Fit tables* ou da *Gherkin language* varia de acordo com o perfil dos participantes. Enquanto os participantes técnicos apontaram ser mais fácil especificar os cenários de teste de aceitação utilizando as *Fit tables,* os não técnicos julgaram mais fácil especificar os cenários de teste de aceitação utilizando a *Gherkin language.*

6.7.4.3 (Q3) **Não** tive dificuldades em implementar novos requisitos funcionais no software de gestão de pessoas, cujas especificações foram expressas através de cenários de teste de aceitação escritos utilizando as *Fit tables*.

A maior parte dos participantes, 56%, relatou que teve dificuldade nas tarefas de implementação, somente 6% dos participantes concordam com esta afirmação. Apesar dos resultados das implementações terem sido positivos, os participantes acharam difícil desenvolver o software a partir dos testes de aceitação.

Figura 27 – Não tive dificuldades em implementar novos requisitos funcionais no software de gestão de pessoas, cujas especificações foram expressas através de cenários de teste de aceitação escritos utilizando as *Fit tables*.

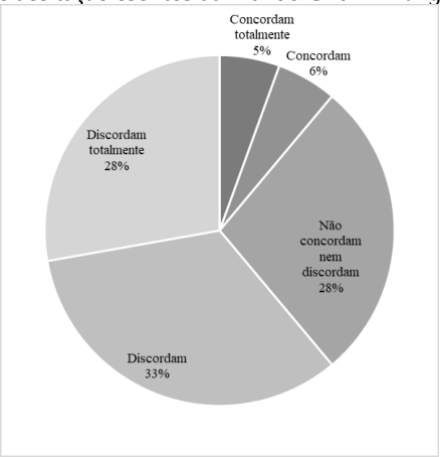


Fonte: Elaborada pelo autor (2018).

6.7.4.4 (Q4) **Não** tive dificuldades em implementar novos requisitos funcionais no software de gestão de pessoas, cujas especificações foram expressas através de cenários de teste de aceitação escritos utilizando *Gherkin language*.

Como pode ser observado na Figura 28, a maior parte dos participantes, 61%, relatou ter dificuldade nas tarefas de implementação, 28% dos participantes não têm certeza, e 11% dos participantes não encontrou dificuldade em implementar as alterações no software de gestão de pessoas tendo com fonte de informação dos requisitos cenários de teste de aceitação escritos utilizando *Gherkin language*.

Figura 28 – Não tive dificuldades em implementar novos requisitos funcionais no software de gestão de pessoas, cujas especificações foram expressas através de cenários de teste de aceitação escritos utilizando *Gherkin language*.



Fonte: Elaborada pelo autor (2018).

Sendo assim, analisando isoladamente as respostas dos participantes para as Q3 e Q4 do questionário, há um forte indício que os desenvolvedores preferiram a técnica *Gherkin language* à técnica *Fit tables*. Por outro lado, os resultados obtidos a partir da avaliação do código-fonte produzidos pelos participantes indicam um melhor desempenho das *Fit tables* como artefato para auxiliar a especificação dos requisitos funcionais do software. No entanto, conforme apresentado na Subseção 6.7.3, não há evidências estatísticas suficientes para afirmar que uma técnica seja melhor que outra para comunicar requisitos de software. Resultado este também apontado no experimento 2, apresentados na Subseção 5.3.7, onde as duas técnicas tiveram a mesma performance na comunicação dos requisitos funcionais de um software. Portanto, o conflito entre a opinião dos participantes e os resultados obtidos através das atividades práticas dos experimentos são indícios de que as técnicas são equivalentes.

6.8 AMEAÇAS PARA A VALIDADE DO EXPERIMENTO

Desde o planejamento do experimento até a sua execução, algumas vulnerabilidades foram encontradas. Algumas delas puderam ser

eliminadas, algumas delas foram tratadas e outras, no entanto, são ameaças para a validade deste experimento.

Por exemplo, requisitos de software atribuídos aos grupos A e B são diferentes para uma mesma etapa do experimento, todavia, para reduzir esta ameaça, buscou-se escolher requisitos que têm lógica e complexidade de negócios semelhantes. Porém, sabe-se, que a complexidade dos requisitos pode afetar os resultados, principalmente em relação ao tempo para a especificação dos cenários de teste de aceitação.

Tanto nas atividades de especificação dos cenários, quanto nas atividades de implementação do software, a validação dos resultados foi realizada por especialistas. Na parte 1, um especialista em testes de aceitação validou os artefatos entregues e, na parte 2, um analista de negócio o fez. Portanto, estes dois indivíduos ocuparam um papel muito importante no experimento, pois foram eles que decidiram o que estava correto e o que estava errado. No entanto, este experimento poderia ter sido conduzido sem estes indivíduos, desde que as seguintes ações fossem tomadas:

- Utilização de testes de aceitação especificados por um especialista como entrada de dados para a parte 2 do experimento, evitando que alguns erros nos cenários de teste de aceitação criados por outros participantes fossem ignorados pelo especialista;
- Utilização de testes de aceitação automatizados, ou até mesmo testes unitários, para validar se os códigos fonte implementados pelos participantes atendem às necessidades do usuário.

No entanto, foi escolhida a abordagem envolvendo tanto o especialista em teste de aceitação, quanto o analista de negócio, para aproximar nosso experimento a um cenário do mundo real, em que há uma variação no estilo de escrita dos cenários de teste de aceitação, como o vocabulário utilizado, expressões idiomáticas e nível de detalhes.

Outra ameaça refere-se ao preenchimento das planilhas eletrônicas para registrar o tempo gasto na tarefa de especificação de cenários de teste de aceitação. É muito difícil garantir, durante a execução do experimento, que todos os participantes estejam marcando corretamente o tempo gasto em cada uma das tarefas. Para dirimir este risco, durante o experimento verificou-se diversas vezes se os participantes estavam preenchendo as planilhas corretamente, além disto, foi solicitado diversas vezes aos participantes para tomar cuidado e atenção no preenchimento destas informações.

Por fim, o tamanho das amostras analisadas é muito pequeno, o que limita a capacidade de testes estatísticos. Neste estudo, o tempo para especificação dos cenários de teste de aceitação foi comparado com o teste t, e para as tabelas de contingência foi utilizado o *Fisher's exact test*.

7 CONCLUSÃO

Adquirir produtos de software é um desafio para as organizações públicas. Apesar de não serem divulgados números nos portais da transparência de organizações públicas, seria interessante estimar quantos milhões de reais são destinados por ano para a aquisição de produtos de software que não atendem as necessidades dos usuários. Números divulgados pelo *Standish Group* para processos de aquisição muito semelhantes aos processos licitatórios, mostram que 80% dos projetos de software entregues não atendem, ou atendem parcialmente as necessidades dos usuários.

Portanto, o emprego das técnicas de teste de aceitação como ferramenta para auxiliar a especificação dos requisitos funcionais de um software em editais de licitação é uma forma de tentar reduzir o risco do software a ser adquirido pela organização pública não atender as suas necessidades. Adicionalmente, a especificação dos testes de aceitação logo no início do projeto também pode contribuir para um melhor detalhamento das necessidades do usuário, pois à medida em que os cenários de teste de aceitação são construídos os usuários exercitam diferentes comportamentos esperados para o sistema, evitando que requisitos sejam esquecidos.

Entretanto, especificar e automatizar testes de aceitação consome tempo de analistas de sistemas, analistas de negócio, usuários e programadores. Dependendo dos recursos disponíveis, o custo para especificação de cenários de teste de aceitação para a especificação de todos os requisitos funcionais de um software pode inviabilizar uma aquisição de software. Assim, uma solução para esta limitação é empregar os cenários de teste de aceitação nos requisitos mais importantes ou críticos para a operação do negócio.

Todavia, nem todas as técnicas de teste de aceitação podem ser empregadas como ferramenta para auxiliar na especificação de requisitos. O conjunto de critérios técnicos e legais definidos nesta dissertação permitiu avaliar as técnicas que podem ser utilizadas com este objetivo.

A definição dos critérios técnicos e legais para a seleção das técnicas de teste de aceitação foi baseada nos resultados alcançados com a Revisão Sistemática de Literatura e a legislação vinculada para aquisição de produtos de software por organizações públicas da esfera federal e poder judiciário dos entes federativos, de acordo com recomendação publicada pelo Conselho Nacional de Justiça (CNJ). Porém, para a análise destes critérios e definição de quais técnicas de testes de aceitação atendiam ou não aos critérios de seleção, além dos

resultados obtidos através da Revisão Sistemática de Literatura, também foram conduzidos uma série de experimentos. Entre as 11 técnicas de teste de aceitação encontradas na Revisão Sistemática de Literatura, duas delas atenderam satisfatoriamente os requisitos legais e técnicos, são as *Fit tables* e a *Gherkin language*.

Um experimento realizado comparando a aplicabilidade das *Fit tables* e da *Gherkin language* como fermenta para auxiliar na especificação de requisitos funcionais mostrou que estas duas técnicas podem ser utilizadas para comunicar requisitos funcionais em projetos de desenvolvimento de software. Além disto, as duas técnicas apresentam níveis de dificuldade similares tanto para especificar os cenários de teste, quanto para entender estes cenários. A única diferença encontrada entre as *Fit tables* e a *Gherkin language* neste experimento está no tempo necessário para a especificação dos cenários de teste. Especificar testes de aceitação utilizando *Gherkin language* é 60% mais rápido do que utilizando *Fit tables.*

Outros experimentos realizados neste trabalho também permitiram observar que usuário não-técnicos são capazes de entender e especificar requisitos utilizando tanto as *Fit tables* quanto a *Gherkin language.* Porém, a comparação dos resultados dos experimentos realizados com técnicos e com usuários não-técnicos apontam que os usuários não-técnicos produziriam cenários de teste com maior qualidade quando assistidos por técnicos.

Além disto, desde o início do estudo, esperava-se a aderência das *Fit tables* e da *Gherkin language* ao objetivo desta dissertação, pois estas técnicas de teste de aceitação já são amplamente aplicadas tanto para validação de software em processos tradicionais quanto em processos ágeis, como o ATDD e BDD. Adicionalmente há uma gama de ferramentas e *frameworks* disponíveis que apoiam a documentação, desenvolvimento de código de cola e execução automatizada dos testes de aceitação especificados com estas técnicas.

Quanto às características, *Fit tables* e a *Gherkin language* são similares. Em uma primeira análise, as *Fit tables* parecem focar na representação dos critérios de aceitação, enquanto que os cenários escritos com a *Gherkin language* parecem focar nos cenários de aceitação. Porém, uma análise mais detalhada aponta que estas duas técnicas são semelhantes, pois enquanto as *Action Fixtures* do *Fit* lembram os *Scenarios* da *Gherkin language*, os *Scenarios Outline* da *Gherkin Language* lembram as *Column Fixtures* do *Fit.*

Por fim, os resultados mostrados nesta dissertação encorajam a utilização tanto das *Fit tables* quanto da *Gherkin language* como

ferramentas para auxiliar a especificação de requisitos em editais de licitação ou em contratos de terceirização de software.

7.1 PUBLICAÇÕES

O experimento detalhado no Capítulo 6 desta dissertação foi apresentado na *19th International Conference on Agile Software and Systems Development* (XP'2018) e publicado como um *full paper* no *Lecture Notes in Business Information Processing*. Além disto, uma versão inicial da Revisão Sistemática de Literatura foi publicada como um resumo estendido no *Proceedings of the 40th International Conference on Software Engineering* (ICSE'2018) e apresentada como pôster no mesmo evento.

7.2 TRABALHOS FUTUROS

Como trabalho futuro, sugere-se: (i) a análise da aplicação dos testes de aceitação em ciclos de manutenção de software, com o objetivo de avaliar fatores relacionadas ao custo para atualização dos testes dado a evolução do produto; (ii) o desenvolvimento de uma ferramenta que possa estimar, de forma automatizada, o tamanho de um software com base nos testes de aceitação, representados através de cenários escritos utilizando *Fit tables* ou *Gherkin language*, especificados para validá-lo; (iii) a análise da utilização dos testes de aceitação como ferramenta de qualidade continuado do software; (iv) o desenvolvimento de técnicas para a geração de dados de teste de aceitação; (v) desenvolvimento de ferramenta para geração do código do software a partir de testes de aceitação e de testes de unidade; e (vi) realização do *snowball* dos estudos selecionado na RSL em dois níveis e revisão dos resultados.

REFERÊNCIAS

ANDREA, J. Generative Acceptance Testing for Difficult-to-Test Software. **Lecture Notes in Computer Science**, p. 29–37, 2004.

ARAÚJO, B. C. et al. **Web-based tool for automatic acceptance test execution and scripting for programmers and customers**. The 2007 Euro American Conference on Telematics and Information System. [S.l.]: [s.n.]. 2007.

BECK, K. **Test-Driven Development:** By Example. [S.l.]: Addison-Wesley Professional, 2003.

BITTENCOURT, S. **Licitação Passo a Passo**. [S.l.]: Editora Fórum, 2014.

BRASIL. **Lei nº 8.666, de 21 de junho de 1993**. Brasília: Imprensa Oficial, 1993.

BRASIL. **Lei nº 10.520, de 17 de julho de 2002**. Brasília: Imprensa Oficial, 2002.

BRASIL. **Decreto nº 5.450, de 31 de maio de 2005**. Brasilia: Imprensa Oficial, 2005.

BRASIL. Instrução Normativa n° 04, de 11 de setembro de 2014. **Portal das Compras Governamentais**, 11 set. 2014. Disponivel em: <https://www.governoeletronico.gov.br/documentos-e-arquivos/1%20-%20IN%204%20%2011-9-14.pdf>. Acesso em: 12 Dezembro 2017.

BRASIL. Guia de Boas Práticas em Contratação de Soluções de Tecnologia da Informação. **Portal do Governo Eletrônico**, 2017. Disponivel em: <https://www.governodigital.gov.br/documentos-e-arquivos/Guia_de_Boas_Praticas_v3.pdf>. Acesso em: 26 Janeiro 2019.

CARRERA, Á.; IGLESIAS, C. A.; GARIJO, M. . Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development. **Information Systems Frontiers**, v. 16, 2014.

CARTER, J.; GARDNER, W. B. **BHive:** Towards behaviour-driven development supported by b-method. IEEE 17th International Conference on Information Reuse and Integration. [S.l.]: IEEE Press. 2016. p. 249–256.

CMMI PRODUCT TEAM. **CMMI for Development, Version 1.3**. Software Engineering Institute. Hanscom AFB, MA, USA. 2010.

COHN, M. **User Stories Applied for Agile Software Development**. Boston: Pearson Education, 2004. ISBN ISBN 0-321-20568-5.

CUCUMBER.IO. Gherkin Reference. **Cucumber Docs**, 2019. Disponivel em: <https://docs.cucumber.io/gherkin/reference/>. Acesso em: 10 jan. 2019.

CUCUMBER.IO. Gherkin Reference. **Cucumber:** Executable Specifications, 2019. Disponivel em: <https://docs.cucumber.io/gherkin/reference/>. Acesso em: 29 jan. 2019.

EL-ATTAR, M.; MILLER, J. Developing comprehensive acceptance tests from use cases and robustness diagrams. **Requirements Engineering**, v. 15, p. 285–306 , 2009.

GRANT THORNTON INTERNATIONAL BUSINESS REPORT. **Outsouring: driving efficiency and growth**. Grant Thornton. [S.l.]. 2014.

HÄTONEN, J.; ERIKSSON, T. 30+ years of research and practice of outsourcing – exploring the past and anticipating the future, Vol. 15 No. 2. 2009. Pág 142-55. **Journal of International Management**, v. 15, n. 2, p. 142-155, 2009.

HSIA, P.; KUNG, D.; SELL, C. **Software requirements and acceptance testing**. Annals of Software Engineering. [S.l.]: [s.n.]. 1997. p. 291–317.

HSIEH, C.; TSAI, C.; C., C. Y. **Test-Duo:** A framework for generating and executing automated acceptance tests from use cases. 8th International Workshop on Automation of Software Test. San Francisco: IEEE. 2013. p. 89-92. doi: 10.1109/IWAST.2013.6595797.

INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS. **A Guide to the Business Analysis Body of Knowledge v3**. Toronto: [s.n.], 2015. ISBN ISBN 978-1-927584-02-6.

ISO/IEC 25010. **Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models**. ISO/IEC. [S.l.]. 2011.

JURISTO, N.; MORENO, A. **Basics of Software Engineering Experimentation**. [S.l.]: Kluwer Academic Publishers, 2001.

KAMALAKAR, S.; EDWARDS, S. H.; DAO, T. M. **Automatically Generating Tests from Natural Language Descriptions of Software Behavior**. Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering. [S.l.]: [s.n.]. 2013. p. 238–245.

KHAN, S. U.; NIAZI, M.; AHMAD, R. A. Barriers in the selection of offshore software development outsourcing vendors: An exploratory study using a systematic literature review, v. 53, n. 7, p. 693-706, 2011. ISSN ISSN 0950-5849. https://doi.org/10.1016/j.infsof.2010.08.003.

KITCHENHAM, B. A. **Guidelines for Performing Systematic Literature Reviews in Software Engineering**. Software Engineering Group, Keele University and Department of Computer Science University of Durham. [S.l.]. 2007. Version 2.3.

LACITY, M. C.; WILLCOCKS, L. Information systems sourcing: examining the privatization option in USA public administration. **Information Systems Journal**, v. 7, p. 85–108, 1997.

LANDHÄUßER, M.; GENAID, A. **Connecting User Stories and Code for Test Development**. The Third International Workshop on Recommendation Systems for Software Engineering. Piscataway, NJ, USA: IEEE Press. 2012. p. 33–37.

LATORRE, R. A successful application of a Test-Driven Development strategy in the industrial environment. **Empirical Software Engineering**, v. 19, p. 53–773 , 2013.

LONGO, D. H.; VILAIN, P. **Creating user scenarios through user interaction diagrams by non-technical customers**. 27th International Conference on Software Engineering and Knowledge Engineering 2015. [S.l.]: [s.n.]. 2015. p. 330-335.

MELNIK, G.; MAURER, F. The Practice of Specifying Requirements Using Executable Acceptance Tests in Computer Science Courses. **Companion to the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications**, San Diego, CA, USA, 2005. 365-370.

MELNIK, G.; MAURER, F.; CHIASSON, M. **Executable Acceptance Tests for Communicating Business Requirements:** Customer Perspective. AGILE 2006. [S.l.]: [s.n.]. 2006. p. 35-46.

MEYER, B. On Formalism in Specifications. **IEEE Software**, v. 2, n. 1, p. 6-26, Janeiro 1985.

MIRANDA, H. S. **Licitações e contratos administrativos:** Lei 8.666/1993. Brasília: Alumnus, 2012. ISBN ISBN 978-85-65295-07-9.

MOKETAR, N. A. et al. **TestMEReq:** Generating Abstract Tests for Requirements Validation. The 3rd International Workshop on Software Engineering Research and Industrial Practice. Austin, TX, USA: ACM. 2016. p. 39–45.

MUGRIDGE, R.; CUNNINGHAN, W. **Fit for Developing Software:** Framework for Integrated Tests. Upper Saddle River: Pearson Education, 2005.

OLIVEIRA, D. R. **Organização e métodos:** uma abordagem gerencial. 14ª edição. ed. São Paulo: Atlas, 2005.

PARK, S. S.; MAURER, F. **The Benefits and Challenges of Executable Acceptance Testing**. Proceedings of the 2008 International

Workshop on Scrutinizing Agile Practices. New York City, NY, USA: ACM. 2008. p. pp. 19–22.

PARK, S.; MAURER, F. Communicating Domain Knowledge in Executable Acceptance Test Driven Development. **Lecture Notes in Business Information Processing**, p. 23–32, 2009.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software:** Uma abordagem profissional. 8ª edição. ed. Porto Alegre: MacGraw Hill Education, 2016. ISBN ISBN 978-85-8055-534-9.

RICCA, F. et al. Using acceptance tests as a support for clarifying requirements: A series of experiments. **Information and Software Technology**, 51, n. 2, 2009. 270-283.

ROSE, S.; WYNNE, M.; HELLESØY, A. **The Cucumber for Java Book:** Behaviour-Driven Development for Testers and Developers. First Edition. ed. [S.l.]: Pragmatic Bookshelf, 2015.

SAUVÉ, J. P.; ABATH NETO, O. L.; CIRNE, W. **EasyAccept:** A Tool to Easily Create, Run and Drive Development with Automated Acceptance Tests. The 2006 International Workshop on Automation of Software Test. New York City, NY, USA: ACM. 2006. p. 111–117.

SEO, K. I.; CHOI, E. M. **Comparison of Five Black-box Testing Methods for Object-Oriented Software**. Fourth International Conference on Software Engineering Research, Management and Applications. [S.l.]: [s.n.]. 2006. p. 213–220.

SOEKEN, M.; WILLE, R.; DRECHSLER, R. Assisted behavior driven development using natural language processing. **Lecture Notes in Computer Science**, v. 7304, p. 269–287 , 2012.

SOMMERVILLE, I. **Software Engineering**. [S.l.]: Pearson Education. 2010.

TALBY, D. et al. **A process-complete automatic acceptance testing framework**. [S.l.]: Proceedings - IEEE International Conference on Software - Science, Technology and Engineering, 2005. 129–138 p.

TORCHIANO, M.; RICCA, F.; PENTA, M. D. **"Talking tests":** A preliminary experimental study on fit user acceptance tests. 1st International Symposium on Empirical Software Engineering and Measurement. [S.l.]: [s.n.]. 2007. p. 464–466.

VAZQUEZ, C. E.; SIMÕES, G. S.; ALBERT, R. M. **Análise de Pontos de Função**. 13ª Edição. ed. [S.l.]: Saraiva, 2013. 272 p. ISBN ISBN 8536504528.

WANDERLEY, F.; SILVA, A.; ARAÚJO, J. **Evaluation of BehaviorMap:** A user-centered behavior language. International Conference on Research Challenges in Information Science. [S.l.]: [s.n.]. 2015. p. 309–320.

WOHLIN, C. et al. **Experimentation in Software Engineering – An Introduction**. [S.l.]: Springer, 2012.

YOUNG, R. **Effective Requirements Practice**. Boston: Addison-Wesley, 2001.

APÊNDICE A – Estudos selecionados

1. Melnik, G., Maurer, F., Chiasson, M.: Executable acceptance tests for communicating business requirements: Customer perspective. In: Proceedings - AGILE Conference, 2006. pp. 35–46 (2006).
2. Read, K., Melnik, G., Maurer, F.: Student experiences with executable acceptance testing. In: Proceedings - AGILE Conference 2005. pp. 312–317 (2005).
3. Melnik, G., Maurer, F.: The Practice of Specifying Requirements Using Executable Acceptance Tests in Computer Science Courses. In: Companion to the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications. pp. 365–370. ACM, New York, NY, USA (2005).
4. Torchiano, M., Ricca, F., Penta, M.D.: "Talking tests": A preliminary experimental study on fit user acceptance tests. In: Proceedings - 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007. pp. 464–466 (2007).
5. Ricca, F., Torchiano, M., Penta, M.D., Ceccato, M., Tonella, P.: Using acceptance tests as a support for clarifying requirements: A series of experiments. Information and Software Technology. 51, 270–283 (2009).
6. Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., Ceccato, M., Visaggio, C.A.: Are fit tables really talking? In: Proceedings of the 30th International Conference on Software Engineering. pp. 361–370. ACM, New York, NY, USA (2008).
7. Gandhi, P.., Haugen, N.C.., Hill, M.., Watt, R..: Creating a living specification using FIT documents. In: Proceedings - AGILE Conference 2005. pp. 253–258 (2005).
8. Martin, R.C., Melnik, G.: Tests and Requirements, Requirements and Tests: A Möbius Strip. IEEE Software. 25, 54–59 (2008).
9. Park, S.S., Maurer, F.: The Benefits and Challenges of Executable Acceptance Testing. In: Proceedings of the 2008 International Workshop on Scrutinizing Agile Practices or Shoot-out at the Agile Corral. pp. 19–22. ACM, New York, NY, USA (2008).
10. Mahmood, S.: The impact of acceptance tests on analyzing component-based systems specifications: An experimental evaluation. In: Proceedings - 10th IEEE International Conference

on Computer and Information Technology, CIT-2010, 7th IEEE International Conference on Embedded Software and Systems, ICESS-2010, ScalCom-2010. pp. 241–248 (2010). Borg, R., Kropp, M.: Automated Acceptance Test Refactoring. In: Proceedings of the 4th Workshop on Refactoring Tools. pp. 15–21. ACM, New York, NY, USA (2011).

11. Connolly, D., Keenan, F., McCaffery, F.: Developing acceptance tests from existing documentation using annotations: An experiment. In: 2009 ICSE Workshop on Automation of Software Test. pp. 123–129 (2009).
12. Chengyao, D., Wilson, P., Maurer, F.: FitClipse: A fit-based eclipse plug-in for executable acceptance test driven development. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 4536 LNCS, 93–100 (2007).
13. Druk, M., Kropp, M.: ReFit: A Fit test maintenance plug-in for the Eclipse refactoring plug-in. In: 2013 3rd International Workshop on Developing Tools as Plug-Ins (TOPI). pp. 7–12 (2013).
14. Ricca, F., Torchiano, M., Ceccato, M., Tonella, P.: Talking Tests: An Empirical Assessment of the Role of Fit Acceptance Tests in Clarifying Requirements. In: Ninth International Workshop on Principles of Software Evolution: In Conjunction with the 6th ESEC/FSE Joint Meeting. pp. 51–58. ACM, New York, NY, USA (2007).
15. Haugset, B., Stalhane, T.: Automated Acceptance Testing as an Agile Requirements Engineering Practice. In: 2012 45th Hawaii International Conference on System Sciences. pp. 5289–5298 (2012).
16. Khandkar, S.H., Park, S., Ghanam, Y., Maurer, F.: FitClipse: A tool for executable acceptance test driven development. Lecture Notes in Business Information Processing. 31 LNBIP, 259–260 (2009).
17. El-Attar, M., Miller, J.: Developing comprehensive acceptance tests from use cases and robustness diagrams. Requirements Engineering. 15, 285–306 (2009).
18. Park, S., Maurer, F.: Communicating Domain Knowledge in Executable Acceptance Test Driven Development. In: Lecture Notes in Business Information Processing. pp. 23–32. Springer Berlin Heidelberg (2009).

19. Andrea, J.: Generative Acceptance Testing for Difficult-to-Test Software. In: Lecture Notes in Computer Science. pp. 29–37. Springer Berlin Heidelberg (2004).
20. Talby, D., Nakar, O., Shmueli, N., Margolin, E., Keren, A.: A process-complete automatic acceptance testing framework. In: Proceedings - IEEE International Conference on Software - Science, Technology and Engineering 2005, SwSTE '05. pp. 129–138 (2005).
21. Massila, K., M., N.A., John, G., John, H., Mark, R.: Automatic Acceptance Test Case Generation From Essential Use Cases. Frontiers in Artificial Intelligence and Applications. 265, 246–255 (2014).
22. Moketar, N.A., Kamalrudin, M., Sidek, S., Robinson, M., Grundy, J.: TestMEReq: Generating Abstract Tests for Requirements Validation. In: Proceedings of the 3rd International Workshop on Software Engineering Research and Industrial Practice. pp. 39–45. ACM, New York, NY, USA (2016).
23. Moketar, N.A., Kamalrudin, M., Sidek, S., Robinson, M., Grundy, J.: An Automated Collaborative Requirements Engineering Tool for Better Validation of Requirements. In: Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. pp. 864–869. ACM, New York, NY, USA (2016).
24. Latorre, R.: A successful application of a Test-Driven Development strategy in the industrial environment. Empirical Software Engineering. 19, 753–773 (2013).
25. Connolly, D., Keenan, F., Caffery, F.M.: AnnoTestWeb/Run: Annotations Based Acceptance Testing. Lecture Notes in Business Information Processing. 48 LNBIP, 381–382 (2010).
26. Nomura, N., Kikushima, Y., Aoyama, M.: Business-Driven Acceptance Testing Methodology and Its Practice for E-Government Software Systems. In: 2013 20th Asia-Pacific Software Engineering Conference (APSEC). pp. 99–104. IEEE (2013).
27. Hsieh, C., Tsai, C, Cheng, Y.C.: Test-duo: A Framework for Generating and Executing Automated Acceptance Tests from Use Cases. In: Proceedings of the 8th International Workshop on Automation of Software Test. pp. 89–92. IEEE Press, Piscataway, NJ, USA (2013).

28. Pereira, V., Do Prado, A.F.: A new agile process for Web development. In: ENASE 2011 - Proceedings of the 6th International Conference on Evaluation of Novel Approaches to Software Engineering. pp. 177–187 (2011).
29. Carrera, Á., Iglesias, C.A., Garijo, M.: Beast methodology: An agile testing methodology for multi-agent systems based on behaviour driven development. Information Systems Frontiers. 16, (2014).
30. Landhäußer, M., Genaid, A.: Connecting User Stories and Code for Test Development. In: Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering. pp. 33–37. IEEE Press, Piscataway, NJ, USA (2012).
31. Kamalakar, S., Edwards, S.H., Dao, T.M.: Automatically Generating Tests from Natural Language Descriptions of Software Behavior. In: Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering. pp. 238–245. Scitepress (2013).
32. Lazar, I., Motogna, S., Parv, B.: Behaviour-Driven Development of Foundational UML Components. Electronic Notes in Theoretical Computer Science. 264, 91–105 (2010).
33. Carter, J., Gardner, W.B.: BHive: Towards behaviour-driven development supported by b-method. In: Proceedings - 2016 IEEE 17th International Conference on Information Reuse and Integration, IRI 2016. pp. 249–256 (2016).
34. King, T.M., Nunez, G., Santiago, D., Cando, A., Mack, C.: Legend: An Agile DSL Toolset for Web Acceptance Testing. In: Proceedings of the 2014 International Symposium on Software Testing and Analysis. pp. 409–412. ACM, New York, NY, USA (2014).
35. Almeida, L., Cirilo, E., Barbosa, E.A.: SS-BDD. In: Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing - SAST. p. 5:1–5:10. ACM Press, New York, NY, USA (2016).
36. Soeken, M., Wille, R., Drechsler, R.: Assisted behavior driven development using natural language processing. Lecture Notes in Computer Science. 7304 LNCS, 269–287 (2012).
37. Wanderley, F., Silva, A., Araujo, J.: Evaluation of BehaviorMap: A user-centered behavior language. In: Proceedings - International Conference on Research Challenges in Information Science. pp. 309–320 (2015).

38. Verma, R.P., Beg, M.R.: Generation of Test Cases from Software Requirements Using Natural Language Processing. In: 2013 6th International Conference on Emerging Trends in Engineering and Technology. pp. 140–147. IEEE (2013).
39. De Santiago Júnior, V.A., Vijaykumar, N.L.: Generating model-based test cases from natural language requirements for space application software. Software Quality Journal. 20, 77–143 (2011).
40. Sneed, H.M.: Testing against Natural Language Requirements. In: Seventh International Conference on Quality Software (QSIC 2007). pp. 380–387. IEEE (2007).
41. Sauvé, J.P., Abath Neto, O.L., Cirne, W.: EasyAccept: A Tool to Easily Create, Run and Drive Development with Automated Acceptance Tests. In: Proceedings of the 2006 International Workshop on Automation of Software Test. pp. 111–117. ACM, New York, NY, USA (2006).
42. Sauvé, J.P., Abath Neto, O.L.: Teaching Software Development with ATDD and Easyaccept. In: SIGCSE Bull. pp. 542–546. ACM, New York, NY, USA (2008).
43. Araújo, B.C., Rocha, A.C., Xavier, A., Muniz, A.I., Garcia, F.P.: Web-based tool for automatic acceptance test execution and scripting for programmers and customers. In: Proceedings of the 2007 Euro American Conference on Telematics and Information Systems. p. 56:1–56:4. ACM, New York, NY, USA (2007).
44. Weiss, J., Mandl, P., Schill, A.: Introducing the QCEP-testing System for Executable Acceptance Test Driven Development of Complex Event Processing Applications. In: Proceedings of the 2013 International Workshop on Joining AcadeMiA and Industry Contributions to Testing Automation. pp. 13–18. ACM, New York, NY, USA (2013).
45. Weiss, J., Schill, A.: Specifying Executable or Nonexecutable Acceptance Test Cases for Event Processing Applications. In: 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). pp. 235–239. IEEE (2016).
46. Hsia, P., Kung, D., Sell, C.: Software requirements and acceptance testing. Annals of Software Engineering. 3, 291–317 (1997).
47. Longo, D.H., Vilain, P.: Creating user scenarios through user interaction diagrams by non-technical customers. In: Proceedings of the International Conference on Software

Engineering and Knowledge Engineering, SEKE. pp. 330–335 (2015).
48. Longo, D.H., Vilain, P.: User scenarios through user interaction diagrams. International Journal of Software Engineering and Knowledge Engineering. 25, 1771–1775 (2015).
49. Clerissi, D., Leotta, M., Reggio, G., Ricca, F.: A lightweight semi-automated acceptance test-driven development approach for web applications. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 9671, 593–597 (2016).
50. Besson, F.M., Beder, D.M., Chaim, M.L.: An automated approach for acceptance web test case modeling and executing. Lecture Notes in Business Information Processing. 48 LNBIP, 160–165 (2010).
51. Borg, R., Kropp, M.: Automated Acceptance Test Refactoring. In: Proceedings of the 4th Workshop on Refactoring Tools. pp. 15–21. ACM, New York, NY, USA (2011).
52. Elghondakly, R., Moussa, S., Badr, N.: Waterfall and agile requirements-based model for automated test cases generation. In: 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS). pp. 607–612. IEEE (2015).
53. Sinha, A., Smidts, C.: An experimental evaluation of a higher-ordered-typed-functional specification-based test-generation technique. Empirical Software Engineering. 11, 173–202 (2006).
54. Elssamadisy, A., Whitmore, J.: Functional Testing: A Pattern to Follow and the Smells to Avoid. In: Proceedings of the 2006 Conference on Pattern Languages of Programs. p. 27:1–27:13. ACM, New York, NY, USA (2006)

APÊNDICE B – Estudos excluídos

Nesta Seção são listados os estudos que foram excluídos da RSL, pois empregavam técnicas de validação interna no sistema, isto é, testes funcionais ou testes de sistema.

1. Seo, K.I., Choi, E.M.: Comparison of Five Black-box Testing Methods for Object-Oriented Software. In: Fourth International Conference on Software Engineering Research, Management and Applications. pp. 213–220 (2006).
2. Lugato, D., Maraux, F., Traon, Y.L., Nebut, C., Normand, V., Dubois, H., Pierron, J., Gallois, J.: Automated functional test case synthesis from THALES industrial requirements. In: Proceedings - IEEE Real-Time and Embedded Technology and Applications Symposium. pp. 104–111 (2004).
3. Stocks, P., Carrington, D.: Test template framework: a specification-based testing case study. In: Proceedings of the 1993 International Symposium on Software Testing and Analysis. pp. 11–18 (1993).
4. Geppert, B., Li, J., Rößler, F., Weiss, D.M.: Towards generating acceptance tests for product lines. Lecture Notes in Computer Science. 3107, 35–48 (2004).
5. Hörcher, H., Peleska, J.: Using formal specifications to support software testing. Software Quality Journal. 4, 309–327 (1995).
6. Razak, R.A., Fahrurazi, F.R.: Agile testing with Selenium. In: 2011 5th Malaysian Conference in Software Engineering, MySEC 2011. pp. 217–219 (2011).
7. Holmes, A., Kellogg, M.: Automating Functional Tests Using Selenium. In: AGILE 2006. pp. 270–275. IEEE (2006).
8. Alégroth, E.., Nass, M.., Olsson, H.H..: JAutomate: A tool for system- and acceptance-test automation. In: Proceedings - IEEE 6th International Conference on Software Testing, Verification and Validation, ICST 2013. pp. 439–446 (2013).
9. Alsmadi, I.: The Utilization of User Sessions in Testing. In: Seventh IEEE/ACIS International Conference on Computer and Information Science, ICIS 2008. pp. 581–585 (2008).
10. Lowell, C., Stell-Smith, J.: Successful automation of GUI driven acceptance testing. Lecture Notes in Computer Science. 2675, 331–333 (2003).
11. Bruns, A., Kornstadt, A., Wichmann, D.: Web application tests with selenium. IEEE Software. 26, 88–91 (2009).

12. Arnold, D., Corriveau, J., Shi, W.: Modeling and Validating Requirements Using Executable Cotnracts and Scenarios. In: 2010 Eighth ACIS International Conference on Software Engineering Research, Management and Applications. pp. 311–320. IEEE (2010).
13. Andrea, J.: Putting a Motor on the Canoo WebTest Acceptance Testing Framework. In: Lecture Notes in Computer Science. pp. 20–28. Springer Berlin Heidelberg (2004).
14. Arnold, D., Corriveau, J., Shi, W.: Scenario-Based Validation: Beyond the User Requirements Notation. In: 2010 21st Australian Software Engineering Conference. pp. 75–84. IEEE (2010).
15. Wong, P.Y.H., Bubel, R., de Boer, F.S., Gómez-Zamalloa, M., de Gouw, S., Hähnle, R., Meinke, K., Sindhu, M.A.: Testing abstract behavioral specifications. International Journal on Software Tools for Technology Transfer. 17, 107–119 (2014).
16. Guillemot, M., König, D.: Web testing made easy. In: Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications - OOPSLA 06. pp. 692–693. ACM Press (2006).
17. Baldini, A., Benso, A., Prinetto, P.: System-level functional testing from UML specifications in end-of-production industrial environments. International Journal on Software Tools for Technology Transfer. 7, 326–340 (2004).
18. Faria, J.P., Paiva, A.C.R.: A toolset for conformance testing against UML sequence diagrams based on event-driven colored Petri nets. International Journal on Software Tools for Technology Transfer. 18, 285–304 (2014).
19. Massollar, J.L., Mello, R.M. de, Travassos, G.H.: Structuring and Verifying Requirement Specifications through Activity Diagrams to Support the Semi-automated Generation of Functional Test Procedures. In: 2012 Eighth International Conference on the Quality of Information and Communications Technology. pp. 239–244 (2012).
20. Huang, C., Chen, H.Y.: A Tool to Support Automated Testing for Web Application Scenario. In: 2006 IEEE International Conference on Systems, Man and Cybernetics. pp. 2179–2184. IEEE (2006).

APÊNDICE C – Cenários de Casos de Uso

Neste apêndice são descritos os cenários de casos de uso utilizados como referência para a construção dos cenários de teste de aceitação que exemplificaram as técnicas encontradas na RSL.

1 SACAR DINHEIRO EM UM CAIXA ELETRÔNICO

Nesta seção são descritos o fluxo básico e três fluxos alternativos de eventos da interação entre um cliente e um caixa eletrônico para realizar o caso de uso "Sacar dinheiro de uma conta corrente".

1.1 PRÉ-CONDIÇÕES

- O cliente identificou-se com sucesso no caixa eletrônico.
- O caixa eletrônico está carregado com cédulas de dinheiro.
- O caixa eletrônico está funcionando corretamente.

1.1.1 Fluxo básico de eventos

1. O cliente escolhe a opção "Sacar".
2. O sistema informa ao cliente as cédulas disponíveis no caixa eletrônico.
3. O cliente informa o montante que deseja sacar.
4. O sistema valida o valor informado.
5. O cliente aprova a transação utilizando uma senha de 6 dígitos numéricos.
6. O sistema determina que a senha é válida.
7. O sistema determina que o saldo do cliente é suficiente para realizar a retirada.
8. O sistema libera o cartão magnético do cliente.
9. O cliente retira o cartão magnético do caixa eletrônico.
10. O sistema entrega o valor solicitado pelo cliente na bandeja do caixa eletrônico.
11. O cliente retira o dinheiro da bandeja.
12. Fim do caso de uso.

1.1.1 Fluxos alternativos

A1) Valor incorreto

4.1. O sistema mostra a mensagem que o valor informado pelo cliente é inválido.

4.2. O sistema solicita ao cliente que tente novamente e retorna para o passo 12 do *Fluxo Básico.*

A2) Senha incorreta

6.1. O sistema mostra a mensagem que a senha informada pelo cliente está errada. Atenção: se o cliente informar a senha errada três vezes no período de 72 horas, o cartão magnético da conta corrente do cliente é automaticamente bloqueado.

6.2. O sistema libera o cartão magnético do cliente.

6.3. O cliente retira o cartão magnético da caixa eletrônico.

6.4. O fluxo retorna para o passo 12 do *Fluxo Básico.*

A3) Saldo insuficiente

7.1. O sistema mostra a mensagem que não há saldo suficiente.

7.2. O sistema libera o cartão magnético do cliente.

7.3. O cliente retira o cartão magnético da caixa eletrônico.

7.4. O fluxo retorna para o passo 12 do *Fluxo Básico.*

2 CALCULAR O VALOR DAS PARCELAS MENSAIS (EMI) DE UM FINANCIAMENTO

As parcelas fixas de um financiamento, denominadas EMI, são valores fixos mensais que o cliente paga ao banco em uma determinada data. Este valor já inclui os juros do financiamento, e pode ser calculado utilizando a fórmula matemática a seguir:

$EMI = P \times R \times \frac{(1+R)^N}{((1+R)^N-1)}$, onde P é o valor total financiado, R é a taxa de juros mensal, e N é o número de parcelas.

2.1 FLUXO BÁSICO DE EVENTOS

1. O sistema mostra o formulário de simulação de financiamento.

2. O usuário informa o montante a ser financiado, a taxa de juros mensais e a quantidade de parcelas.
3. O sistema mostra o valor das parcelas e o valor final total a ser pago pelo capital com os juros.

APÊNDICE D – Técnicas de teste de aceitação

Neste apêndice são descritas cada uma destas classes de testes de aceitação a as técnicas atribuídas a elas. As especificações dos cenários de testes de aceitação apresentadas neste apêndice foram construídas com base nos cenários de casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

1 TABELAS

Nas técnicas de testes de aceitação tabulares, os cenários de testes são expressados através de linhas e colunas, formando tabelas que contêm as entradas de dados e as saídas esperadas do software. Estes cenários podem ou não vir acompanhados de pequenos parágrafos de texto, cujo objetivo é complementar as informações apresentadas nas tabelas. Nos estudos selecionados foram identificadas quatro técnicas tabulares de testes de aceitação, são elas:

- T01 – *Fit*/*Fit tables*
- T02 – Planilha eletrônica com descrição de contexto
- T03 – Planilha eletrônica baseada em controle de estado dos objetos
- T04 – Linguagem de Teste de Domínio Específico estruturada em tabelas

1.1 FIT/*FIT TABLES*

O *Fit*, acrônimo para *Framework for Integrated Tests*, é um *framework* para especificação e execução de testes de aceitação. Destacado pelos estudos como a técnica de testes de aceitação mais popular (PARK e MAURER, 2008), o *Fit* utiliza tabelas, conhecidas como *Fit tables* ou *Fit Fixtures* para expressar os cenários de teste de aceitação. A biblioteca padrão do *Fit – Fit library* – provê um conjunto de tabelas ou *fixtures*, como por exemplo, a *Column Fixture*, a *Set Up Fixture* e a *Action Fixture*. Além das tabelas disponíveis na *Fit library*, ainda é possível criar novas tabelas para atender a necessidades específicas (MUGRIDGE e CUNNINGHAN, 2005).

A Tabela 25 mostra um exemplo de *Column Fixture* para o fluxo básico de eventos do caso de uso "Calcular parcelas mensais de um financiamento" especificado no APÊNDICE C – Cenários de Casos de Uso. Cada linha desta tabela é um caso de teste, exceto a primeira e

segunda linhas. A primeira linha indica para o motor do *Fit* qual é classe que conecta o cenário de teste ao software em teste, esta classe é conhecida como código de cola. A segunda linha fornece identificadores para dados de entrada e saídas esperadas. Identificadores que são seguidos pelo ponto de interrogação '?' ou parêntese '()' são saídas esperadas (valores de asserção), os outros são dados de entrada.

Tabela 25 – Exemplo de *Column Fixture*

br.ufsc.acceptanceTests.calcularEMIColumnFixture				
valor a financiar	% de juros mensal	número de parcelas	valor das parcelas?	valor total financiado com juros?
10.000	0,5334	2	862,51	10.350,09
500	0,1250	5	100,38	501,88
3.250.500	0,7665	72	58.911,94	4.241.659,83

Fonte: Elaborada pelo autor (2018).

Ocasionalmente, a execução de um caso de teste depende de que objetos da aplicação apresentem um determinado estado. Para satisfazer estes pré-requisitos são realizadas configurações na aplicação antes de executar esse caso de teste. No *Fit*, estas configurações são promovidas pelas *SetUp Fixtures.*

A Tabela 26 mostra um exemplo de *SetUp Fixture*. A primeira linha indica a classe de código de cola. A segunda linha o conjunto de parâmetros que serão passados como argumentos para o método de configuração da aplicação. Finalmente, a terceira linha contém o valor de cada um destes parâmetros. O objetivo desta *SetUp Fixture* é preparar o aplicativo para executar o teste que será apresentado na Tabela 27, que é uma *Action Fixture*.

Tabela 26 – Exemplo de *SetUp Fixture*.

br.ufsc.acceptanceTests.SacarDinheiroSetUpFixture					
usuário	Conta corrente	saldo	número do cartão	senha	notas (nota, quantidade)
J.F. Piper	125654-08	90,00	9999 8888 7777 6666 5555	123456	{(5,0), (10,200), (20,0), (50,150), (100,100)}

Fonte: Elaborada pelo autor (2018).

Tabela 27 – Exemplo de *Action Fixture*

ActionFixture		
start	br.ufsc.acceptanceTests.SacarDinheiroActionFixture	
enter	valor	30.00
press	OK	
check	é um valor válido	true
enter	senha	123456
press	confirmar	
check	é uma senha válida	true
check	saldo atualizado	60.00

Fonte: Elaborada pelo autor (2018).

Uma *Action Fixture* expressa uma sequência de ações representadas por comandos específicos que descrevem a interação entre o usuário e o software. A Tabela 27 mostra um exemplo de *Action Fixture*.

As Tabelas Tabela 26 e Tabela 27, quando inseridas em um mesmo contexto, especificam o teste para o fluxo básico de eventos do caso de uso "Sacar dinheiro em um caixa eletrônico" especificado no APÊNDICE C – Cenários de Casos de Uso. Na primeira linha da Tabela 27, o comando **ActionFixture** define que a tabela é uma *Action Fixture*. A primeira coluna de cada uma das linhas subsequentes contém um comando. Na biblioteca padrão do *Fit*, as *Action Fixtures* oferecem apenas quatro comandos, todavia, podem ser criadas subclasses que ampliem este conjunto. Os comandos são:

- **start**: é o primeiro comando de uma *Action Fixture* e indica qual é a classe de código de cola. Os comandos subsequentes serão invocações para métodos desta classe
- **enter**: é um comando que possui dois parâmetros: o nome de um método e um valor. Ao ser executado, o comando **enter** irá invocar este método – implementado na classe indicada pelo comando **start** – passando o valor informado como argumento. Portanto, este comando equivale a uma entrada de dados do usuário.
- **press**: possui um único parâmetro, o nome de um método. Ao ser executado, o comando **press** invocará este método, representando desta forma uma ação executada pelo usuário.
- **check**: contém os seguintes parâmetros: nome de método e um valor. Este comando invocará o método informado e irá verificar se o retorno do método é igual ao valor informado no teste. O objetivo deste comando é validar os retornos do software.

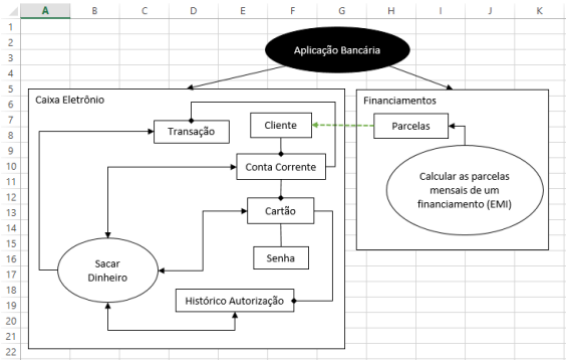
O APÊNDICE E – Exemplo de cenários de teste escritos utilizando Fit Tables contém a versão completa com a especificação de todos os cenários de teste para os fluxos dos casos de uso detalhados no APÊNDICE C – Cenários de Casos de Uso.

1.2 PLANILHA ELETRÔNICA COM DESCRIÇÃO DE CONTEXTO

Os cenários de teste de aceitação representados através de planilhas com descrição de contexto, apresentam além das tabelas que determinam as saídas esperadas do software para um conjunto de entradas, e também fornecem uma descrição, através de um diagrama, do contexto dos

cenários, oferecendo aos desenvolvedores uma visão geral sobre o domínio da aplicação. A Figura 29 ilustra um exemplo de diagrama utilizado para descrever o contexto de um aplicativo bancário. Neste diagrama, as elipses brancas representam as funcionalidades do software. Estas elipses, por sua vez, estão conectadas a pequenos retângulos, que representam as entidades. A elipse preta simboliza o aplicativo – o domínio dos testes –, e está conectada aos retângulos grandes, que representam os subdomínios ou módulos do aplicativo (PARK e MAURER, 2009).

Figura 29 – Descrição do contexto dos cenários de teste



Fonte: Elaborada pelo autor (2019).

Dado que esta técnica é um formato tabular, os casos de teste e a descrição de contexto são especificados em planilhas eletrônicas. A Tabela 28 mostra um exemplo de cenário de teste usando esta notação para o caso de uso "Calcular parcelas mensais de um financiamento" detalhado no APÊNDICE C – Cenários de Casos de Uso. Nesta tabela, as colunas com cabeçalho escrito em letras minúsculas representam as entradas de dados, enquanto os cabeçalhos escritos com letras maiúsculas representam as saídas esperadas do sistema.

Tabela 28 – Exemplo de teste de aceitação em planilha eletrônica com descrição de contexto.

valor a financiar	% de juros mensal	número de parcelas	VALOR DAS PARCELAS?	VALOR TOTAL FINANCIADO COM JUROS?
10.000	0,5334	2	862,51	10.350,09
500	0,1250	5	100,38	501,88
3.250.500	0,7665	72	58.911,94	4.241.659,83

Fonte: Elaborada pelo autor (2019).

Tabela 29 – Exemplo de teste de aceitação em planilha eletrônica com contexto.

cliente	conta corrente	saldo	número do cartão	número de vezes que informou a senha errada nas últimas 72 horas.	valor	senha	VALOR VÁLIDO?	SENHA CORRETA?	SALDO?	MENSAGEM DE RETORNO
J. F. Piper	125654-08	90,00	9999 8888 7777 6666	<=2	30	123456	true	true	60,00	Operação realizada com Sucesso! Retire o dinheiro na bandeja.
					31	*	false	*	90,00	O valor informado é inválido! Tente novamente.
					100	123456	true	true	90,00	Saldo insuficiente. Verifique seu saldo e tente novamente.
				<1	30	123412	true	false	90,00	Senha inválida!
				=1	30	123412	true	false	90,00	Senha inválida! Você tem mais uma tentativa, caso informe a senha errada novamente seu cartão será bloqueado.
				=2	30	123412	true	false	90,00	Senha inválida! Seu cartão foi bloqueado. Contate a central de atendimento ao cliente para maiores informações.
				>=3	30	123456	true	true	90,00	Seu cartão está bloqueado! Por favor, entre em contato com a central de atendimento para maiores informações.

Fonte: Elaborada pelo autor (2019).

Nos casos em que a entrada de dados é a mesma para vários casos de teste, é possível mesclar as células de uma mesma coluna, assim como apresentado na Tabela 29. Esta tabela apresenta a especificação dos casos de teste para os fluxos principal e alternativos do caso de uso "Sacar dinheiro em um caixa eletrônico" detalhado no APÊNDICE C – Cenários de Casos de Uso.

1.3 PLANILHA ELETRÔNICA COM VALIDAÇÃO ATRAVÉS DO ESTADO DOS OBJETOS DA APLICAÇÃO

A Figura 30 mostra a especificação de testes de aceitação utilizando planilha eletrônica com validação através do estado dos objetos da aplicação. O objetivo deste teste é validar o fluxo básico do caso de uso "Sacar dinheiro em uma caixa eletrônico" especificado no APÊNDICE C – Cenários de Casos de Uso. Assim como ilustra a Figura 30, a planilha que especifica este cenário de teste é dividida em três seções: pré-condições, processamento e resultados esperados.

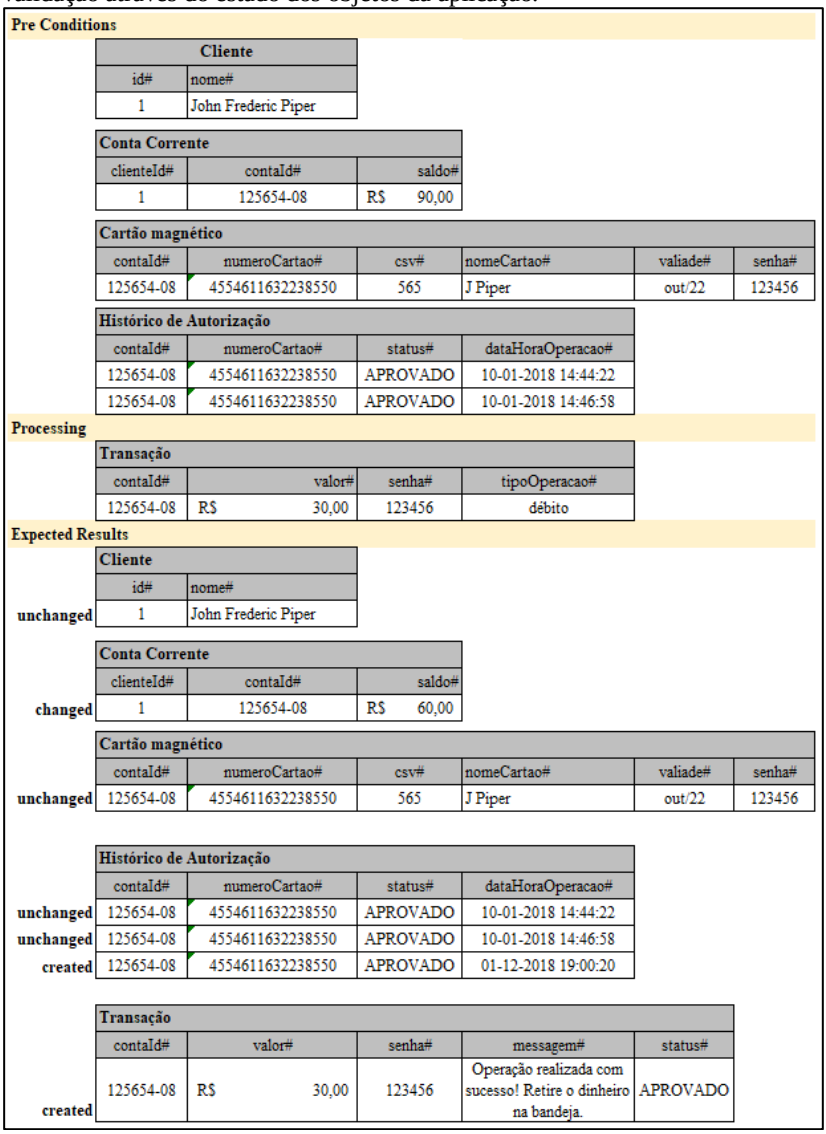
Na seção pré-condições (*Pre Conditions*) são definidos os estados dos objetos da aplicação antes da execução do processamento. Observe que, as entidades da aplicação (classes) são apresentadas através de tabelas e os dados (objetos) são as linhas destas tabelas (ANDREA, 2004).

A seção de processamento (*Processing*) mostra o impacto das ações executadas pelo usuário sobre os objetos do sistema em teste, por meio da entrada de dados através de uma ou mais entidades. No cenário de teste apresentado na Figura 30, a seção de processamento manipula somente um objeto de uma entidade, que é denominada Transação (ANDREA, 2004).

Por fim, a última seção mostra os resultados esperados (*Expected Results*). Nesta seção, são determinados os estados esperados dos objetos após o processamento. Um rótulo adicionado ao lado de cada linha das tabelas estabelece se um objeto não sofreu alterações (*unchanged*), sofreu alterações (*changed*) ou se foi criado (*created*) (ANDREA, 2004).

O APÊNDICE F – Exemplo de cenários de teste escritos utilizando planilhas eletrônicas com validação através do estado dos objetos da aplicação contém a versão completa com a especificação de todos os cenários de teste para os fluxos dos casos de uso detalhados no APÊNDICE C – Cenários de Casos de Uso.

Figura 30 – Exemplo de teste de aceitação utilizando planilha eletrônica com validação através do estado dos objetos da aplicação.



Pre Conditions

Cliente	
id#	nome#
1	John Frederic Piper

Conta Corrente		
clienteId#	contaId#	saldo#
1	125654-08	R$ 90,00

Cartão magnético					
contaId#	numeroCartao#	csv#	nomeCartao#	valiade#	senha#
125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autorização			
contaId#	numeroCartao#	status#	dataHoraOperacao#
125654-08	4554611632238550	APROVADO	10-01-2018 14:44:22
125654-08	4554611632238550	APROVADO	10-01-2018 14:46:58

Processing

Transação			
contaId#	valor#	senha#	tipoOperacao#
125654-08	R$ 30,00	123456	débito

Expected Results

	Cliente	
	id#	nome#
unchanged	1	John Frederic Piper

	Conta Corrente		
	clienteId#	contaId#	saldo#
changed	1	125654-08	R$ 60,00

	Cartão magnético					
	contaId#	numeroCartao#	csv#	nomeCartao#	valiade#	senha#
unchanged	125654-08	4554611632238550	565	J Piper	out/22	123456

	Histórico de Autorização			
	contaId#	numeroCartao#	status#	dataHoraOperacao#
unchanged	125654-08	4554611632238550	APROVADO	10-01-2018 14:44:22
unchanged	125654-08	4554611632238550	APROVADO	10-01-2018 14:46:58
created	125654-08	4554611632238550	APROVADO	01-12-2018 19:00:20

	Transação				
	contaId#	valor#	senha#	messagem#	status#
created	125654-08	R$ 30,00	123456	Operação realizada com sucesso! Retire o dinheiro na bandeja.	APROVADO

Fonte: Elaborada pelo autor (2019).

1.4 TABELAS COM LINGUAGEM DE TESTE DE DOMÍNIO ESPECÍFICO

Esta técnica é a combinação de uma Linguagem Específica de Teste de Domínio, do inglês, *Domain Test Specific Language* (DTSL), com planilhas eletrônicas, o que a torna muito similar às *Action Fixtures* do *Fit*, descritas na Seção 0. A Tabela 30 apresenta um exemplo de caso de teste utilizando esta técnica.

Tabela 30 – Exemplo de uso da técnica Tabelas com DTSL

Command	**Parameter1**	**Parameter2**
Open Form	sacar	
Edit Field	valor	30,00
Do Action	OK	
Check Field	valor	true
Edit Field	senha	123456
Do Action	confirmar	
Check Field	senha	true
Check Field	saldo	60.00

Fonte: Elaborada pelo autor (2019).

O exemplo na Tabela 30 é a especificação do teste de aceitação para o fluxo básico de eventos do caso de uso "Sacar dinheiro em um caixa eletrônico" descrito no APÊNDICE C – Cenários de Casos de Uso. Neste exemplo, a primeira coluna contém os comandos da DTSL, nas demais colunas estão os valores passados como parâmetro para este comando. Por padrão, são ofertados seis comandos:

- *Open form* (abrir formulário): recebe como parâmetro o nome da funcionalidade do sistema que será testada. Sendo assim, os comandos subsequentes serão ações direcionadas para esta funcionalidade (TALBY, NAKAR, *et al.*, 2005).
- *Edit field* (editar campo): este comando recebe dois parâmetros: o nome de campo e o valor a ser atribuído para este campo. O objetivo desta ação é realizar uma entrada em determinado campo do sistema (TALBY, NAKAR, *et al.*, 2005).
- *Do Action* (executar ação): recebe como parâmetro o método de um método/ação implementado no sistema em teste que deverá ser executado (TALBY, NAKAR, *et al.*, 2005).
- *Check field* (verificar campo): este comando recebe dois parâmetros e seu objetivo é realizar uma asserção. A verificação é realizada através da comparação do valor do campo no sistema informado no primeiro parâmetro com o valor esperado

informado no segundo parâmetro (TALBY, NAKAR, *et al.*, 2005).

- *Open Query* (abrir consulta): executa um método do sistema em teste que retorna uma tabela de dados. Esta tabela representa uma classe do sistema em teste e suas linhas são objetos (instâncias) desta classe (TALBY, NAKAR, *et al.*, 2005).
- *Select table line* (selecionar linha da tabela): seleciona um objeto – linha da tabela – retornado pelo do comando "*Open Query*" (TALBY, NAKAR, *et al.*, 2005).

O APÊNDICE G – Exemplos de cenários de teste escrito em Linguagem de Teste Domínio Específico estruturada em tabelas contém a especificação de todos os cenários de teste para os casos de uso especificado no APÊNDICE C – Cenários de Casos de Uso.

2 LINGUAGEM NATURAL

Algumas técnicas de testes de aceitação utilizam linguagem natural ou linguagens específicas de domínio (DSL's) para especificar os cenários de teste. Estas técnicas descrevem o comportamento esperado do sistema, detalhando suas entradas e saídas esperadas, através de uma linguagem muito similar da utilizada entre as pessoas no dia a dia para se comunicar. A fim de permitir a execução automatizada dos testes de aceitação, algumas destas técnicas requerem o uso de anotações, marcações ou formatos específicos de escrita dos testes.

Nos estudos selecionados foram identificadas quatro técnicas de teste de aceitação baseadas em linguagem natural, são elas:

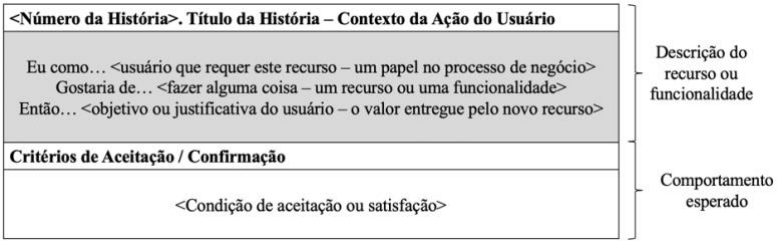
- T05 – Histórias de usuário com exemplos;
- T06 – Histórias de usuário com exemplos e marcações;
- T07 – Cenário de casos de uso com exemplos e anotações.
- T08 – BDD/*Gherkin language*.

2.1 HISTÓRIAS DE USUÁRIO COM EXEMPLOS

As histórias de usuário são descrições das funcionalidades de um sistema sob a perspectiva das pessoas que utilizarão estas funcionalidades. Há diversas formas de organizar e representar uma história de usuário, seja através de tópicos ordenados, parágrafos de texto ou ainda, cartões de história de usuário. A Figura 31 ilustra um exemplo de cartão de história de usuário composto pela descrição do recurso ou

funcionalidade e a descrição do comportamento esperado (COHN, 2004) (SOMMERVILLE, 2010).

Figura 31 – Exemplos de história de usuário



Fonte: Elaborada pelo autor (2019).

A área cinza da Figura 31, identificada pelo rótulo *descrição do recurso*, é usada para detalhar a funcionalidade usando um formato narrativo, enquanto que, a área identificada pelo rótulo *comportamento esperado*, descreve os critérios de aceitação através do detalhamento dos fluxos da interação entre o usuário e a aplicação ou da enumeração das condições de satisfação.

Originalmente o objetivo das histórias de usuário limitava-se a comunicar as necessidades dos usuários e os critérios de aceitação às equipes de desenvolvimento de software, todavia, o aprimoramento da descrição dos critérios de aceitação, seja pela adição de marcações ou pelo uso de linguagens específicas de domínio, permitiu a aplicação das histórias de usuário como artefato para especificar testes de aceitação automatizados (COHN, 2004) (MOKETAR, KAMALRUDIN, *et al.*, 2016).

Para utilizar histórias de usuário como cenários de teste de aceitação, cada história de usuário deve possuir pelo menos um exemplo de uso descrito na área Critérios de Aceitação / Confirmação (COHN, 2004). Este exemplo deverá detalhar, além da interação entre o usuário e o sistema, as entradas de dados e saídas esperadas. Se a história de usuário tiver outros fluxos além do básico, outros parágrafos exemplificando estes fluxos deverão ser adicionados aos critérios de aceitação. A Figura 32 ilustra um trecho de uma história de usuário para o caso de uso "Sacar dinheiro em um caixa eletrônico", detalhado no APÊNDICE C – Cenários de Casos de Uso.

O APÊNDICE H – Exemplos de cenários de teste escritos utilizando Histórias de Usuário contém a especificação de todos os cenários de teste

de aceitação para os fluxos dos casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

Figura 32 – Trecho de uma história de usuário e seus critérios de aceitação

001. Sacar dinheiro – Caixa Eletrônico
Eu como cliente de um banco **Gostaria de** sacar dinheiro da minha conta corrente **Então** eu vou até um caixa eletrônico, realizo a identificação através de cartão magnético e senha, escolho o valor desejado e realizo o saque
Critérios de Aceitação / Confirmação
J. F. Piper é cliente do The American Bank. Ele tem uma conta corrente número **125654-08** e o saldo da conta é **R$ 90,00**. Piper vai a um caixa eletrônico e se identifica com o cartão da conta corrente número **8888777766665555**. A senha deste cartão é **123456**. O caixa eletrônico tem, disponível para saque, a seguinte quantidade de notas: 200 notas de R$ 10, 0 notas de R$ 20, 150 notas de R$ 50 e 110 notas de R$ 100. Critério de aceitação 01: Piper escolhe a opção ***Saque***. Ele informa que deseja sacar **R$ 30,00.** O caixa eletrônico pede a senha do cartão. Piper informa a senha **123456** e confirma a operação. Piper retira o cartão do caixa eletrônico. O caixa eletrônico mostra a mensagem: **"A transação foi completada com sucesso!"**. O caixa eletrônico entrega o dinheiro na bandeja. O saldo da conta corrente de Piper atualizado deverá ser **R$ 60,00.** Critério de aceitação 02: **...**

Fonte: Elaborada pelo autor (2019).

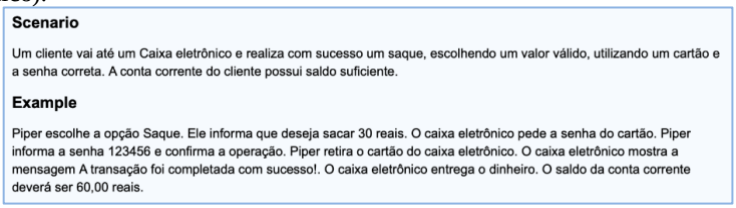
2.2 HISTÓRIAS DE USUÁRIO COM EXEMPLOS E MARCAÇÕES

Sob a perspectiva do usuário, a técnica Histórias de Usuário com exemplos e marcações utiliza a mesma notação da técnica detalhada na seção 2.1. No entanto, nesta técnica, são adicionas às histórias de usuário marcações utilizando *HyperText Markup Language* (HTML) ou *eXtensible Markup Language* (XML) são adicionadas às histórias de usuário. As marcações não são visíveis aos usuários não-técnicos, mantendo a leitura das especificações amigáveis, todavia, estas marcações permitem a automação da execução dos testes. As Figura 33 mostra um exemplo desta técnica do ponto de vista do usuário, enquanto a Figura 34 mostra o trecho equivalente com as marcações em HTML.

No exemplo ilustrado na Figura 34, as marcações adicionadas às histórias de usuário são compatíveis como uma ferramenta de execução automatizada de testes de aceitação denominada Concordion. Nas *tags* do

documento HTML são inseridos comandos que refletem ações na camada de negócio do sistema em teste. Por exemplo, o comando *set* altera o valor de um atributo de um objeto, o comando *execute* executa um método e o comando *assert-equals* verifica se o retorno de um método é igual ao esperado.

Figura 33 – Trecho de uma história de usuário com marcações (visão não-técnico).



Fonte: Elaborada pelo autor.

O APÊNDICE I – Exemplos de cenários de teste escritos utilizando Histórias de Usuário com anotações contém a especificação de todos os cenários de teste para os fluxos dos casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

Figura 34 - Trecho de uma história de usuário com marcações (visão técnico).

```
<h3>Example</h3><p>
J.F. Piper escolhe a opção Saque.
Ele informa que deseja sacar
      <span concordion:set="#amount">30</span> reais.
O caixa eletrônico pede a senha do cartão.
Piper informa a senha
      <span concordion:set="#pin">123456</span>
e confirma a operação.
Piper retira o cartão do caixa eletrônico.
O caixa eletrônico mostra a mensagem:
      <span concordion:assert-equals="withdraw(#bankCardNumber,#pin,#amount)">
            A transação foi completada com sucesso!
      </span>.
O caixa eletrônico entrega o dinheiro.
O saldo da conta corrente deverá ser
      <span concordion:assert-equals="getBalance(#bankCardNumber)">
            60,00
      </span> reais.
</p>
```

Fonte: Elaborada pelo autor (2019).

2.3 GHERKIN LANGUAGE

A *Gherkin language* é uma linguagem específica de domínio para escrever testes de aceitação. Os cenários de teste são escritos com base

em um conjunto de cláusulas, cujo objetivo é estabelecer uma estrutura para descrever o comportamento esperado do sistema de forma padronizada (CUCUMBER.IO, 2019). As principais cláusulas desta linguagem são:

- **Given:** define as pré-condições para realização de um teste;
- **When:** define as ações executadas pelo usuário;
- **Then:** determina o resultado esperado a partir das ações executadas pelo usuário.

A Figura 35 ilustra a estrutura básica dos cenários especificados utilizado *Gherkin language.* Apesar das cláusulas *Given*, *When* e *Then* estarem representadas na língua inglesa, há pacotes da *Gherkin language* que possibilitam expressar estes mesmos comandos em outros idiomas, deixando a leitura dos cenários mais natural para não falantes do inglês e sem prejuízo a automação da execução dos testes (CUCUMBER.IO, 2019).

Figura 35 - Estrutura básica de um cenário de teste de aceitação especificado através da *Ghekin language.*

given	(dado)	uma determinada situação
when	(quando)	o usuário executar uma ou mais ações;
then	(então)	o sistema deverá apresentar determinados resultados ou realizar determinadas validações.

Fonte: Elaborada pelo autor (2019).

Figura 36 - Exemplo de cenário e caso de teste de aceitação utilizando *Gherkin language*.

```
Scenario: Um cliente vai até um caixa eletrônico e realiza com sucesso um
saque, escolhendo um valor válido, utilizando um cartão e a senha correta. A
conta corrente do cliente possui saldo suficiente.

Given o cliente 'J. F.Piper'
 And o número da conta corrente do cliente é '125654-08'
 And o número do cartão da conta corrente é '9988776655443322'
 And a senha do cartão da conta corrente é '123456'
 And o saldo da conta corrente do cliente é $90.00
 And o cliente escolhe a opção 'Sacar'
 And no caixa eletrônico há 200 notas 10
 And 0 notas de 20
 And 150 notas 50
 And 110 notas 100
 And o cliente não errou a senha do cartão nas últimas 72 horas
When o cliente informar o valor para sacar de $30.00
 And informar a senha '123456'
 And pressionar a tecla 'Confirma'
Then o caixa eletrônico disponibiliza o dinheiro na bandeja
 And o saldo atualizado da conta correte do cliente é de $60.00
```

Fonte: Elaborada pelo autor (2019).

A Figura 36 apresenta um exemplo de cenário de teste aceitação escrito em *Gherkin language* referente ao fluxo básico de eventos do caso

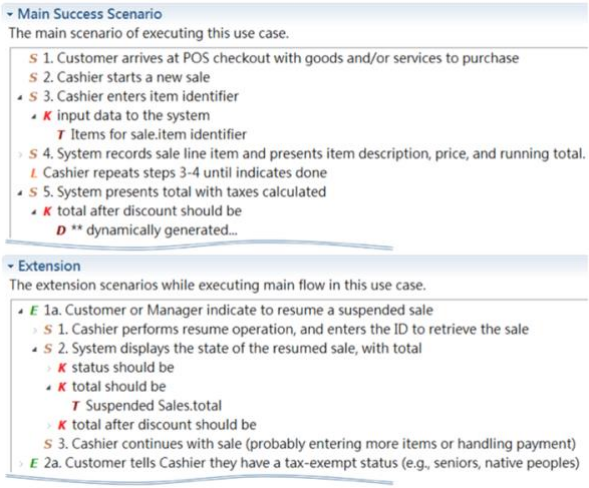
de uso “Sacar dinheiro em um caixa eletrônico” especificado no APÊNDICE C – Cenários de Casos de Uso. Embora não ilustrado na Figura 36, pois esta mostra apenas um trecho de toda a especificação do teste, os cenários de teste de aceitação utilizando *Gherkin language* são acompanhados das respectivas histórias de usuário, da mesma forma como foi apresentado nas técnicas abordadas nas seções 2.1 e 2.2.

O APÊNDICE J – Exemplos de cenários de teste escritos utilizando *Gherkin Language* contém uma versão completa de todos os testes de aceitação para os fluxos dos casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

2.4 CASOS DE USO COM EXEMPLOS E ANOTAÇÕES

Os fluxos de casos de uso descrevem a interação entre o ator – usuário do sistema – e o sistema (SOMMERVILLE, 2010). Os casos de uso com exemplos de entrada de dados e anotações são uma técnica de teste de aceitação que consiste na inclusão de anotações aos fluxos especificados tornando-os legíveis por ferramentas de automação de testes, como o ROBOT. Além disso, se combinados com a associação de exemplos de dados de entrada, é possível executar os fluxos dos casos de uso como testes de aceitação automatizados (HSIEH, TSAI e C., 2013).

Figura 37 - Caso de uso "Processo de vendas" com as anotações



Fonte: (HSIEH, TSAI e C., 2013)

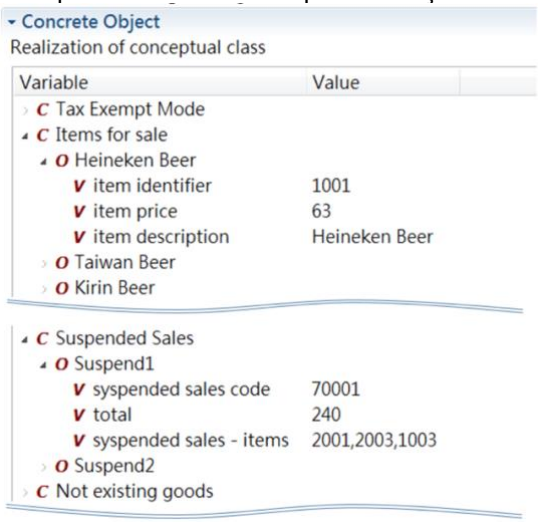
A Figura 37 ilustra o caso de uso “Processo de vendas”. A este caso de uso, utilizando um *plug-in* do Eclipse denominado *Acceplipse*, foram

adicionadas anotações. Estas anotações expõem o caso de uso de forma que se torna possível a sua execução por uma ferramenta denominada ROBOT (HSIEH, TSAI e C., 2013).

A Seção "*Main Success Scenario*", ilustrada na Figura 37, mostra as etapas do fluxo básico de ações de caso de uso, enquanto a seção "*Extension*" mostra os fluxos alternativos. Cada passo no caso de uso é marcado com um S e o loop de etapas é marcado com um L. Os itens marcados com K são as palavras-chave de teste. Os itens marcados com a letra T são argumentos cujo valor é determinado antes do caso de teste ser executado. Os itens indicados pela letra D são argumento dinâmicos e terão seus valores calculados em tempo de execução (HSIEH, TSAI e C., 2013).

Para o caso de uso "Processo de vendas", o conjunto de dados criados são mostrados na Figura 38. Cada classe conceitual no conjunto de dados é marcada com a letra C. Dentro de cada classe conceitual estão os objetos concretos indicados pela letra O. Recuados sob os objetos concretos estão os valores dos atributos (marcados com a letra V), estes valores serão utilizados como exemplos de entrada para a execução dos testes de aceitação (HSIEH, TSAI e C., 2013).

Figura 38 - Exemplos de dados criados para execução dos testes



Fonte: (HSIEH, TSAI e C., 2013)

Esta técnica não foi aplicada a nenhum dos casos de uso propostos no APÊNDICE C – Cenários de Casos de Uso, pois o *plug-in Acceplipse* não está disponível para *download.*

3 LINGUAGEM DE PROGRAMAÇÃO

Nos estudos selecionados foi encontrada somente uma técnica de teste de aceitação que utiliza uma linguagem de programação para especificação dos cenários de teste, o *EasyAccept*.

3.1 EASYACCEPT

O *EasyAccept* é uma técnica de teste de aceitação que utiliza *scripts* escritos em um arquivo de texto para expressar os testes. Os *scripts* são formados por um conjunto de comandos, detalhados na Tabela 31. Cada comandos deve ocupar uma linha do arquivo. Os comandos devem ser lidos de cima para baixo, no mesmo sentido que são executados pelo motor de testes (SAUVÉ, ABATH NETO e CIRNE, 2006).

Tabela 31 – Comando internos do *EasyAccept*

Comando	**Descrição**
`stringDelimiter`	Altera o delimitador de *string* para um delimitador especificado. Por padrão, o delimitador de cadeia é “.
`expect`	Verifica se um método de negócio do sistema em teste produz o resultado esperado.
`expectDifferent`	Verifica se um método de negócio do sistema em teste produz um resultado diferente de uma determinada *string*.
`expectWithin`	Verifica se um método de negócio do sistema em teste produz o resultado esperado dentro de uma precisão desejada.
`expectError`	Verifica se um método de negócios do sistema retorna um erro ou exceção em determinadas condições.
`equalFiles`	Compara o conteúdo de dois arquivos.
`stackTrace`	Obtém os dados da pilha de execução da aplicação durante a depuração.
`quit`	Finaliza a execução do *script*.
`executeScript`	Executa um determinado *script*. A execução do *script* atual é pausada até que o novo *script* seja executado até a conclusão.
`repeat`	Repete um determinado comando pelo número de vezes que for especificado.
`threadPool`	Cria um thread *pool* para executar os *scripts*.
`echo`	Retorna os parâmetros do *script* concatenados.

Fonte: Elaborada pelo autor (2019).

A Figura 39 mostra um exemplo de *script* criado usando o *EasyAccept*. A primeira linha do *script* é um comentário, pois inicia com o símbolo “#”. As demais linhas são comandos. A segunda linha invoca o método de negócio `createCustomer` passando quatro parâmetros. A terceira linha faz uma asserção verificando se o método

`getCustomerNameByAccountNumber` retorna o valor “J. F. Piper” quando receber o parâmetro `accountNumber` igual a “125654-08”.

Figura 39 - Exemplo de caso de teste utilizando o *EasyAccept*

```
#EasyAccept - create a customer
createCustomer name="J. F. Piper" accountNumber="125654-08" balance=90.00 pin="123456"
expected "J. F. Piper" getCustomerNameByAccountNumber accountNumber="125654-08"
```

Fonte: Elaborada pelo autor (2019).

O APÊNDICE K – Exemplos de cenários de teste escritos utilizando o *EasyAccept* contém uma versão completa de todos os testes de aceitação para os fluxos dos casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

4 DIAGRAMAS

As técnicas de testes de aceitação classificadas como Diagramas são aquelas que representam os testes através de textos e figuras geométricas. Nos estudos selecionados foram identificadas duas técnicas cujo os cenários de teste são especificados por meio de diagramas, são elas:

- T10 – Cenários de usuário através de diagramas de interação do usuário (US-UID); e
- T11 – Máquina de Estados Finita.

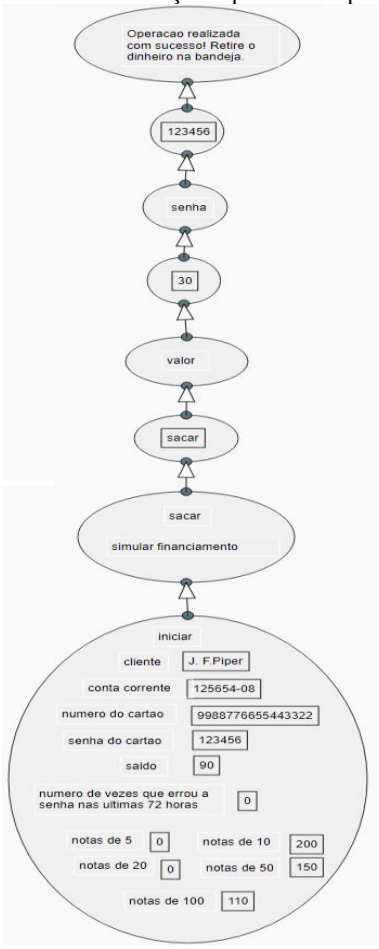
4.1 CENÁRIOS DE USUÁRIO ATRAVÉS DE DIAGRAMAS DE INTERAÇÃO DO USUÁRIO (US-UID)

US-UID é a sigla para *User Scenarios through User Interactions Diagrams* ou, em português, Cenários de Usuário através de Diagramas de Interação do Usuário (LONGO e VILAIN, 2015). Os US-UIDs são compostos por quatro elementos:

- **Elipses:** representam os estados da interação entre o usuário e o sistema (LONGO e VILAIN, 2015);
- **Setas:** representam as transições entre cada um dos estados de interação (LONGO e VILAIN, 2015);
- **Retângulos**: representam as entradas do usuário (LONGO e VILAIN, 2015);
- **Textos sem bordas:** representam as saídas do sistema (LONGO e VILAIN, 2015).

A Figura 40 ilustra um diagrama US-UID para testar o fluxo básico de eventos da funcionalidade “Sacar dinheiro em um caixa eletrônico” especificada no APÊNDICE C – Cenários de Casos de Uso. Para cada elemento do diagrama, exceto as transições, é necessário definir um valor para a propriedade “*Model*”. Esta propriedade é visível somente através da ferramenta de edição dos diagramas US-UID, a Scenario, e será utilizada pelo código de cola para conectar os diagramas US-UIDs com o código-fonte da aplicação em teste, possibilitando, desta forma, a execução de testes automatizados.

Figura 40 - Exemplo de teste de aceitação representado por US-UID



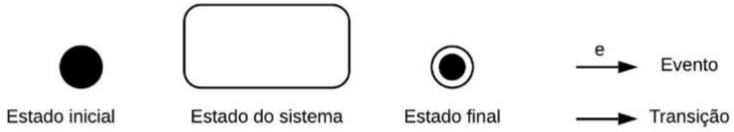
Fonte: Elaborada pelo autor (2019).

O APÊNDICE L – Exemplos de cenários de teste escritos utilizando US-UIDAPÊNDICE L – Exemplos de cenários de teste escritos utilizando US-UID contém uma versão completa de todos os testes de aceitação para os fluxos dos casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

4.2 MÁQUINA DE ESTADOS FINITA

Esta técnica consiste na representação do comportamento esperado do sistema através de máquinas de estados finitas. Uma máquina de estados finita é uma representação gráfica formal de um sistema e é formada de cinco elementos: um estado inicial, um estado final, um conjunto de estados do sistema, um conjunto de transições e um conjunto de eventos (HSIA, KUNG e SELL, 1997). A representação gráfica destes elementos está ilustrada na Figura 41.

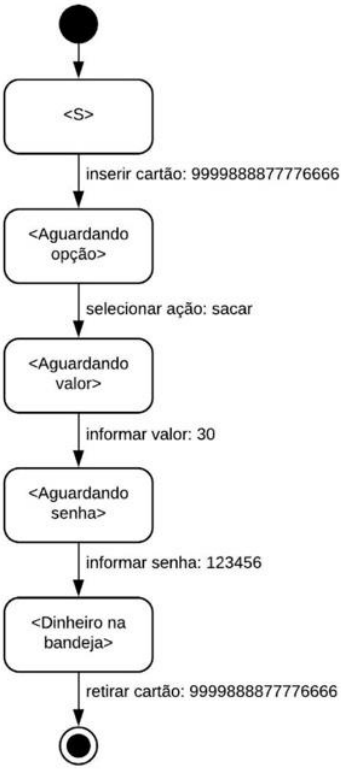
Figura 41 - Elementos das máquinas de estados finita



Fonte: Elaborada pelo autor (2019), adaptado de (HSIA, KUNG e SELL, 1997).

A Figura 42 apresenta um exemplo de teste de aceitação para o fluxo básico de eventos do caso de uso “Sacar dinheiro em um caixa eletrônico” especificado no APÊNDICE C – Cenários de Casos de Uso. Nesta figura, os retângulos representam os estados do caixa eletrônico, enquanto as transições são as ações do usuário. Ao lado das transições são colocados exemplos de entrada de dados.

Figura 42 – Exemplo de caso de teste de aceitação utilizando Máquina de Estados Finita.



Fonte: Elaborada pelo autor (2019).

O APÊNDICE M – Exemplos de cenários de teste escritos utilizando Máquina de Estados Finita contém uma versão completa de todos os testes de aceitação para os fluxos dos casos de uso especificados no APÊNDICE C – Cenários de Casos de Uso.

APÊNDICE E – Exemplo de cenários de teste escritos utilizando Fit Tables

A - História de Usuário:
Eu como cliente de um banco;
Quero retirar dinheiro da minha conta corrente;
Então realizo um saque utilizando um Caixa Eletrônico.

Cenário de Aceitação 1: O saque é realizado com sucesso.

br.ufsc.acceptanceTests.SacarDinheiroSetUpFixture					
usuário	**conta corrente**	**saldo**	**número do cartão**	**senha**	**notas (nota, quantidade)**
J.F. Piper	125654-08	90,00	9999 8888 7777 6666 5555	123456	{(5,0), (10,200), (20,0), (50,150), (100,100)}

start	br.ufsc.acceptanceTests.SacarDinheiroActionFixture	
enter	número do cartão	9999 8888 7777 6666 5555
enter	valor para sacar	30
press	OK	
check	o valor é valido	verdadeiro
enter	senha	123456
check	confirmar	Operação realizada com sucesso!
check	saldo atual	60,00

Cenário de Aceitação 2: O saque não é realizado, pois o usuário informou um valor inválido.

br.ufsc.acceptanceTests.SacarDinheiroSetUpFixture					
usuário	**conta corrente**	**saldo**	**número do cartão**	**senha**	**notas (nota, quantidade)**

J.F. Piper	125654-08	90,00	9999 8888 7777 6666 5555	123456	{(5,0), (10,200), (20,0), (50,150), (100,100)}

start	br.ufsc.acceptanceTests.SacarDinheiroActionFixture	
enter	número do cartão	9999 8888 7777 6666 5555
enter	valor para sacar	31
press	OK	
check	o valor é válido	falso
check	saldo atual	90,00

Cenário de Aceitação 3: O saque não é realizado, pois o saldo da conta do usuário é insuficiente.

br.ufsc.acceptanceTests.SacarDinheiroSetUpFixture					
usuário	**conta corrente**	**saldo**	**número do cartão**	**senha**	**notas (nota, quantidade)**
J.F. Piper	125654-08	90,00	9999 8888 7777 6666 5555	123456	{(5,0), (10,200), (20,0), (50,150), (100,100)}

start	br.ufsc.acceptanceTests.SacarDinheiroActionFixture	
enter	número do cartão	9999 8888 7777 6666 5555
enter	valor para sacar	100
press	OK	
check	o valor é valido	verdadeiro
enter	senha	123456
check	confirmar	Saldo insuficiente!
Check	saldo atual	90,00

Cenário de Aceitação 4: O saque não é realizado, pois o usuário informou uma senha inválida. É a primeira vez que o usuário informa uma senha inválida nas últimas 72 horas.

br.ufsc.acceptanceTests.SacarDinheiroSetUpFixture					
usuário	**conta corrente**	**saldo**	**número do cartão**	**senha**	**notas (nota, quantidade)**
J.F. Piper	125654-08	90,00	9999 8888 7777 6666 5555	123456	{(5,0), (10,200), (20,0), (50,150), (100,100)}

start	br.ufsc.acceptanceTests.SacarDinheiroActionFixture	
enter	número do cartão	9999 8888 7777 6666 5555
enter	valor para sacar	30
press	OK	
check	o valor é valido	verdadeiro
enter	senha	654321
check	confirmar	Senha inválida!
check	saldo atual	90,00

Cenário de Aceitação 5: O saque não é realizado, pois o usuário informou uma senha inválida. É a segunda vez que o usuário informa uma senha inválida nas últimas 72 horas.

br.ufsc.acceptanceTests.SacarDinheiroSetUpFixture					
usuário	**conta corrente**	**saldo**	**número do cartão**	**senha**	**notas (nota, quantidade)**
J.F. Piper	125654-08	90,00	9999 8888 7777 6666 5555	123456	{(5,0), (10,200), (20,0), (50,150), (100,100)}

br.ufsc.acceptanceTests.HistoricoAutenticacaoSetupFixture		
número do cartão	**número da ocorrência**	**número de horas atrás que informou a senha do cartão errada**
9999 8888 7777 6666 5555	1	2

start	br.ufsc.acceptanceTests.SacarDinheiroActionFixture	
enter	número do cartão	9999 8888 7777 6666 5555
enter	valor para sacar	30
press	OK	
check	o valor é valido	verdadeiro
enter	senha	654321
check	confirmar	Senha inválida! Você tem mais uma tentativa nas próximas 70 horas, do contrário, seu cartão será bloqueado por motivos de segurança.
check	saldo atual	90,00

Cenário de Aceitação 6: O saque não é realizado, pois o usuário informou uma senha inválida. É a terceira vez que o usuário informa uma senha inválida nas últimas 72 horas.

br.ufsc.acceptanceTests.SacarDinheiroSetUpFixture					
usuário	**conta corrente**	**saldo**	**número do cartão**	**senha**	**notas (nota, quantidade)**
J.F. Piper	125654-08	90,00	9999 8888 7777 6666 5555	123456	{(5,0), (10,200), (20,0), (50,150), (100,100)}

br.ufsc.acceptanceTests.HistoricoAutenticacaoSetupFixture		
número do cartão	**número da ocorrência**	**número de horas atrás que informou a senha do cartão errada**
99998888777766665555	1	2,1
99998888777766665555	2	2

start	br.ufsc.acceptanceTests.SacarDinheiroActionFixture	
enter	número do cartão	99998888777766665555
enter	valor para sacar	30
press	OK	
check	o valor é valido	verdadeiro

enter	senha	654321
check	confirmar	Senha inválida! Seu cartão foi bloqueado por motivos de segurança! Entre em contato com a central de serviços para maiores informações.
check	Saldo atual	90.00

Cenário de Aceitação 7: O saque não é realizado, pois o cartão da conta corrente do usuário está bloqueado.

br.ufsc.acceptanceTests.SacarDinheiroSetUpFixture					
usuário	**conta corrente**	**saldo**	**número do cartão**	**senha**	**notas (nota, quantidade)**
J.F. Piper	125654-08	90,00	9999 8888 7777 6666 5555	123456	{(5,0), (10,200), (20,0), (50,150), (100,100)}

br.ufsc.acceptanceTests.HistoricoAutenticacaoSetupFixture		
número do cartão	**número da ocorrência**	**número de horas atrás que informou a senha do cartão errada**
99998888777766665555	1	2,2
99998888777766665555	2	2,1
99998888777766665555	3	2

start	br.ufsc.acceptanceTests.SacarDinheiroActionFixture	
enter	número do cartão	9999 8888 7777 6666 5555
enter	valor para sacar	30
press	OK	
check	o valor é valido	verdadeiro
enter	senha	654321
check	confirmar	Este cartão está bloqueado!
check	saldo atual	90.00

B - História de Usuário:
Eu como cliente de um banco;
Quero calcular o valor das parcelas mensais de um financiamento;
Então informo os valores para o sistema e obtenho os resultados.

Cenário de Aceitação: A sigla EMI refere-se ao valor das parcelas mensais de um empréstimo pagas por um cliente em determinada data ao banco. O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa:

- O valor total a financiar;
- A taxa de juros (em percentual);
- O número de parcelas que deseja pagar.
- O sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar.

A fórmula matemática para o cálculo do EMI é:

$EMI = P \times R \times \left[\frac{(1+R)^N}{(1+R)^N-1}\right]$, onde: P é o valor total a financiar, R é a taxa de juros e N é o número de parcelas mensais.

br.ufsc.acceptanceTests.SimularFinanciamento ColumnFixture				
valor a financiar	**taxa de juros**	**número de parcelas**	**valor das parcelas mensais?**	**valor total a pagar com juros?**
10000,00	0,5334	12	862,51	10350,09
100000000,00	0,3227	36	2946725,65	106082123,23
500,00	0,1250	5	100,38	501,88
3250500,00	0,7665	72	58911,94	4241659,83

APÊNDICE F – Exemplo de cenários de teste escritos utilizando planilhas eletrônicas com validação através do estado dos objetos da aplicação

A - História de Usuário:
Eu como cliente de um banco;
Quero retirar dinheiro da minha conta corrente;
Então realizo um saque utilizando um Caixa Eletrônico.

Cenário de Aceitação 1: O saque é realizado com sucesso.

Pre Conditions

Cliente

id#	nome#
1	John Frederic Piper

Conta Corrrente

clienteId#	contaId#	saldo
1	125654-08	R$ 90,00

Cartão Magnético

contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação

accountId#	cardNumber#	dateTime#	status#
125654-08	4554611632238550	1/10/18 14:44	APROVADA
125654-08	4554611632238550	1/10/18 14:46	APROVADA

Processing

Transação

contaId#	valor#	senha#	tipo#
125654-08	R$ 30,00	123456	Débito

Expected Results

Cliente

	id#	nome#
unchanged	1	John Frederic Piper

Conta Corrrente

	clienteId#	contaId#	saldo
changed	1	125654-08	R$ 60,00

Cartão Magnético

	contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
unchanged	125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação

	accountId#	cardNumber#	dateTime#	status#
unchanged	125654-08	4554611632238550	1/10/18 14:44	APROVADA
unchanged	125654-08	4554611632238550	1/10/18 14:46	APROVADA
created	125654-08	4554611632238550	29/1/18 14:24	APROVADA

Transação

	contaId#	valor#	senha#	tipo#	mensagem	status#
created	125654-08	R$ 30,00	123456	Débito	Operação realizada com sucesso! Retire o dinheiro na bandeja.	APROVADA

Cenário de Aceitação 2: O saque não é realizado, pois o usuário informou um valor inválido.

Pre Conditions

Cliente

id#	nome#
1	John Frederic Piper

Conta Corrrente

clienteId#	contaId#	saldo
1	125654-08	R$ 90,00

Cartão Magnético

contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação

accountId#	cardNumber#	dateTime#	status#
125654-08	4554611632238550	1/10/18 14:44	APROVADA
125654-08	4554611632238550	1/10/18 14:46	APROVADA

Processing

Transação

contaId#	valor#	senha#	tipo#
125654-08	R$ 31,00	-	Débito

Expected Results

Cliente

	id#	nome#
unchanged	1	John Frederic Piper

Conta Corrrente

	clienteId#	contaId#	saldo
unchanged	1	125654-08	R$ 90,00

Cartão Magnético

	contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
unchanged	125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação

	accountId#	cardNumber#	dateTime#	status#
unchanged	125654-08	4554611632238550	1/10/18 14:44	APROVADA
unchanged	125654-08	4554611632238550	1/10/18 14:46	APROVADA

Transação

	contaId#	valor#	senha#	tipo#	mensagem	status#
created	125654-08	R$ 31,00	-	Débito	O valor informádo é inválido!	ABORTADA

Cenário de Aceitação 3: O saque não é realizado, pois o saldo da conta do usuário é insuficiente.

Pre Conditions

Cliente

id#	nome#
1	John Frederic Piper

Conta Corrrente

clienteId#	contaId#	saldo
1	125654-08	R$ 90,00

Cartão Magnético

contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação

accountId#	cardNumber#	dateTime#	status#
125654-08	4554611632238550	1/10/18 14:44	APROVADA
125654-08	4554611632238550	1/10/18 14:46	APROVADA

Processing

Transação

contaId#	valor#	senha#	tipo#
125654-08	R$ 100,00	123456	Débito

Expected Results

Cliente

	id#	nome#
unchanged	1	John Frederic Piper

Conta Corrrente

	clienteId#	contaId#	saldo
unchanged	1	125654-08	R$ 90,00

Cartão Magnético

	contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
unchanged	125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação

	accountId#	cardNumber#	dateTime#	status#
unchanged	125654-08	4554611632238550	1/10/18 14:44	APROVADA
unchanged	125654-08	4554611632238550	1/10/18 14:46	APROVADA
created	125654-08	4554611632238550	29/1/18 14:24	APROVADA

Transação

	contaId#	valor#	senha#	tipo#	mensagem	status#
created	125654-08	R$ 100,00	123456	Débito	Saldo insuficiente!	APROVADA

Cenário de Aceitação 4: O saque não é realizado, pois o usuário informou uma senha inválida. É a primeira vez que o usuário informa uma senha inválida nas últimas 72 horas.

Pre Conditions

Cliente	
id#	nome#
1	John Frederic Piper

Conta Corrrente		
clienteId#	contaId#	saldo
1	125654-08	R$ 90,00

Cartão Magnético					
contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação			
accountId#	cardNumber#	dateTime#	status#
125654-08	4554611632238550	1/10/18 14:44	APROVADA
125654-08	4554611632238550	1/10/18 14:46	APROVADA

Processing

Transação			
contaId#	valor#	senha#	tipo#
125654-08	R$ 30,00	654321	Débito

Expected Results

	Cliente	
	id#	nome#
unchanged	1	John Frederic Piper

	Conta Corrrente		
	clienteId#	contaId#	saldo
unchanged	1	125654-08	R$ 90,00

	Cartão Magnético					
	contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
unchanged	125654-08	4554611632238550	565	J Piper	out/22	123456

	Histórico de Autenticação			
	accountId#	cardNumber#	dateTime#	status#
unchanged	125654-08	4554611632238550	1/10/18 14:44	APROVADA
unchanged	125654-08	4554611632238550	1/10/18 14:46	APROVADA
created	125654-08	4554611632238550	1/10/18 15:00	NEGADA

	Transação					
	contaId#	valor#	senha#	tipo#	mensagem	status#
created	125654-08	R$ 30,00	654321	Débito	Senha incorreta!	NEGADA

Cenário de Aceitação 5: O saque não é realizado, pois o usuário informou uma senha inválida. É a segunda vez que o usuário informa uma senha inválida nas últimas 72 horas.

Pre Conditions

Cliente

id#	nome#
1	John Frederic Piper

Conta Corrente

clienteId#	contaId#	saldo
1	125654-08	R$ 90,00

Cartão Magnético

contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação

accountId#	cardNumber#	dateTime#	status#
125654-08	4554611632238550	1/10/18 14:44	APROVADA
125654-08	4554611632238550	1/10/18 14:46	APROVADA
125654-08	4554611632238550	1/10/18 15:00	NEGADA

Processing

Transação

contaId#	valor#	senha#	tipo#
125654-08	R$ 30,00	654321	Débito

Expected Results

Cliente

	id#	nome#
unchanged	1	John Frederic Piper

Conta Corrente

	clienteId#	contaId#	saldo
unchanged	1	125654-08	R$ 90,00

Cartão Magnético

	contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
unchanged	125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação

	accountId#	cardNumber#	dateTime#	status#
unchanged	125654-08	4554611632238550	1/10/18 14:44	APROVADA
unchanged	125654-08	4554611632238550	1/10/18 14:46	APROVADA
unchanged	125654-08	4554611632238550	1/10/18 15:00	NEGADA
created	125654-08	4554611632238550	1/10/18 15:04	NEGADA

Transação

	contaId#	valor#	senha#	tipo#	mensagem	status#
created	125654-08	R$ 30,00	654321	Débito	Senha incorreta! Você tem mais uma tentativa! Caso informe a senha errada novamente seu cartão será bloqueado.	NEGADA

Cenário de Aceitação 6: O saque não é realizado, pois o usuário informou uma senha inválida. É a terceira vez que o usuário informa uma senha inválida nas últimas 72 horas.

Pre Conditions

Cliente	
id#	nome#
1	John Frederic Piper

Conta Corrrente		
clienteId#	contaId#	saldo
1	125654-08	R$ 90,00

Cartão Magnético					
contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação			
accountId#	cardNumber#	dateTime#	status#
125654-08	4554611632238550	1/10/18 14:44	APROVADA
125654-08	4554611632238550	1/10/18 14:46	NEGADA
125654-08	4554611632238550	1/10/18 15:00	NEGADA

Processing

Transação			
contaId#	valor#	senha#	tipo#
125654-08	R$ 30,00	654321	Débito

Expected Results

	Cliente	
	id#	nome#
unchanged	1	John Frederic Piper

	Conta Corrrente		
	clienteId#	contaId#	saldo
unchanged	1	125654-08	R$ 90,00

	Cartão Magnético					
	contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
unchanged	125654-08	4554611632238550	565	J Piper	out/22	123456

	Histórico de Autenticação			
	accountId#	cardNumber#	dateTime#	status#
unchanged	125654-08	4554611632238550	1/10/18 14:44	APROVADA
unchanged	125654-08	4554611632238550	1/10/18 14:46	NEGADA
unchanged	125654-08	4554611632238550	1/10/18 15:00	NEGADA
created	125654-08	4554611632238550	1/10/18 15:04	NEGADA

	Transação						
	contaId#	valor#	senha#	tipo#		mensagem	status#
created	125654-08	R$ 30,00	654321	Débito		Cartão bloqueado!	NEGADA

Cenário de Aceitação 7: O saque não é realizado, pois o cartão da conta corrente do usuário está bloqueado.

Pre Conditions

Cliente	
id#	nome#
1	John Frederic Piper

Conta Corrrente		
clienteId#	contaId#	saldo
1	125654-08	R$ 90,00

Cartão Magnético					
contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
125654-08	4554611632238550	565	J Piper	out/22	123456

Histórico de Autenticação			
accountId#	cardNumber#	dateTime#	status#
125654-08	4554611632238550	1/10/18 14:44	NEGADA
125654-08	4554611632238550	1/10/18 14:46	NEGADA
125654-08	4554611632238550	1/10/18 15:00	NEGADA

Processing

Transação			
contaId#	valor#	senha#	tipo#
125654-08	R$ 30,00	123456	Débito

Expected Results

	Cliente	
	id#	nome#
unchanged	1	John Frederic Piper

	Conta Corrrente		
	clienteId#	contaId#	saldo
unchanged	1	125654-08	R$ 90,00

	Cartão Magnético					
	contaId#	numeroCartao#	csv#	nomeCartao#	validade#	senha#
unchanged	125654-08	4554611632238550	565	J Piper	out/22	123456

	Histórico de Autenticação			
	accountId#	cardNumber#	dateTime#	status#
unchanged	125654-08	4554611632238550	1/10/18 14:44	NEGADA
unchanged	125654-08	4554611632238550	1/10/18 14:46	NEGADA
unchanged	125654-08	4554611632238550	1/10/18 15:00	NEGADA
created	125654-08	4554611632238550	1/10/18 15:04	NEGADA

	Transação					
	contaId#	valor#	senha#	tipo#	mensagem	status#
created	125654-08	R$ 30,00	654321	Débito	Seu cartão está bloqueado!	NEGADA

B - História de Usuário:
Eu como cliente de um banco;
Quero calcular o valor das parcelas mensais de um financiamento;
Então informo os valores para o sistema e obtenho os resultados.

Cenário de Aceitação: A sigla EMI refere-se ao valor das parcelas mensais de um empréstimo pagas por um cliente em determinada data ao banco. O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa:

- O valor total a financiar;
- A taxa de juros (em percentual);
- O número de parcelas que deseja pagar.
- O sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar.

A fórmula matemática para o cálculo do EMI é:

$EMI = P \times R \times \left[\frac{(1+R)^N}{(1+R)^N-1}\right]$, onde: P é o valor total a financiar, R é a taxa de juros e N é o número de parcelas mensais.

Pre Conditions

Processing

	Financiamento		
	valor#	pcJuros#	numeroParcelas#
	R$ 10.000,00	0,5334	12
	R$ 500,00	0,125	5
	R$ 3.250.500,00	0,7665	72

Expected Results

	Financiamento				
	valor#	pcJuros#	numeroParcelas#	valorParcelas#	valorTotal#
created	R$ 10.000,00	0,5334	12	R$ 862,51	R$ 10.350,09
created	R$ 500,00	0,125	5	R$ 100,38	R$ 501,88
created	R$ 3.250.500,00	0,7665	72	R$58.911,94	R$ 4.241.659,83

APÊNDICE G – Exemplos de cenários de teste escrito em Linguagem de Teste Domínio Específico estruturada em tabelas

A - História de Usuário:
Eu como cliente de um banco;
Quero retirar dinheiro da minha conta corrente;
Então realizo um saque utilizando um Caixa Eletrônico.

Cenário de Aceitação 1: O saque é realizado com sucesso.

Command	**Parameter1**	**Parameter2**
Open Form	SetUp	
Edit Field	usuário	J. F. Piper
Edit Field	conta corrente	125654-08
Edit Field	saldo	90,00
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	senha	123456
Edit Field	notas de 5	0
Edit Field	notas de 10	200
Edit Field	notas de 20	0
Edit Field	notas de 25	150
Edit Field	notas de 100	100

Command	**Parameter1**	**Parameter2**
Open Form	Sacar	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	valor para sacar	30
Do Action	confirmar valor	
Check Field	valor válido	verdadeiro
Edit Field	senha	123456
Do Action	confirmar operação	
Check Field	mensagem	Operação realizada com sucesso!
Check Field	saldo	60,00

Cenário de Aceitação 2: O saque não é realizado, pois o usuário informou um valor inválido.

Command	**Parameter1**	**Parameter2**
Open Form	SetUp	
Edit Field	usuário	J. F. Piper
Edit Field	conta corrente	125654-08
Edit Field	saldo	90,00
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	senha	123456
Edit Field	notas de 5	0
Edit Field	notas de 10	200
Edit Field	notas de 20	0
Edit Field	notas de 25	150
Edit Field	notas de 100	100

Command	**Parameter1**	**Parameter2**
Open Form	Sacar	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	valor para sacar	31
Do Action	confirmar valor	
Check Field	valor válido	falso
Check Field	mensagem	O valor informado não é válido!
Check Field	saldo	90,00

Cenário de Aceitação 3: O saque não é realizado, pois o saldo da conta do usuário é insuficiente.

Command	**Parameter1**	**Parameter2**
Open Form	SetUp	
Edit Field	usuário	J. F. Piper
Edit Field	conta corrente	125654-08
Edit Field	saldo	90,00
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	senha	123456
Edit Field	notas de 5	0
Edit Field	notas de 10	200
Edit Field	notas de 20	0
Edit Field	notas de 25	150
Edit Field	notas de 100	100

Command	**Parameter1**	**Parameter2**
Open Form	Sacar	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	valor para sacar	100
Do Action	confirmar valor	
Check Field	valor válido	verdadeiro
Edit Field	senha	123456
Do Action	confirmar operação	
Check Field	mensagem	Saldo insuficiente!
Check Field	saldo	90,00

Cenário de Aceitação 4: O saque não é realizado, pois o usuário informou uma senha inválida. É a primeira vez que o usuário informa uma senha inválida nas últimas 72 horas.

Command	**Parameter1**	**Parameter2**
Open Form	SetUp	
Edit Field	usuário	J. F. Piper
Edit Field	conta corrente	125654-08
Edit Field	saldo	90,00
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	senha	123456
Edit Field	notas de 5	0
Edit Field	notas de 10	200
Edit Field	notas de 20	0
Edit Field	notas de 25	150
Edit Field	notas de 100	100

Command	**Parameter1**	**Parameter2**
Open Form	Sacar	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	valor para sacar	100
Do Action	confirmar valor	
Check Field	valor válido	verdadeiro
Edit Field	senha	654321
Do Action	confirmar operação	
Check Field	mensagem	Senha incorreta!
Check Field	saldo	90,00

Cenário de Aceitação 5: O saque não é realizado, pois o usuário informou uma senha inválida. É a segunda vez que o usuário informa uma senha inválida nas últimas 72 horas.

Command	**Parameter1**	**Parameter2**
Open Form	SetUp	
Edit Field	usuário	J. F. Piper
Edit Field	conta corrente	125654-08
Edit Field	saldo	90,00
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	senha	123456
Edit Field	notas de 5	0
Edit Field	notas de 10	200
Edit Field	notas de 20	0
Edit Field	notas de 25	150
Edit Field	notas de 100	100

Command	**Parameter1**	**Parameter2**
Open Form	SetUpHistoricoCartao	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	número da ocorrência	1
Edit Field	número de horas atrás que informou a senha do cartão errada	2

Command	**Parameter1**	**Parameter2**
Open Form	Sacar	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	valor para sacar	100
Do Action	confirmar valor	
Check Field	valor válido	verdadeiro
Edit Field	senha	654321
Do Action	confirmar operação	
Check Field	mensagem	Senha inválida! Você tem mais uma tentativa nas próximas 70 horas, do contrário, seu cartão será bloqueado por motivos de segurança.
Check Field	saldo	90,00

Cenário de Aceitação 6: O saque não é realizado, pois o usuário informou uma senha inválida. É a terceira vez que o usuário informa uma senha inválida nas últimas 72 horas.

Command	**Parameter1**	**Parameter2**
Open Form	SetUp	
Edit Field	usuário	J. F. Piper
Edit Field	conta corrente	125654-08
Edit Field	saldo	90,00
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	senha	123456
Edit Field	notas de 5	0
Edit Field	notas de 10	200
Edit Field	notas de 20	0
Edit Field	notas de 25	150
Edit Field	notas de 100	100

Command	**Parameter1**	**Parameter2**
Open Form	SetUpHistoricoCartao	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	número da ocorrência	1
Edit Field	número de horas atrás que informou a senha do cartão errada	2,1

Command	**Parameter1**	**Parameter2**
Open Form	SetUpHistoricoCartao	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	número da ocorrência	2
Edit Field	número de horas atrás que informou a senha do cartão errada	2

Command	**Parameter1**	**Parameter2**
Open Form	Sacar	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	valor para sacar	100
Do Action	confirmar valor	
Check Field	valor válido	verdadeiro
Edit Field	senha	654321
Do Action	confirmar operação	
Check Field	mensagem	Senha inválida! Seu cartão foi bloqueado por motivos de segurança! Entre em contato com a central de serviços para maiores informações.
Check Field	saldo	90,00

Cenário de Aceitação 7: O saque não é realizado, pois o cartão da conta corrente do usuário está bloqueado.

Command	**Parameter1**	**Parameter2**
Open Form	SetUp	
Edit Field	usuário	J. F. Piper
Edit Field	conta corrente	125654-08
Edit Field	saldo	90,00
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	senha	123456
Edit Field	notas de 5	0
Edit Field	notas de 10	200
Edit Field	notas de 20	0
Edit Field	notas de 25	150
Edit Field	notas de 100	100

Command	**Parameter1**	**Parameter2**
Open Form	SetUpHistoricoCartao	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	número da ocorrência	1
Edit Field	número de horas atrás que informou a senha do cartão errada	2,2

Command	**Parameter1**	**Parameter2**
Open Form	SetUpHistoricoCartao	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	número da ocorrência	2
Edit Field	número de horas atrás que informou a senha do cartão errada	2,1

Command	**Parameter1**	**Parameter2**
Open Form	SetUpHistoricoCartao	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	número da ocorrência	3
Edit Field	número de horas atrás que informou a senha do cartão errada	2

Command	**Parameter1**	**Parameter2**
Open Form	Sacar	
Edit Field	número do cartão	9999 8888 7777 6666 5555
Edit Field	valor para sacar	100
Do Action	confirmar valor	
Check Field	valor válido	verdadeiro
Edit Field	senha	654321
Do Action	confirmar operação	
Check Field	mensagem	Este cartão está bloqueado!
Check Field	saldo	90,00

B - História de Usuário:
Eu como cliente de um banco;
Quero calcular o valor das parcelas mensais de um financiamento;
Então informo os valores para o sistema e obtenho os resultados.

Cenário de Aceitação: O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa: O valor total a financiar, a taxa de juros (em percentual) e o número de parcelas que deseja pagar. Então, o sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar.

A fórmula matemática para o cálculo do EMI é:

$EMI = P \times R \times \left[\frac{(1+R)^N}{(1+R)^N-1}\right]$, onde: P é o valor total a financiar, R é a taxa de juros e N é o número de parcelas mensais.

Command	**Parameter1**	**Parameter2**
Open Form	Simular Financiamento	
Edit Field	valor a financiar	10000,00
Edit Field	taxa de juros	0,5334
Edit Field	número de parcelas	12
Check Field	valor das parcelas mensais	862,51
Check Field	valor total a pagar com juros	10350,09

Command	**Parameter1**	**Parameter2**
Open Form	Simular Financiamento	
Edit Field	valor a financiar	500,00
Edit Field	taxa de juros	0,1250
Edit Field	número de parcelas	5
Check Field	valor das parcelas mensais	100,38
Check Field	valor total a pagar com juros	501,88

Command	**Parameter1**	**Parameter2**
Open Form	Simular Financiamento	
Edit Field	valor a financiar	3250500,00
Edit Field	taxa de juros	0,7665
Edit Field	número de parcelas	72
Check Field	valor das parcelas mensais	58911,94
Check Field	valor total a pagar com juros	4241659,83

APÊNDICE H – Exemplos de cenários de teste escritos utilizando Histórias de Usuário

História de Usuário

Eu como cliente de um banco,
Quero fazer uma retirada em dinheiro da minha conta corrente,
Então eu vou até um ATM, realizo a identificação através de cartão magnético e senha, escolho o valor desejado e realizo o saque.

Critérios de Aceitação

Regras de Negócio / Confirmação:

- Se o cliente informar a senha do cartão incorretamente três vezes dentro de 72 horas, o cartão é bloqueado.
- Se o cliente informar um valor para sacar maior que o saldo da conta corrente, o caixa eletrônico emite uma mensagem que o saldo é insuficiente e encerra a operação.
- Se não há notas disponíveis para entregar determinado valor ao cliente, o caixa eletrônico deve emitir uma mensagem e encerrar a operação.

Pré-condições para os critérios de aceitação:
J. F. Piper é cliente do The American Bank. Ele tem uma conta corrente número 125654-08 e o saldo da conta é R$ 90,00. Piper vai a um caixa eletrônico e se identifica com o cartão da conta corrente número 8888777766665555. A senha deste cartão é 123456. O caixa eletrônico tem, disponível para saque, a seguinte quantidade de notas: 200 notas de R$ 10, 0 notas de R$ 20, 150 notas de R$ 50 e 110 notas de R$ 100.

Critério de Aceitação 01: Um cliente vai até um Caixa eletrônico e realiza com sucesso um saque, escolhendo um valor válido, utilizando um cartão magnético da conta corrente e a senha correta. A conta corrente do cliente possui saldo suficiente.
Exemplos:

- Piper escolhe a opção *Saque*. Ele informa que deseja sacar R$ 30,00. O caixa eletrônico pede a senha do cartão. Piper informa a senha 123456 e confirma a operação. Piper retira o cartão do caixa eletrônico. O caixa eletrônico mostra a mensagem: "A transação foi completada com sucesso!". O caixa eletrônico

entrega o dinheiro na bandeja. O saldo da conta corrente de Piper atualizado deverá ser R$ 60,00.

Critério de Aceitação 02: Um cliente vai até um Caixa eletrônico e não consegue realizar o saque pois informa uma senha incorreta.
Exemplos:

- Piper nunca informou a senha do seu cartão incorretamente. Piper escolhe a opção *Saque*. Ele informa que deseja sacar R$ 30,00. O caixa eletrônico pede a senha do cartão. Piper informa a senha 123123 e confirma a operação. Piper retira o cartão do caixa eletrônico. O caixa eletrônico mostra a mensagem: "Senha inválida!". O saldo atualizado da conta corrente de Piper deverá ser R$ 90,00.
- Piper informou a senha do seu cartão incorretamente pela 1ª vez há duas horas atrás. Piper escolhe a opção *Saque*. Ele informa que deseja sacar R$ 30,00. O caixa eletrônico pede a senha do cartão. Piper informa a senha 123123 e confirma a operação. Piper retira o cartão do caixa eletrônico. O caixa eletrônico mostra a mensagem "Senha inválida! Você tem mais uma tentativa nas próximas 70 horas, do contrário, seu cartão será bloqueado por motivos de segurança.". O saldo atualizado da conta corrente de Piper deverá ser R$ 90,00.
- Piper informou a senha do seu cartão incorretamente 2 vezes nas últimas 72 horas. Piper escolhe a opção *Saque*. Ele informa que deseja sacar R$ 30,00. O caixa eletrônico pede a senha do cartão Piper informa a senha 123123 e confirma a operação. Piper retira o cartão do caixa eletrônico. O caixa eletrônico mostra a mensagem "Senha inválida! Seu cartão foi bloqueado por motivos de segurança! Entre em contato com a central de serviços para maiores informações.". O saldo atualizado da conta corrente de Piper deverá ser R$ 90,00.

Critério de Aceitação 03: Um cliente vai até um Caixa eletrônico e não consegue realizar o saque pois informa um valor incorreto, isto é, não existem notas disponíveis no caixa eletrônico para entregar o valor solicitado pelo cliente.
Exemplos:

- Piper escolhe a opção *Saque*. Ele informa que deseja sacar R$ 31,00. O caixa eletrônico pede a senha do cartão, Piper informa a senha 123456 e confirma a operação. Piper retira o cartão do

caixa eletrônico. O caixa eletrônico mostra a mensagem: "O valor informado é inválido!". O saldo atualizado da conta corrente de Piper deverá ser R$ 90,00.

Critério de Aceitação 04: Um cliente vai até um Caixa eletrônico e não consegue realizar o saque pois o saldo disponível é insuficiente.
Exemplos:

- Piper escolhe a opção Saque. Ele informa que deseja sacar R$ 110. O caixa eletrônico pede a senha do cartão, Piper informa a senha 123456 e confirma a operação. Piper retira o cartão do caixa eletrônico. O caixa eletrônico mostra a mensagem: "Saldo insuficiente! Por favor, verifique seu saldo e tente novamente.". O saldo atualizado da conta corrente de Piper deverá ser R$ 90,00.

História de Usuário

Eu como cliente de um banco;
Quero calcular o valor das parcelas mensais de um financiamento;
Então informo os valores para o sistema e obtenho os resultados.

Critérios de Aceitação

Regras de Negócio / Confirmação:
A sigla EMI refere-se ao valor das parcelas mensais de um empréstimo pagas por um cliente em determinada data ao banco. O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa:

- O valor total a financiar;
- A taxa de juros (em percentual);
- O número de parcelas que deseja pagar.

O sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar. A fórmula matemática para o cálculo do EMI é:

$EMI = P \times R \times \left[\frac{(1+R)^N}{(1+R)^N - 1}\right]$, onde: P é o valor total a financiar, R é a taxa de juros e N é o número de parcelas mensais.

Pré-condições para os critérios de aceitação:
J. F. Piper é cliente do The American Bank. Ele tem uma conta corrente número 125654-08 e o saldo da conta é R$ 90,00. Piper vai a um caixa

eletrônico e se identifica com o cartão da conta corrente número 8888777766665555. A senha deste cartão é 123456. O caixa eletrônico tem, disponível para saque, a seguinte quantidade de notas: 200 notas de R$ 10, 0 notas de R$ 20, 150 notas de R$ 50 e 110 notas de R$ 100.

Exemplos:

- J. F. Piper quer simular o valor das parcelas mensais (EMI) de um financiamento. Piper escolhe a opção "Simular financiamento". Ele informa o valor a financiar, a taxa de juros, e o número de parcelas que deseja pagar. A aplicação retorna o valor das parcelas do financiamento e o valor total a pagar com juros.

valor a financiar	taxa de juros	número de parcelas	valor das parcelas mensais?	valor total a pagar com juros?
10000,00	0,5334	12	862,51	10350,09
100000000,00	0,3227	36	2946725,65	106082123,23
500,00	0,1250	5	100,38	501,88
3250500,00	0,7665	72	58911,94	4241659,83

APÊNDICE I – Exemplos de cenários de teste escritos utilizando Histórias de Usuário com anotações

A exibição dos cenários de teste, do ponto de vista do usuário, utilizado esta técnica de teste de aceitação, tem o mesmo valor semântico do que quanto utilizamos a técnica apresentada no APÊNDICE H – Exemplos de cenários de teste escritos utilizando Histórias de Usuário. Portanto, neste apêndice serão disponibilizados somente trechos dos códigos HTML dos testes.

A - História de Usuário:
Eu como cliente de um banco;
Quero retirar dinheiro da minha conta corrente;
Então realizo um saque utilizando um Caixa Eletrônico.

Pré-condições:

```
<div concordion:example="before">
<p concordion:execute="setCustomer(#customerName, #accountNumber,
   #accountBalance, #bankCardNumber, #bankCardPIN)">
<span concordion:set="#customerName">J. F. Piper</span>
   é cliente do The American Bank
Ele tem uma conta corrente número
   <span concordion:set="#accountNumber">12565408</span>
   e o saldo da conta é
   <span concordion:set="#accountBalance">90,00</span> reais.
Piper vai a um caixa eletrônico e se indentifica com o cartão número
   <span concordion:set="#bankCardNumber">8888777766665555</span>.
A senha deste cartão é
   <span concordion:set="#bankCardPIN">123456</span>.
O Caixa eletrônico tem a seguinte quantidade de notas de
   10, 20, 50 e 100, respectivamente:
   <span concordion:execute="setAvailableBills(#TEXT)">
      200,0,150,110,100
   </span>.
</p></div>
```

Cenário de Aceitação 1: O saque é realizado com sucesso.

```
<div concordion:example="basic-flow">
<p>
Piper escolhe a opção Saque.
Ele informa que deseja sacar
   <span concordion:set="#amount">30</span> reais.
O caixa eletrônico pede a senha do cartão.
Piper informa a senha
   <span concordion:set="#pin">123456</span>
   e confirma a operação.
Piper retira o cartão do caixa eletrônico.
O caixa eletrônico mostra a mensagem
   <span concordion:assertequals="withdraw(#bankCardNumber,
      #pin,#amount)">
      A transação foi completada com sucesso!
   </span>.
O caixa eletrônico entrega o dinheiro.
O saldo da conta corrente deverá ser
   <span concordion:assert-equals="getBalance(#bankCardNumber)">
      60,00
   </span> reais.
</p>
</div>
```

Cenário de Aceitação 2: O saque não é realizado, pois o usuário informou um valor inválido.

```
<div concordion:example="basic-flow"><p>
Piper escolhe a opção Saque.
Ele informa que deseja sacar
   <span concordion:set="#amount">31</span> Reais.
O caixa eletrônico pede a senha do cartão.
Piper informa a senha
   <span concordion:set="#pin">123456</span>
   e confirma a operação.
Piper retira o cartão do caixa eletrônico.
O caixa eletrônico mostra a mensagem
   <span concordion:assertequals="withdraw(#bankCardNumber,
      #pin,#amount)">
      O valor informado é inválido!</span>.
O saldo da conta corrente deverá ser
   <span concordion:assert-equals="getBalance(#bankCardNumber)">
      90,00</span> Reais.
</p></div>
```

Cenário de Aceitação 3: O saque não é realizado, pois o saldo da conta do usuário é insuficiente.

```
<div concordion:example="basic-flow"><p>
Piper escolhe a opção Saque.
Ele informa que deseja sacar
   <span concordion:set="#amount">100</span> Reais.
O caixa eletrônico pede a senha do cartão.
Piper informa a senha
   <span concordion:set="#pin">123456</span>
   e confirma a operação.
Piper retira o cartão do caixa eletrônico.
O caixa eletrônico mostra a mensagem
   <span concordion:assertequals="withdraw(#bankCardNumber,
      #pin,#amount)">
      Saldo insuficiente!</span>.
O saldo da conta corrente deverá ser
   <span concordion:assert-equals="getBalance(#bankCardNumber)">
      90,00</span> Reais.
</p></div>
```

Cenário de Aceitação 4: O saque não é realizado, pois o usuário informou uma senha inválida. É a primeira vez que o usuário informa uma senha inválida nas últimas 72 horas.

```
<div concordion:example="basic-flow"><p>
Piper informou a senha do seu cartão incorretamente
<span concordion:execute="setNumberOfIncorretPINWithin72Hours(#TEXT,
   #bankCardNumber)">0</span> vezes em 72 horas.
Piper escolhe a opção Saque.
Ele informa que deseja sacar
   <span concordion:set="#amount">90</span> Reais.
O caixa eletrônico pede a senha do cartão.
Piper informa a senha
   <span concordion:set="#pin">654321</span>
   e confirma a operação.
Piper retira o cartão do caixa eletrônico.
O caixa eletrônico mostra a mensagem
   <span concordion:assertequals="withdraw(#bankCardNumber,
      #pin,#amount)">
      Senha incorreta!
   </span>.
O saldo da conta corrente deverá ser
   <span concordion:assert-equals="getBalance(#bankCardNumber)">
      90,00</span> Reais.
</p></div>
```

Cenário de Aceitação 5: O saque não é realizado, pois o usuário informou uma senha inválida. É a segunda vez que o usuário informa uma senha inválida nas últimas 72 horas.

```
<div concordion:example="basic-flow"><p>
Piper acabou de informou a senha do seu cartão incorretamente
<span concordion:execute="setNumberOfIncorretPINWithin72Hours(#TEXT,
   #bankCardNumber)">1</span> vez.
Piper escolhe a opção Saque.
Ele informa que deseja sacar
   <span concordion:set="#amount">90</span> Reais.
O caixa eletrônico pede a senha do cartão.
Piper informa a senha
   <span concordion:set="#pin">654321</span>
   e confirma a operação.
Piper retira o cartão do caixa eletrônico.
O caixa eletrônico mostra a mensagem
   <span concordion:assertequals="withdraw(#bankCardNumber,
      #pin,#amount)">
      Senha inválida! Você tem mais uma tentativa nas próximas
      72 horas, do contrário, seu cartão será bloqueado por motivos
      de segurança</span>.
O saldo da conta corrente deverá ser
   <span concordion:assert-equals="getBalance(#bankCardNumber)">
      90,00</span> Reais.
</p></div>
```

Cenário de Aceitação 6: O saque não é realizado, pois o usuário informou uma senha inválida. É a terceira vez que o usuário informa uma senha inválida nas últimas 72 horas.

```
<div concordion:example="basic-flow"><p>
Piper informou a senha do seu cartão incorretamente
<span concordion:execute="setNumberOfIncorretPINWithin72Hours(#TEXT,
   #bankCardNumber)">2</span> vezes nas últimas 2 horas.
Piper escolhe a opção Saque.
Ele informa que deseja sacar
   <span concordion:set="#amount">90</span> Reais.
O caixa eletrônico pede a senha do cartão.
Piper informa a senha
   <span concordion:set="#pin">654321</span> e confirma a operação.
Piper retira o cartão do caixa eletrônico.
O caixa eletrônico mostra a mensagem
   <span concordion:assertequals="withdraw(#bankCardNumber,
      #pin,#amount)">
      Senha inválida! Seu cartão foi bloqueado por motivos
      de segurança! Entre em contato com a central de serviços
      para maiores informações.</span>.
O saldo da conta corrente deverá ser
   <span concordion:assert-equals="getBalance(#bankCardNumber)">
      90,00</span> Reais.
</p></div>
```

Cenário de Aceitação 7: O saque não é realizado, pois o cartão da conta corrente do usuário está bloqueado.

```
<div concordion:example="basic-flow"><p>
Piper informou a senha do seu cartão incorretamente
<span concordion:execute="setNumberOfIncorretPINWithin72Hours(#TEXT,
   #bankCardNumber)">3</span> vezes em um período de 72 horas.
Piper escolhe a opção Saque.
Ele informa que deseja sacar
<span concordion:set="#amount">90</span> Reais.
O caixa eletrônico pede a senha do cartão.
Piper informa a senha
   <span concordion:set="#pin">123456</span> e confirma a operação.
Piper retira o cartão do caixa eletrônico.
O caixa eletrônico mostra a mensagem
   <span concordion:assertequals="withdraw(#bankCardNumber,
      #pin,#amount)">
      Este cartão está bloqueado!
      Entre em contato com a central de serviços
      para maiores informações.</span>.
O saldo da conta corrente deverá ser
   <span concordion:assert-equals="getBalance(#bankCardNumber)">
      90,00</span> Reais.
</p></div>
```

B - História de Usuário:
Eu como cliente de um banco;
Quero calcular o valor das parcelas mensais de um financiamento;
Então informo os valores para o sistema e obtenho os resultados.

Cenário de Aceitação: O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa: O valor total a financiar, a taxa de juros (em percentual) e o número de parcelas que deseja pagar. Então, o sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar.

A fórmula matemática para o cálculo do EMI é:

$EMI = P \times R \times \left[\frac{(1+R)^N}{(1+R)^N-1}\right]$, onde: P é o valor total a financiar, R é a taxa de juros e N é o número de parcelas mensais.

```
<table concordion:execute="#result = calculateEMI(#amout,
   #interestRate, #numberOfMonthlyInstallments)">
<tr><th concordion:set="#amout">Valor financiado</th>
   <th concordion:set="#interestRate">Taxa de juros (%)</th>
   <th concordion:set="#numberOfMonthlyInstallments">
      Número de parcelas</th>
   <th concordion:assert-equals="#result.emiValue">
      Valor das parcelas</th>
   <th concordion:assert-equals="#result.loanAmountWithInterest">
      Valor total financiado com Juros</th></tr>
<tr><td>10000.00</td>
   <td>0.5334</td>
   <td>12</td>
   <td>862.51</td>
   <td>10350.09</td></tr>
<tr><td>100000000.00</td>
   <td>0.3227</td>
   <td>36</td>
   <td>2946725.65</td>
   <td>106082123.23</td></tr>
<tr><td>500.00</td>
   <td>0.1250</td>
   <td>5</td>
   <td>100.38</td>
   <td>501.88</td></tr>
<tr<td>3250500.00</td>
   <td>0.7665</td>
   <td>72</td>
   <td>58911.94</td>
   <td>4241659.83</td></tr></table>
```

APÊNDICE J – Exemplos de cenários de teste escritos utilizando *Gherkin Language*

```
Feature: Caixa Eletrônico
Eu como cliente de um banco,
Quero fazer uma retirada em dinheiro da minha conta corrente,
Então eu vou até um ATM, realizo a identificação através de um cartão
magnético e uma senha de 6 dígitos numéricos, escolho o valor desejado
e realizo o saque.

Scenario: Um cliente vai até um caixa eletrônico e realiza com sucesso
um saque, escolhendo um valor válido, utilizando um cartão e a senha
correta. A conta corrente do cliente possui saldo suficiente.

Given o cliente 'J. F.Piper'
 And o número da conta corrente do cliente é '125654-08'
 And o número do cartão da conta corrente é '9988776655443322'
 And a senha do cartão da conta corrente é '123456'
 And o saldo da conta corrente do cliente é $90.00
 And o cliente escolhe a opção 'Sacar'
 And no caixa eletrônico há 200 notas 10
 And 0 notas de 20
 And 150 notas 50
 And 110 notas 100
 And o cliente não errou a senha do cartão nas últimas 72 horas
When o cliente informar o valor para sacar de $30.00
 And informar a senha '123456'
 And pressionar a tecla 'Confirma'
Then o caixa eletrônico disponibiliza o dinheiro na bandeja
 And o saldo atualizado da conta correte do cliente é de $60.00

Scenario: Um cliente vai até um caixa eletrônico e não consegue realizar
o saque pois informa um valor incorreto, isto é, não existem notas
disponíveis no caixa eletrônico para entregar o valor solicitado pelo
cliente.

Given o cliente 'J. F.Piper'
 And o número da conta corrente do cliente é '125654-08'
 And o número do cartão da conta corrente é '9988776655443322'
 And a senha do cartão da conta corrente é '123456'
 And o saldo da conta corrente do cliente é $90.00
 And o cliente escolhe a opção 'Sacar'
 And no caixa eletrônico há 200 notas 10
 And 0 notas de 20
 And 150 notas 50
 And 110 notas 100
 And o cliente não errou a senha do cartão nas últimas 72 horas
When o cliente informar o valor para sacar de $31.00
Then o caixa eletrônico mostra a mensagem 'O valor informado é
inválido!'
```

```
Scenario: Um cliente vai até um caixa eletrônico e não consegue realizar
o saque pois o saldo disponível na conta corrente é insuficiente.

Given o cliente 'J. F.Piper'
 And o número da conta corrente do cliente é '125654-08'
 And o número do cartão da conta corrente é '9988776655443322'
 And a senha do cartão da conta corrente é '123456'
 And o saldo da conta corrente do cliente é $90.00
 And o cliente escolhe a opção 'Sacar'
 And no caixa eletrônico há 200 notas 10
 And 0 notas de 20
 And 150 notas 50
 And 110 notas 100
 And o cliente não errou a senha do cartão nas últimas 72 horas
When o cliente informar o valor para sacar de $100.00
 And informar a senha '123456'
 And pressionar a tecla 'Confirma'
Then o caixa eletrônico mostra a mensagem 'Saldo insuficiente! Por
favor, verifique seu saldo e tente novamente.'
 And o saldo atualizado da conta correte do cliente é de $90.00

Scenario: Um cliente vai até um caixa eletrônico e não consegue realizar
o saque pois informa uma senha incorreta. É a primeira vez que o cliente
erra a senha dentro do período de 72 horas. Se ele errar a senha três
vezes no período de 72 horas o cartão é bloqueado por motivos de
segurança.

Given o cliente 'J. F.Piper'
 And o número da conta corrente do cliente é '125654-08'
 And o número do cartão da conta corrente é '9988776655443322'
 And a senha do cartão da conta corrente é '123456'
 And o saldo da conta corrente do cliente é $90.00
 And o cliente escolhe a opção 'Sacar'
 And no caixa eletrônico há 200 notas 10
 And 0 notas de 20
 And 150 notas 50
 And 110 notas 100
 And o cliente não errou a senha do cartão nas últimas 72 horas
When o cliente informar o valor para sacar de $30.00
 And informar a senha '654321'
 And pressionar a tecla 'Confirma'
Then o caixa eletrônico mostra a mensagem 'Senha inválida!'
 And o saldo atualizado da conta correte do cliente é de $90.00
```

```
Scenario: Um cliente vai até um Caixa eletrônico e não consegue realizar
o saque pois informa uma senha incorreta. É a segunda vez que o cliente
erra a senha dentro do período de 72 horas. Se ele errar a senha três
vezes no período de 72 horas o cartão é bloqueado por motivos de
segurança.

Given o cliente 'J. F.Piper'
 And o número da conta corrente do cliente é '125654-08'
 And o número do cartão da conta corrente é '9988776655443322'
 And a senha do cartão da conta corrente é '123456'
 And o saldo da conta corrente do cliente é $90.00
 And o cliente escolhe a opção 'Sacar'
 And no caixa eletrônico há 200 notas 10
 And 0 notas de 20
 And 150 notas 50
 And 110 notas 100
 And o cliente informou 'pela primeira vez' sua senha incorretamente
há 2 horas atrás
When o cliente informar o valor para sacar de $30.00
 And informar a senha '654321'
 And pressionar a tecla 'Confirma'
Then o caixa eletrônico mostra a mensagem 'Senha inválida! Você tem
mais uma tentativa nas próximas 70 horas, do contrário, seu cartão será
bloqueado por motivos de segurança.'
 And o saldo atualizado da conta correte do cliente é de $90.00

Scenario: Um cliente vai até um Caixa eletrônico e não consegue realizar
o saque pois informa uma senha incorreta. É a terceira vez que o cliente
erra a senha dentro do período de 72 horas, portanto, seu cartão será
bloqueado por motivos de segurança.

Given o cliente 'J. F.Piper'
 And o número da conta corrente do cliente é '125654-08'
 And o número do cartão da conta corrente é '9988776655443322'
 And a senha do cartão da conta corrente é '123456'
 And o saldo da conta corrente do cliente é $90.00
 And o cliente escolhe a opção 'Sacar'
 And no caixa eletrônico há 200 notas 10
 And 0 notas de 20
 And 150 notas 50
 And 110 notas 100
 And o cliente informou 'pela primeira vez' sua senha incorretamente
há 2.1 horas atrás
 And o cliente informou 'pela segunda vez' sua senha incorretamente há
2 horas atrás
When o cliente informar o valor para sacar de $30.00
 And informar a senha '654321'
 And pressionar a tecla 'Confirma'
Then o caixa eletrônico mostra a mensagem 'Senha inválida! Seu cartão
foi bloqueado por motivos de segurança! Entre em contato com a central
de serviços para maiores informações.'
 And o saldo atualizado da conta correte do cliente é de $90.00
```

```
Scenario: Um cliente vai até um Caixa eletrônico e não consegue realizar
o saque pois seu cartão está bloqueado.

Given o cliente 'J. F.Piper'
 And o número da conta corrente do cliente é '125654-08'
 And o número do cartão da conta corrente é '9988776655443322'
 And a senha do cartão da conta corrente é '123456'
 And o saldo da conta corrente do cliente é $90.00
 And o cliente escolhe a opção 'Sacar'
 And no caixa eletrônico há 200 notas 10
 And 0 notas de 20
 And 150 notas 50
 And 110 notas 100
 And o cliente informou 'pela primeira vez' sua senha incorretamente
há 2.2 horas atrás
 And o cliente informou 'pela segunda vez' sua senha incorretamente há
2.1 horas atrás
 And o cliente informou 'pela terceira vez' sua senha incorretamente
há 2 horas atrás
When o cliente informar o valor para sacar de $30.00
 And informar a senha '123456'
 And pressionar a tecla 'Confirma'
      Then o caixa eletrônico mostra a mensagem 'Este cartão está
bloqueado!'
```

Feature: Simulação de Empréstimo
Eu como cliente,
Quero simular o valor das parcelas e o valor total com juros de um empréstimo,
Então vou acessar o site do banco, informar o montante a financiar, a taxa de juros mensal e número de parcelas. O site irá calcular o valor de cada parcela e o valor final do empréstimo com juros.

Scenario Outline: A sigla EMI refere-se ao valor das parcelas mensais de um empréstimo pagas por um cliente em determinada data ao banco. O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa:

- O valor total a financiar;
- A taxa de juros (em percentual);
- O número de parcelas que deseja pagar.

O sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar.

A fórmula matemática para o cálculo do EMI é:

$$EMI = P \times R \times \left[\frac{(1+R)^N}{(1+R)^N - 1}\right]$$, onde: P é o valor total a financiar, R é a taxa de juros e N é o número de parcelas mensais.

Given um cliente qualquer
When o cliente informar R$ <vlFinanciamento> como o valor a ser financiado
And e que a taxa de juros mensais do financiamento é <pcJuros>%
And e que o financiamento será pago em <nuParcelas> parcelas
Then o sistema calcula que o valor de cada uma das parcelas é <vlParcelas>
And que o valor total do financiamento com juros e <vlFinComJuros>

Examples:

vlFinanciamento	pcJuros	nuParcelas	vlParcelas	vlFinComJuros
10000.00	0.5334	12	862.51	10350.09
100000000.00	0.3227	36	2946725.65	106082123.23
500.00	0.1250	5	100.38	501.88
3250500.00	0.7665	72	58911.94	4241659.83

APÊNDICE K – Exemplos de cenários de teste escritos utilizando o *EasyAccept*

A - História de Usuário:
Eu como cliente de um banco;
Quero retirar dinheiro da minha conta corrente;
Então realizo um saque utilizando um Caixa Eletrônico.

Cenário de Aceitação 1: O saque é realizado com sucesso.

```
incluirCliente nome="J.F.Piper" nuCartao="9999888877776666" nuContaCorrente="125-8" saldo=90,00 senha="123456"
carregarNotasCaixaEletronico notasDe5=0 notasDe10=200 notasDe20=0 notasDe50=150 notasDe100=150
setErroDeSenhaNasUltimas72horas nuCartao ="9999888877776666" nuDeErros=0 nuHorasDoPrimeiroErro=0
expect "Sucesso!" sacar nuCartao="9999888877776666" valor=30,00 senha=123456
expect 60,00 verSaldo nuDoCartao="9999888877776666"
```

Cenário de Aceitação 2: O saque não é realizado, pois o usuário informou um valor inválido.

```
incluirCliente nome="J.F.Piper" nuCartao="9999888877776666" nuContaCorrente="125-8" saldo=90,00 senha="123456"
carregarNotasCaixaEletronico notasDe5=0 notasDe10=200 notasDe20=0 notasDe50=150 notasDe100=150
setErroDeSenhaNasUltimas72horas nuCartao="9999888877776666" nuErros=0 nuHorasDoPrimeiroErro=0
expectError "O valor informado é inválido!" sacar nuCartao="9999888877776666" valor=31,00 senha=123456
expect 90,00 verSaldo nuCartao="9999888877776666"
```

Cenário de Aceitação 3: O saque não é realizado, pois o saldo da conta do usuário é insuficiente.

```
incluirCliente nome="J.F.Piper" nuCartao="9999888877776666" nuContaCorrente="125-8" saldo=90.00 senha="123456"
carregarNotasCaixaEletronico notasDe5=0 notasDe10=200 notasDe20=0 notasDe50=150 notasDe100=150
setErroDeSenhaNasUltimas72horas nuCartao="9999888877776666" nuErros=0 nuHorasDoPrimeiroErro=0
expectError "Saldo insuficiente!" sacar nuCartao="9999888877776666" valor=100,00 senha=123456
expect 90,00 verSaldo nuCartao="9999888877776666"
```

Cenário de Aceitação 4: O saque não é realizado, pois o usuário informou uma senha inválida.

```
incluirCliente nome="J.F.Piper" nuCartao="9999888877776666" nuContaCorrente="125-8" saldo=90,00 senha="123456"
carregarNotasCaixaEletronico notasDe5=0 notasDe10=200 notasDe20=0 notasDe50=150 notasDe100=150
setErroDeSenhaNasUltimas72horas nuCartao="9999888877776666" nuErros=0 nuHorasDoPrimeiroErro=0
expectError "Senha errada!" sacar nuCartao="9999888877776666" valor=30,00 senha=123123
expect 90,00 verSaldo nuCartao="9999888877776666"
```

Cenário de Aceitação 5: O saque não é realizado, pois o usuário informou uma senha inválida. É a segunda vez que o usuário informa uma senha inválida nas últimas 72 horas.

```
incluirCliente nome="J.F.Piper" nuCartao="9999888877776666" nuContaCorrente="125-8" saldo=90,00 senha="123456"
carregarNotasCaixaEletronico notasDe5=0 notasDe10=200 notasDe20=0 notasDe50=150 notasDe100=150
setErroDeSenhaNasUltimas72horas nuCartao="9999888877776666" nuErros=1 nuHorasDoPrimeiroErro=2
expectError "Senha errada! Resta mais uma tentativa." sacar nuCartao="9999888877776666" valor=30,00 senha=12323
expect 90,00 verSaldo nuCartao="9999888877776666"
```

Cenário de Aceitação 6: O saque não é realizado, pois o usuário informou uma senha inválida. É a terceira vez que o usuário informa uma senha inválida nas últimas 72 horas.

```
incluirCliente nome="J.F.Piper" nuCartao="9999888877776666" nuContaCorrente="125-8" saldo=90,00 senha="123456"
carregarNotasCaixaEletronico notasDe5=0 notasDe10=200 notasDe20=0 notasDe50=150 notasDe100=150
setErroDeSenhaNasUltimas72horas nuCartao="9999888877776666" nuErros=2 nuHorasDoPrimeiroErro=2
expectError "Senha errada! Seu cartão foi bloqueado.” sacar nuCartao="9999888877776666" valor=30,00 senha=12323
expect 90,00 verSaldo nuCartao="9999888877776666"
```

Cenário de Aceitação 7: O saque não é realizado, pois o cartão da conta corrente do usuário está bloqueado.

```
incluirCliente nome="J.F.Piper" nuCartao="9999888877776666" nuContaCorrente="125-8" saldo=90,00 senha="123456"
carregarNotasCaixaEletronico notasDe5=0 notasDe10=200 notasDe20=0 notasDe50=150 notasDe100=150
setErroDeSenhaNasUltimas72horas nuCartao="9999888877776666" nuErros=3 nuHorasDoPrimeiroErro=2
expectError "Cartão bloqueado.” sacar nuCartao="9999888877776666" valor=30,00 senha=123456
expect 90,00 verSaldo nuCartao="9999888877776666"
```

B - História de Usuário:
Eu como cliente de um banco;
Quero calcular o valor das parcelas mensais de um financiamento;
Então informo os valores para o sistema e obtenho os resultados.

Cenário de Aceitação: O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa: O valor total a financiar, a taxa de juros (em percentual) e o número de parcelas que deseja pagar. Então, o sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar.

A fórmula matemática para o cálculo do EMI é:

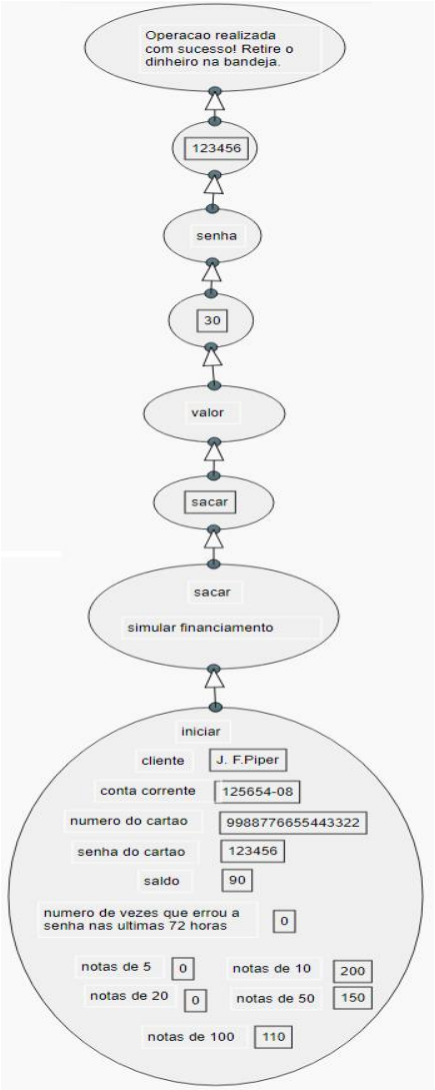
$EMI = P \times R \times \left[\frac{(1+R)^N}{(1+R)^N-1}\right]$, onde: P é o valor total a financiar, R é a taxa de juros e N é o número de parcelas mensais.

```
expect 10350.09 calcularVTotalDoFinanciamentoComJuros vlFinanciar=10000.00 pcJuros=0.5334 nuParcelas=12
expect 862.51 calcularVTotalDoFinanciamentoComJuros vlFinanciar=10000.00 pcJuros=0.5334 nuParcelas=12
expect 501.88 calcularVTotalDoFinanciamentoComJuros vlFinanciar=500.00 pcJuros=0.1250 nuParcelas=5
expect 100.38 calcularVTotalDoFinanciamentoComJuros vlFinanciar=500.00 pcJuros=0.1250 nuParcelas=5
expect 4241659.83 calcularVTotalDoFinanciamentoComJuros vlFinanciar=3250500.00 pcJuros=0.7665 nuParcelas=72
expect 58911.94 calcularVTotalDoFinanciamentoComJuros vlFinanciar=3250500.00 pcJuros=0.7665 nuParcelas=72
```

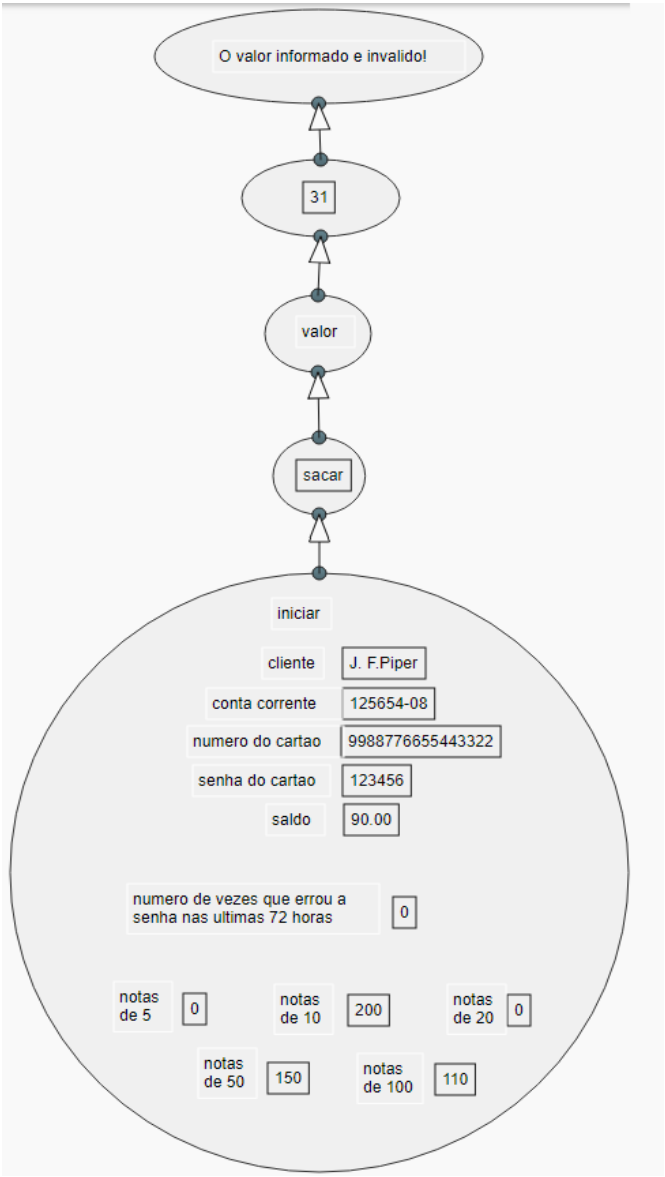
APÊNDICE L – Exemplos de cenários de teste escritos utilizando US-UID

Caso de Uso: Sacar dinheiro em um caixa eletrônico.

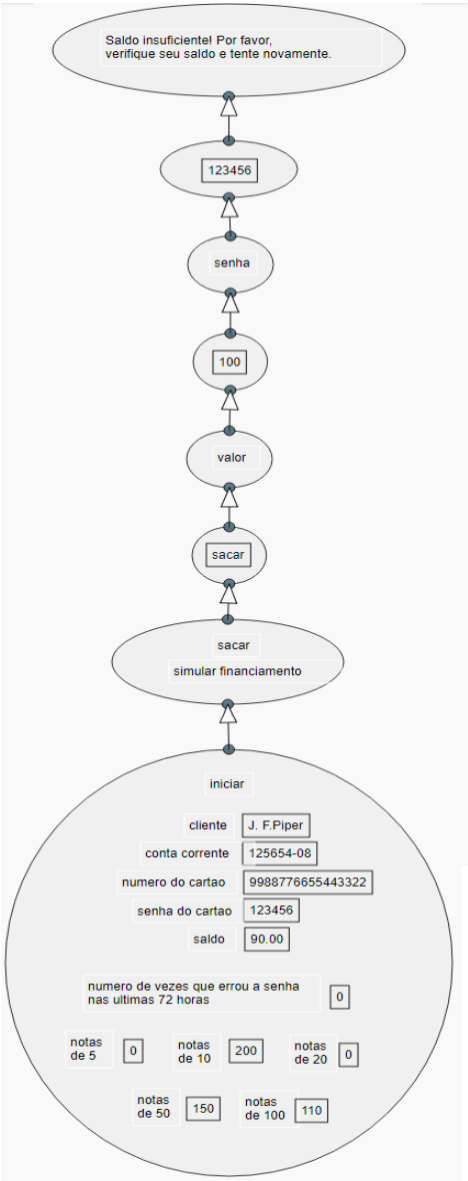
Cenário de Aceitação 1: O saque é realizado com sucesso.



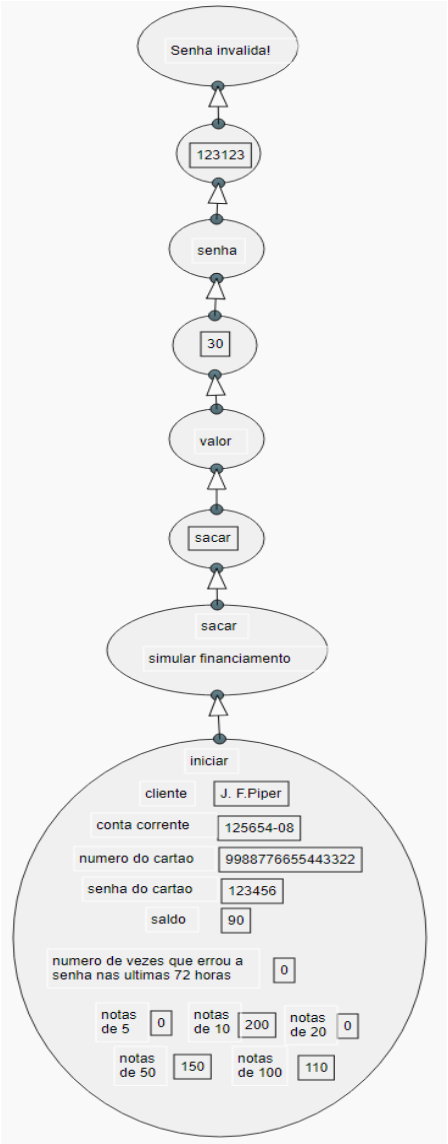
Cenário de Aceitação 2: O saque não é realizado, pois o usuário informou um valor inválido.



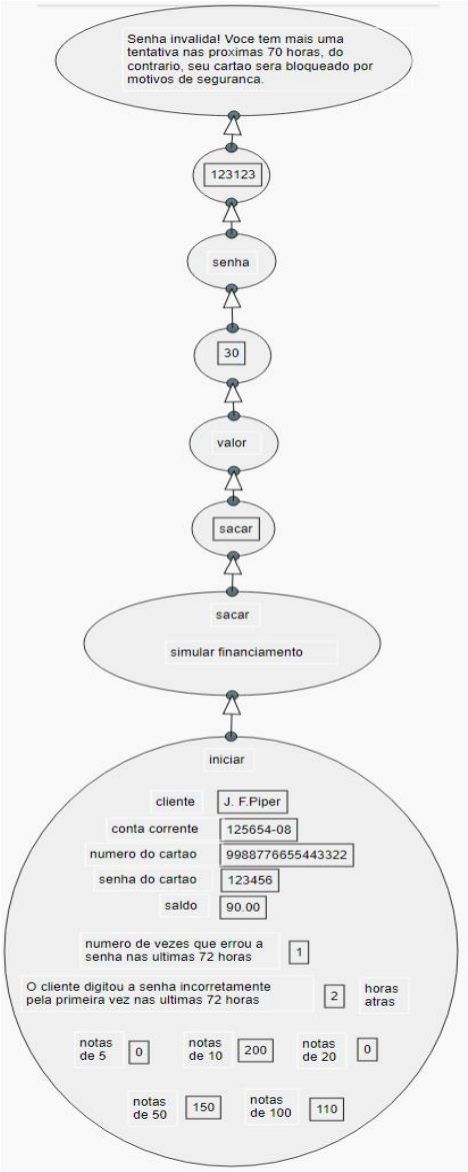
Cenário de Aceitação 3: O saque não é realizado, pois o saldo da conta do usuário é insuficiente.



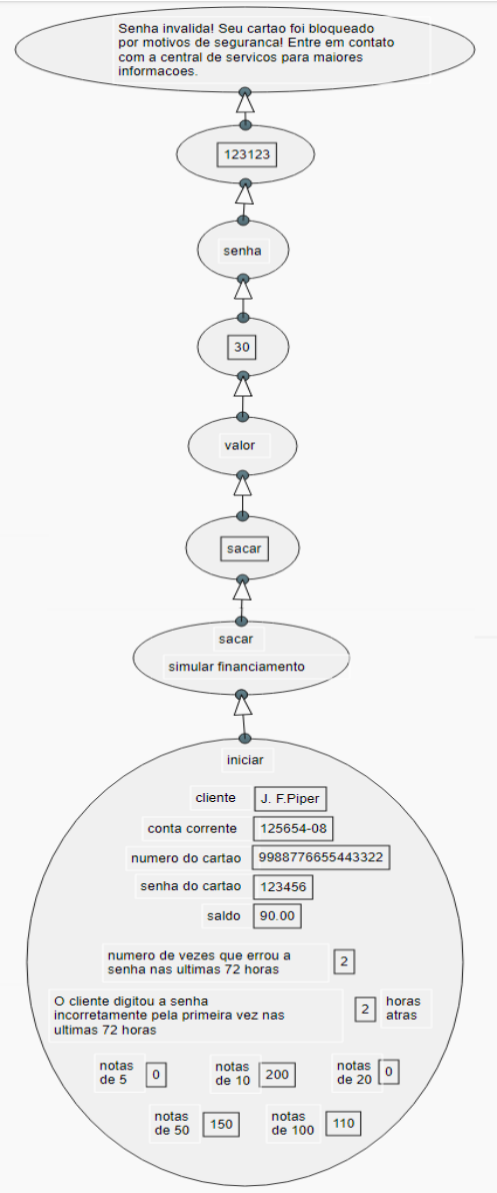
Cenário de Aceitação 4: O saque não é realizado, pois o usuário informou uma senha inválida. É a primeira vez que o usuário informa uma senha inválida nas últimas 72 horas.



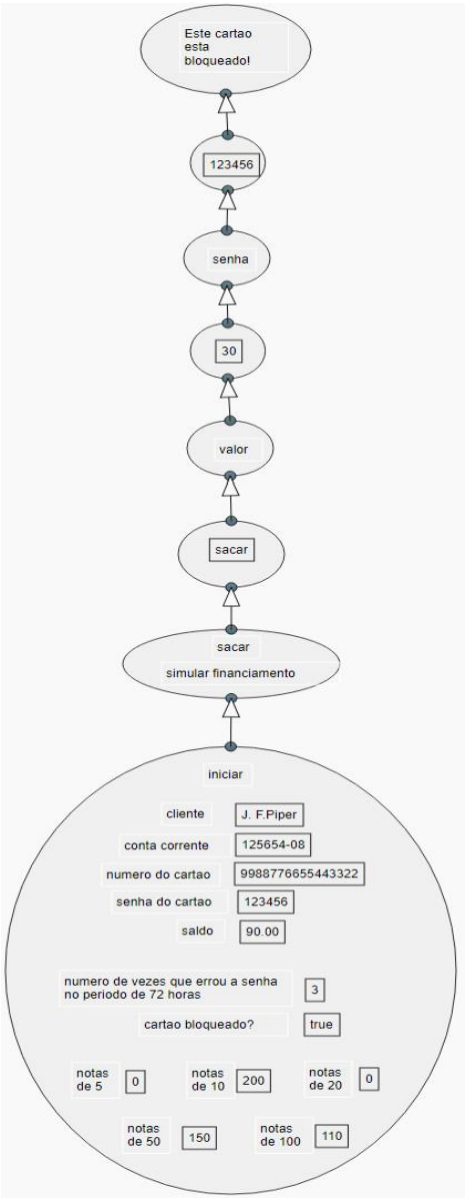
Cenário de Aceitação 5: O saque não é realizado, pois o usuário informou uma senha inválida. É a segunda vez que o usuário informa uma senha inválida nas últimas 72 horas.



Cenário de Aceitação 6: O saque não é realizado, pois o usuário informou uma senha inválida. É a terceira vez que o usuário informa uma senha inválida nas últimas 72 horas.



Cenário de Aceitação 7: O saque não é realizado, pois o cartão da conta corrente do usuário está bloqueado.

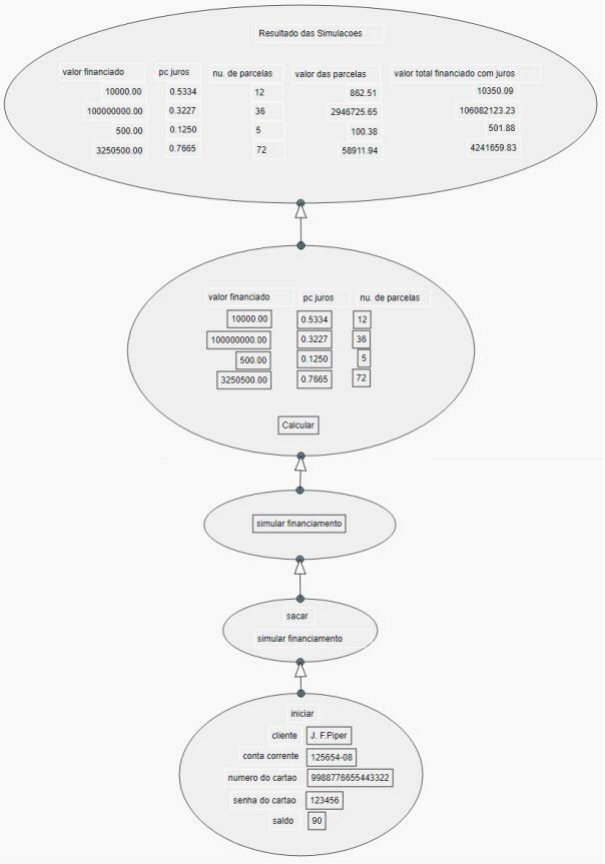


Caso de Uso: Simular financiamento

Cenário de Aceitação: O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa: O valor total a financiar, a taxa de juros (em percentual) e o número de parcelas que deseja pagar. Então, o sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar.

A fórmula matemática para o cálculo do EMI é:

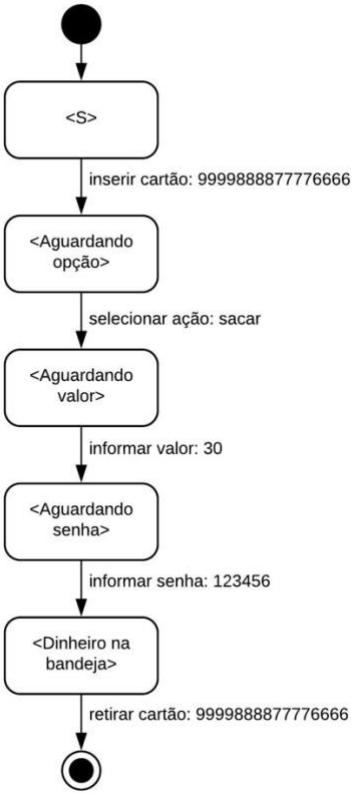
$EMI = P \times R \times \left[\frac{(1+R)^N}{(1+R)^N-1}\right]$, onde: P é o valor total a financiar, R é a taxa de juros e N é o número de parcelas mensais.



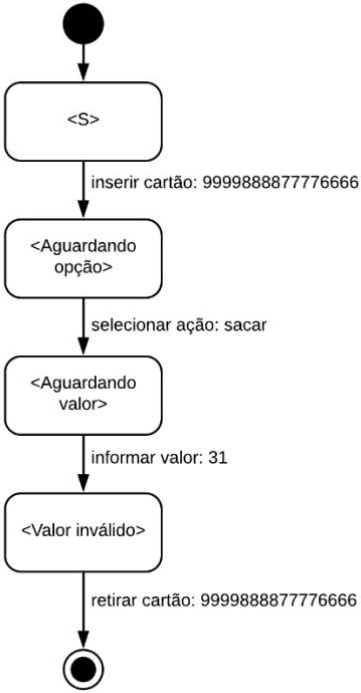
APÊNDICE M – Exemplos de cenários de teste escritos utilizando Máquina de Estados Finita

Caso de Uso: Sacar dinheiro em um caixa eletrônico.

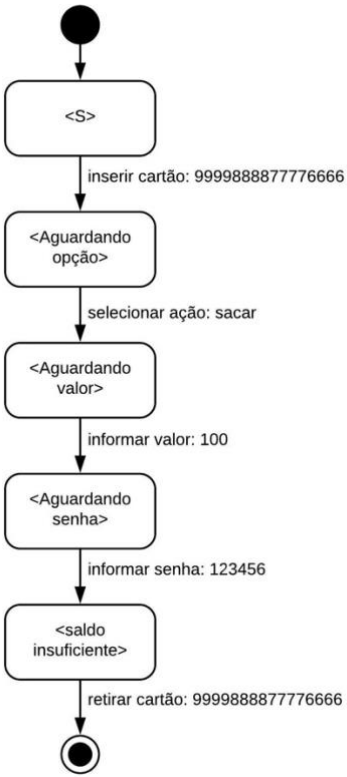
Cenário de Aceitação 1: O saque é realizado com sucesso.



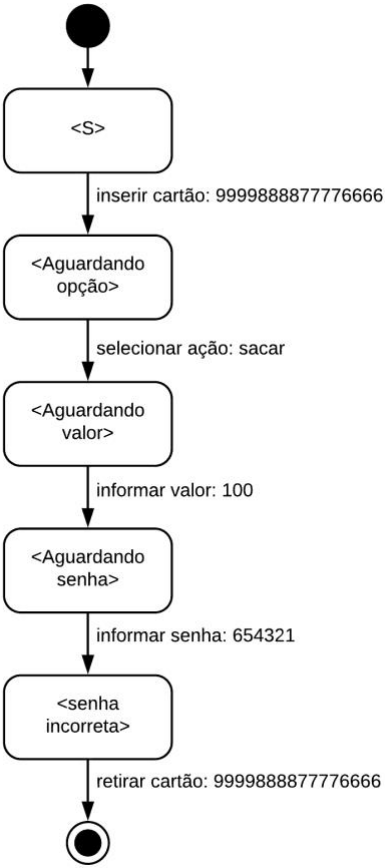
Cenário de Aceitação 2: O saque não é realizado, pois o usuário informou um valor inválido.



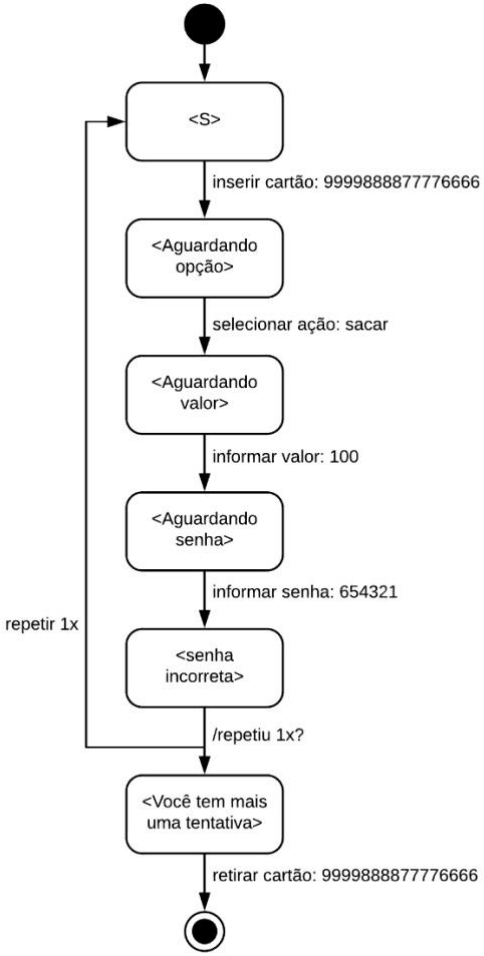
Cenário de Aceitação 3: O saque não é realizado, pois o saldo da conta do usuário é insuficiente.



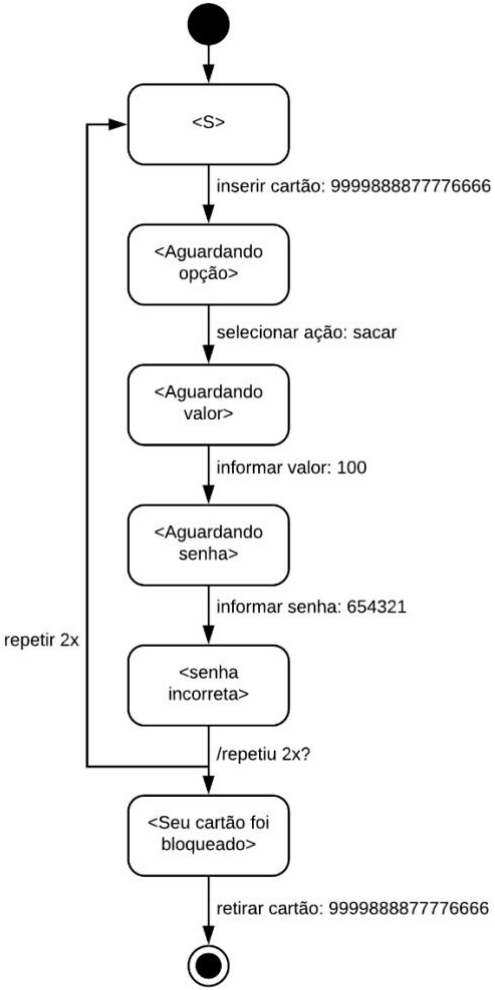
Cenário de Aceitação 4: O saque não é realizado, pois o usuário informou uma senha inválida. É a primeira vez que o usuário informa uma senha inválida nas últimas 72 horas.



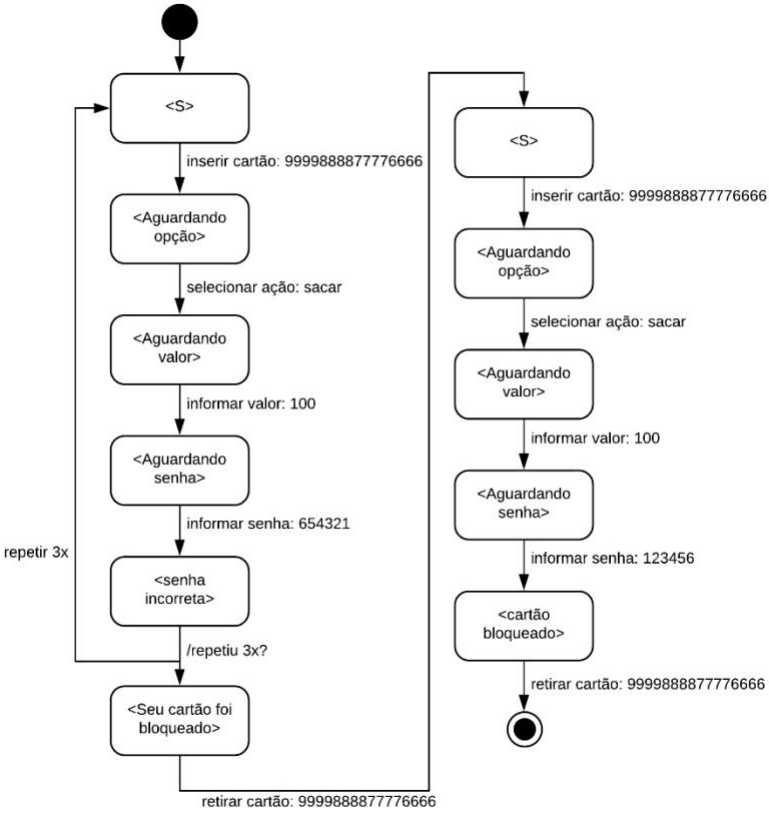
Cenário de Aceitação 5: O saque não é realizado, pois o usuário informou uma senha inválida. É a segunda vez que o usuário informa uma senha inválida nas últimas 72 horas.



Cenário de Aceitação 6: O saque não é realizado, pois o usuário informou uma senha inválida. É a terceira vez que o usuário informa senha inválida nas últimas 72 horas.

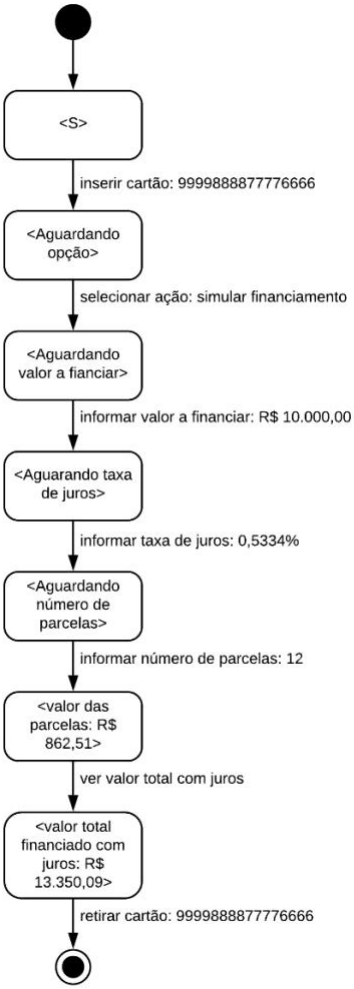


Cenário de Aceitação 7: O saque não é realizado, pois o cartão da conta corrente do usuário está bloqueado.



Caso de Uso: Simular financiamento

Cenário de Aceitação: O Aplicativo deverá possuir um simulador para calcular o valor das parcelas mensais, onde o cliente informa: O valor total a financiar, a taxa de juros (em percentual) e o número de parcelas que deseja pagar. Então, o sistema deverá calcular o EMI - valor das parcelas mensais -, e o valor total com juros que o cliente irá pagar.



APÊNDICE N – Materiais do experimento 1

Este apêndice contém o e-mail e material encaminhado para os participantes do experimento 1.

1 E-MAIL ENCAMINHADO AOS PARTICIPANTES

Olá $NOME_DO_PARTICIPANTE, Boa Tarde!

Muito obrigado por ter optado em participar do experimento da minha dissertação.

Anexo segue um questionário – documento Word – com todas as informações necessárias.

Queria somente ressaltar alguns pontos:

O objetivo do experimento é avaliar 5 técnicas diferentes de teste de aceitação. O seu experimento envolverá somente uma destas técnicas. Logo, se você achar a técnica complicada, difícil de entender, ou se você não entendeu nada: abra seu coração, este feedback é muito importante. O mesmo serve para caso você considere a técnica fácil de entender e usar.

Caso não saiba responder a alguma pergunta do questionário, não tem problema, você pode deixar em branco.

Infelizmente eu não posso te ajudar a responder o questionário ou te dar maiores explicações, isto iria atrapalhar a validade do experimento.

As respostas são anônimas, em nenhum momento seu nome será citado ou referenciado.

Se possível, favor encaminhar o questionário respondido em até 7 dias.

Depois que você enviar as respostas, se tiver interesse em assistir aos vídeos sobre as outras técnicas, é só me avisar que eu te mando os links.

2 QUESTIONÁRIOS

Nesta seção são apresentados os textos dos questionários encaminhados para cada um dos grupos.

2.1 GRUPO A – T01 – Fit/*Fit tables*

Questionário usuários não técnicos/analistas de negócio

Objetivos:

- Verificar a capacidade de comunicação dos requisitos para usuários não técnicos através de testes de aceitação.
- Verificar a capacidade de usuário não-técnicos em especificar testes de aceitação de um requisito de software.

Nas questões a seguir, quando for solicitado uma resposta na escala de 1 a 5 (1-5), considere o que é apresentado a seguir:

1 – Com certeza sim
2 – Sim
3 – Não estou certo
4 – Não
5 – Com certeza não

Nas demais questões, responda o que se pede. Caso você não saiba responder alguma questão, é só deixar em branco. Se quiser colocar observações adicionais, fique à vontade!

Avaliação de Perfil:

1. Qual é a sua formação (Graduação/Especialização/Mestrado/Doutorado) e em qual curso?
2. Qual é a sua área de atuação profissional (Ex.: Administração/Direito/Construção Civil/Psicologia)?
3. Quanto tempo de experiência você tem na sua área de atuação profissional?
4. Você já utilizou testes de aceitação anteriormente? Se sim, qual técnica foi utilizada, em qual área de domínio (ex.: recursos humanos, materiais, compras, etc.) e qual o objetivo (representar e/ou validar os requisitos de software)?

Treinamento:

- Agora, assista ao vídeo sobre "Testes de Aceitação: Introdução": https://youtu.be/yGlVAp4GHkY
- Na sequência, assista ao vídeo "Testes de Aceitação: *Fit Tables*": https://youtu.be/VQ2xjKrqVf4

Compreensão:

Para responder as questões abaixo, veja um exemplo da utilização das *Fit Tables* no documento "*Fit tables* – Caixa Eletrônico" disponível em: http://www.leb.inf.ufsc.br/wp-content/uploads/2019/01/Fit-tables-%E2%80%93-Caixa-Eletr%C3%B4nico.pdf

1. Você compreendeu totalmente o conceito de testes de aceitação? (1-5)
2. Enumere as funcionalidades e regras de negócio que estão especificadas através de testes de aceitação nos exemplos apresentados no documento "*Fit tables* – Caixa Eletrônico".
3. Você teve alguma dificuldade em compreender os casos de testes de aceitação apresentados nos exemplos do documento "*Fit tables* – Caixa Eletrônico"? (1-5)
4. Você tem alguma opinião sobre *Fit tables* que deseja compartilhar? Se sim, qual é a sua opinião?

Aplicação:

No primeiro vídeo, "Testes de Aceitação: Introdução", Olívia solicitou a Brutus' Sistemas o desenvolvimento de um módulo para a Gestão de Programas de Fidelidade. Dentre as funcionalidades do sistema, foi apresentado no vídeo o seguinte requisito: o sistema deverá atribuir 1 ponto, OlivePoint, para cada real gasto pelo cliente referente ao valor total da compra menos o frete. Especifique o teste de aceitação para este requisito utilizando *Fit tables*. Você pode fazer aqui mesmo, neste documento, ou se preferir, pode fazer utilizando papel e lápis/caneta, tirar uma foto, e encaminhar para meu WhatsApp ou e-mail.

1. Você acredita que seus testes de aceitação estão especificados corretamente? (1-5)
2. Você teve dificuldade em especificar requisitos utilizando esta técnica de testes de aceitação? (1-5)
3. Você utilizará esta técnica de testes de aceitação para especificação de requisitos em projetos futuros? (1-5)

Pronto!

Agora envie este questionário respondido para ernani.santos@posgrad.ufsc.br ou para meu WhatsApp +55 48 99900XXXX, se for enviar pelo WhatsApp, envie no privado, por gentileza. Não envie suas respostas para grupos.

2.2 GRUPO B – T08 – BDD/*Gherkin Language*

Questionário usuários não técnicos/analistas de negócio

Objetivos:

- Verificar a capacidade de comunicação dos requisitos para usuários não técnicos através de testes de aceitação.
- Verificar a capacidade de usuário não-técnicos em especificar testes de aceitação de um requisito de software.

Nas questões a seguir, quando for solicitado uma resposta na escala de 1 a 5 (1-5), considere o que é apresentado a seguir:

1 – Com certeza sim
2 – Sim
3 – Não estou certo
4 – Não
5 – Com certeza não

Nas demais questões, responda o que se pede. Caso você não saiba responder alguma questão, é só deixar em branco. Se quiser colocar observações adicionais, fique à vontade!

Avaliação de Perfil:

1. Qual é a sua formação (Graduação/Especialização/Mestrado/Doutorado) e em qual curso?
2. Qual é a sua área de atuação profissional (Ex.: Administração/Direito/Construção Civil/Psicologia)?
3. Quanto tempo de experiência você tem na sua área de atuação profissional?
4. Você já utilizou testes de aceitação anteriormente? Se sim, qual técnica foi utilizada, em qual área de domínio (ex.: recursos humanos, materiais, compras, etc.) e qual o objetivo (representar e/ou validar os requisitos de software)?

Treinamento:

- Agora, assista ao vídeo sobre "Testes de Aceitação: Introdução": https://youtu.be/yGlVAp4GHkY
- Na sequência, assista ao vídeo "Testes de Aceitação: *Gherkin Language*": https://youtu.be/4akMiFRPWxU

Compreensão:

Para responder as questões abaixo, veja um exemplo da utilização da *Gherkin Language* no documento "Gherkin Language – Caixa Eletrônico" disponível em: http://www.leb.inf.ufsc.br/wp-content/uploads/2019/01/Gherkin-Language-%E2%80%93-Caixa-Eletr%C3%B4nico.pdf

1. Você compreendeu totalmente o conceito de testes de aceitação? (1-5)
2. Enumere as funcionalidades e regras de negócio que estão especificadas através de testes de aceitação nos exemplos apresentados no documento "*Gherkin Language* – Caixa Eletrônico".
3. Você teve alguma dificuldade em compreender os casos de testes de aceitação apresentados nos exemplos do documento "*Gherkin Language* – Caixa Eletrônico"? (1-5)
4. Você tem alguma opinião sobre *Gherkin Language* que deseja compartilhar? Se sim, qual é a sua opinião?

Aplicação:

No primeiro vídeo, "Testes de Aceitação: Introdução", Olívia solicitou a Brutus' Sistemas o desenvolvimento de um módulo para a Gestão de Programas de Fidelidade. Dentre as funcionalidades do sistema, foi apresentado no vídeo o seguinte requisito: o sistema deverá atribuir 1 ponto, *OlivePoint*, para cada real gasto pelo cliente referente ao valor total da compra menos o frete. Especifique o teste de aceitação para este requisito utilizando *Gherkin Language*. Você pode fazer aqui mesmo, neste documento, ou se preferir, pode fazer utilizando papel e lápis/caneta, tirar uma foto, e encaminhar para meu *WhatsApp* ou e-mail.

1. Você acredita que seus testes de aceitação estão especificados corretamente? (1-5)
2. Você teve dificuldade em especificar requisitos utilizando esta técnica de testes de aceitação? (1-5)
3. Você utilizará esta técnica de testes de aceitação para especificação de requisitos em projetos futuros? (1-5)

Pronto!

Agora envie este questionário respondido para ernani.santos@posgrad.ufsc.br ou para meu *WhatsApp* +55 48 99900XXXX, se for enviar pelo *WhatsApp*, envie no privado, por gentileza. Não envie suas respostas para grupos.

2.3 GRUPO C – T06 – Histórias de usuário com exemplos e marcações

Questionário usuários não técnicos/analistas de negócio

Objetivos:

- Verificar a capacidade de comunicação dos requisitos para usuários não técnicos através de testes de aceitação.
- Verificar a capacidade de usuário não-técnicos em especificar testes de aceitação de um requisito de software.

Nas questões a seguir, quando for solicitado uma resposta na escala de 1 a 5 (1-5), considere o que é apresentado a seguir:

1 – Com certeza sim
2 – Sim
3 – Não estou certo
4 – Não
5 – Com certeza não

Nas demais questões, responda o que se pede. Caso você não saiba responder alguma questão, é só deixar em branco. Se quiser colocar observações adicionais, fique à vontade!

Avaliação de Perfil:

1. Qual é a sua formação (Graduação/Especialização/Mestrado/Doutorado) e em qual curso?
2. Qual é a sua área de atuação profissional (Ex.: Administração/Direito/Construção Civil/Psicologia)?
3. Quanto tempo de experiência você tem na sua área de atuação profissional?
4. Você já utilizou testes de aceitação anteriormente? Se sim, qual técnica foi utilizada, em qual área de domínio (ex.: recursos humanos, materiais, compras, etc.) e qual o objetivo (representar e/ou validar os requisitos de software)?

Treinamento:

- Agora, assista ao vídeo sobre "Testes de Aceitação: Introdução": https://youtu.be/yGlVAp4GHkY
- Na sequência, assista ao vídeo "Testes de Aceitação: Histórias de usuário com exemplos": https://youtu.be/1YUc3okAdNM

Compreensão:

Para responder as questões abaixo, veja um exemplo da utilização das Histórias de Usuário com Exemplos no documento "Histórias de usuário com exemplo – Caixa Eletrônico" disponível em: http://www.leb.inf.ufsc.br/wp-content/uploads/2019/01/Hist%C3%B3rias-de-Usu%C3%A1rio-com-Exemplos-%E2%80%93-Caixa-Eletr%C3%B4nico.pdf

1. Você compreendeu totalmente o conceito de testes de aceitação? (1-5)
2. Enumere as funcionalidades e regras de negócio que estão especificadas através de testes de aceitação nos exemplos apresentados no documento "Histórias de usuário com exemplo – Caixa Eletrônico".
3. Você teve alguma dificuldade em compreender os casos de testes de aceitação apresentados nos exemplos do documento "Histórias de usuário com exemplo – Caixa Eletrônico"? (1-5)
4. Você tem alguma opinião sobre Histórias de usuário com exemplos que deseja compartilhar? Se sim, qual é a sua opinião?

Aplicação:

No primeiro vídeo, "Testes de Aceitação: Introdução", Olívia solicitou a Brutus' Sistemas o desenvolvimento de um módulo para a Gestão de Programas de Fidelidade. Dentre as funcionalidades do sistema, foi apresentado no vídeo o seguinte requisito: o sistema deverá atribuir 1 ponto, *OlivePoint*, para cada real gasto pelo cliente referente ao valor total da compra menos o frete. Especifique o teste de aceitação para este requisito utilizando *Fit tables*. Você pode fazer aqui mesmo, neste documento, ou se preferir, pode fazer utilizando papel e lápis/caneta, tirar uma foto, e encaminhar para meu *WhatsApp* ou e-mail.

4. Você acredita que seus testes de aceitação estão especificados corretamente? (1-5)
5. Você teve dificuldade em especificar requisitos utilizando esta técnica de testes de aceitação? (1-5)
6. Você utilizará esta técnica de testes de aceitação para especificação de requisitos em projetos futuros? (1-5)

Pronto!

Agora envie este questionário respondido para ernani.santos@posgrad.ufsc.br ou para meu *WhatsApp* +55 48 99900XXXX, se for enviar pelo *WhatsApp*, envie no privado, por gentileza. Não envie suas respostas para grupos.

2.4 GRUPO D – T09 – *EasyAccept*

Questionário usuários não técnicos/analistas de negócio

Objetivos:

- Verificar a capacidade de comunicação dos requisitos para usuários não técnicos através de testes de aceitação.
- Verificar a capacidade de usuário não-técnicos em especificar testes de aceitação de um requisito de software.

Nas questões a seguir, quando for solicitado uma resposta na escala de 1 a 5 (1-5), considere o que é apresentado a seguir:

1 – Com certeza sim
2 – Sim
3 – Não estou certo
4 – Não
5 – Com certeza não

Nas demais questões, responda o que se pede. Caso você não saiba responder alguma questão, é só deixar em branco. Se quiser colocar observações adicionais, fique à vontade!

Avaliação de Perfil:

1. Qual é a sua formação (Graduação/Especialização/Mestrado/Doutorado) e em qual curso?
2. Qual é a sua área de atuação profissional (Ex.: Administração/Direito/Construção Civil/Psicologia)?
3. Quanto tempo de experiência você tem na sua área de atuação profissional?
4. Você já utilizou testes de aceitação anteriormente? Se sim, qual técnica foi utilizada, em qual área de domínio (ex.: recursos humanos, materiais, compras, etc.) e qual o objetivo (representar e/ou validar os requisitos de software)?

Treinamento:

- Agora, assista ao vídeo sobre "Testes de Aceitação: Introdução": https://youtu.be/yGlVAp4GHkY
- Na sequência, assista ao vídeo "Testes de Aceitação: *EasyAccept*": https://youtu.be/Tzbi-4Lim3Q

Compreensão:

Para responder as questões abaixo, veja um exemplo da utilização das *EasyAccept* no documento "*EasyAccept* – Caixa Eletrônico" disponível em: http://www.leb.inf.ufsc.br/wp-content/uploads/2019/01/EasyAccept-%E2%80%93-Caixa-Eletr%C3%B4nico.pdf

1. Você compreendeu totalmente o conceito de testes de aceitação? (1-5)
2. Enumere as funcionalidades e regras de negócio que estão especificadas através de testes de aceitação nos exemplos apresentados no documento "*EasyAccept* – Caixa Eletrônico".
3. Você teve alguma dificuldade em compreender os casos de testes de aceitação apresentados nos exemplos do documento "*EasyAccept* – Caixa Eletrônico"? (1-5)
4. Você tem alguma opinião sobre *EasyAccept* que deseja compartilhar? Se sim, qual é a sua opinião?

Aplicação:

No primeiro vídeo, "Testes de Aceitação: Introdução", Olívia solicitou a Brutus' Sistemas o desenvolvimento de um módulo para a Gestão de Programas de Fidelidade. Dentre as funcionalidades do sistema, foi apresentado no vídeo o seguinte requisito: o sistema deverá atribuir 1 ponto, *OlivePoint*, para cada real gasto pelo cliente referente ao valor total da compra menos o frete. Especifique o teste de aceitação para este requisito utilizando *EasyAccept*. Você pode fazer aqui mesmo, neste documento, ou se preferir, pode fazer utilizando papel e lápis/caneta, tirar uma foto, e encaminhar para meu *WhatsApp* ou e-mail.

1. Você acredita que seus testes de aceitação estão especificados corretamente? (1-5)
2. Você teve dificuldade em especificar requisitos utilizando esta técnica de testes de aceitação? (1-5)
3. Você utilizará esta técnica de testes de aceitação para especificação de requisitos em projetos futuros? (1-5)

Pronto!

Agora envie este questionário respondido para ernani.santos@posgrad.ufsc.br ou para meu *WhatsApp* +55 48 99900XXXX, se for enviar pelo *WhatsApp*, envie no privado, por gentileza. Não envie suas respostas para grupos.

2.5 GRUPO E – T11 – US-UID

Questionário usuários não técnicos/analistas de negócio

Objetivos:

- Verificar a capacidade de comunicação dos requisitos para usuários não técnicos através de testes de aceitação.
- Verificar a capacidade de usuário não-técnicos em especificar testes de aceitação de um requisito de software.

Nas questões a seguir, quando for solicitado uma resposta na escala de 1 a 5 (1-5), considere o que é apresentado a seguir:

1 – Com certeza sim
2 – Sim
3 – Não estou certo
4 – Não
5 – Com certeza não

Nas demais questões, responda o que se pede. Caso você não saiba responder alguma questão, é só deixar em branco. Se quiser colocar observações adicionais, fique à vontade!

Avaliação de Perfil:

1. Qual é a sua formação (Graduação/Especialização/Mestrado/Doutorado) e em qual curso?
2. Qual é a sua área de atuação profissional (Ex.: Administração/Direito/Construção Civil/Psicologia)?
3. Quanto tempo de experiência você tem na sua área de atuação profissional?
4. Você já utilizou testes de aceitação anteriormente? Se sim, qual técnica foi utilizada, em qual área de domínio (ex.: recursos humanos, materiais, compras, etc.) e qual o objetivo (representar e/ou validar os requisitos de software)?

Treinamento:

- Agora, assista ao vídeo sobre "Testes de Aceitação: Introdução": https://youtu.be/yGlVAp4GHkY
- Na sequência, assista ao vídeo "Testes de Aceitação: US-UID": https://youtu.be/US3QOIAoZ5w

Compreensão:

Para responder as questões abaixo, veja um exemplo da utilização das US-UID no documento "US-UID – Caixa Eletrônico" disponível em: http://www.leb.inf.ufsc.br/wp-content/uploads/2019/01/US-UID-%E2%80%93-Caixa-Eletr%C3%B4nico.pdf

1. Você compreendeu totalmente o conceito de testes de aceitação? (1-5)
2. Enumere as funcionalidades e regras de negócio que estão especificadas através de testes de aceitação nos exemplos apresentados no documento "US-UID– Caixa Eletrônico".
3. Você teve alguma dificuldade em compreender os casos de testes de aceitação apresentados nos exemplos do documento "US-UID – Caixa Eletrônico".? (1-5)
4. Você tem alguma opinião sobre UD-UID que deseja compartilhar? Se sim, qual é a sua opinião?

Aplicação:

No primeiro vídeo, "Testes de Aceitação: Introdução", Olívia solicitou a Brutus' Sistemas o desenvolvimento de um módulo para a Gestão de Programas de Fidelidade. Dentre as funcionalidades do sistema, foi apresentado no vídeo o seguinte requisito: o sistema deverá atribuir 1 ponto, *OlivePoint*, para cada real gasto pelo cliente referente ao valor total da compra menos o frete. Especifique o teste de aceitação para este requisito utilizando US-UID. Você pode fazer aqui mesmo, neste documento, ou se preferir, pode fazer utilizando papel e lápis/caneta, tirar uma foto, e encaminhar para meu *WhatsApp* ou e-mail.

1. Você acredita que seus testes de aceitação estão especificados corretamente? (1-5)

2. Você teve dificuldade em especificar requisitos utilizando esta técnica de testes de aceitação? (1-5)
3. Você utilizará esta técnica de testes de aceitação para especificação de requisitos em projetos futuros? (1-5)

Pronto!

Agora envie este questionário respondido para ernani.santos@posgrad.ufsc.br ou para meu *WhatsApp* +55 48 99900XXXX, se for enviar pelo *WhatsApp*, envie no privado, por gentileza. Não envie suas respostas para grupos.

APÊNDICE O – Questionário do Experimento 4

Nesta seção é apresentado o questionário utilizado no experimento do Capítulo 6 da dissertação. Os demais artefatos utilizados no experimento estão disponíveis em www.leb.inf.ufsc.br/index.php/xp2018/.

1 QUESTIONÁRIO

Universidade Federal de Santa Catarina
INE5448 – Tópicos Especiais em Aplicações Tecnológicas I
Testes de Aceitação

Aluno:

Idade:

Há quantos anos e meses você curso Ciência da Computação na UFSC?

Responda as questões a seguir de acordo com a escala abaixo.

1. Pouco ou Nenhum - Nenhuma experiência anterior, ou experiência de até 1 ano.
2. Básico – De 1 a 2 anos de experiência.
3. Médio – De 2 a 5 anos de experiência.
4. Alto - De 5 a 10 anos de experiência.
5. Avançado – 5 anos de experiência.

1. Como você avalia sua experiência prática com Projetos de Desenvolvimento de Software? (1-5)
2. Como você avalia sua experiência prática com Programação Java? (1-5)
3. Como você avalia sua experiência prática com Testes de Aceitação? (1-5)
4. Como você avalia sua experiência prática com Fit? (1-5)
5. Como você avalia sua experiência prática com Cucumber? (1-5)
6. Como você avalia sua experiência prática na elicitação e documentação de requisitos de software? (1-5)

Atividade I

O objetivo desta atividade é avaliar a capacidade do aluno em especificar Testes de Aceitação utilizando uma das técnicas vistas em sala de aula. Este exercício é composto por duas *Tasks*. Em cada uma das *Taks* será especificado os Testes de Aceitação para uma funcionalidade do sistema.

Atenção, você deverá preencher nos quadros abaixo o horário que iniciou e terminou o desenvolvimento de cada uma das *Tasks*. Caso, no desenvolvimento da *Task* você precise realizar pausas para, por exemplo, ajustar configurações na máquina não relacionadas a atividade, ir ao banheiro ou ajudar um colega, você deverá indicar o horário de início e fim de cada uma das pausas. O objetivo é saber quanto tempo você utilizou para especificar os testes de aceitação da funcionalidade.

Task 1

Horário Início

Horário Fim

Pausas

Horário Início Pausa

Horário Fim Pausa

Task 2

Horário Início	*Horário Fim*

Pausas

Horário Início Pausa	*Horário Fim Pausa*

Após finalizar as tarefas da atividade I, responda as questões a seguir:

Qual técnica de Testes de Aceitação você utilizou nas *Taks* do Exercício I?				
□ Cucumber			□ Fit	
Sobre a ***Task I***, qual grau de dificuldade você atribui para especificar o teste de aceitação para esta *task?*				
□ Muito Fácil	□ Fácil	□ Médio	□ Difícil	□ Muito Difícil
Sobre a ***Task II,*** qual grau de dificuldade você atribui para especificar o teste de aceitação para esta *task?*				
□ Muito Fácil	□ Fácil	□ Médio	□ Difícil	□ Muito Difícil

Sobre o resultado da ***Task I***, como você avalia os testes produzidos por você?				
□ Muito Ruim	□ Ruim	□ Médio	□ Bom	□ Muito Bom
Sobre o resultado da ***Task II***, como você avalia os testes produzidos por você?				
□ Muito Ruim	□ Ruim	□ Médio	□ Bom	□ Muito Bom
Como você avalia o detalhamento da funcionalidade fornecido no enunciado da ***Task I***?				
□ Muito Ruim	□ Ruim	□ Médio	□ Bom	□ Muito Bom
Como você avalia o detalhamento da funcionalidade fornecido no enunciado da ***Task II***?				
□ Muito Ruim	□ Ruim	□ Médio	□ Bom	□ Muito Bom
Você utilizaria esta técnica de testes de aceitação em um projeto futuro?				
□ Com certeza não	□ Não	□ Talvez	□ Sim	□ Com certeza sim

Atividade II

O objetivo da Atividade II é avaliar a capacidade do aluno em entender testes de aceitação e codificar em Java as funcionalidades de um sistema a partir deles, bem como, desenvolver os códigos de cola (*Glue code* ou *Fixtures*) para executar os testes de aceitação e validar a implementação. Este exercício é composto por duas *Tasks*. Em cada uma das *Taks* são fornecidas especificações dos Testes de Aceitação para uma funcionalidade do sistema.

Você deverá preencher nos quadros abaixo o horário que iniciou e terminou o desenvolvimento de cada uma das *Tasks*. Caso, no desenvolvimento da *Task* você precise realizar pausas para, por exemplo, ajustar configurações na máquina não relacionadas a atividade, ir ao banheiro ou ajudar um colega, você deverá indicar o horário de início e fim de cada uma das pausas. O objetivo é saber quanto tempo você utilizou para codificar as funcionalidades especificadas através dos testes de aceitação.

Task 1

Horário Início	*Horário Fim*

Pausas

Horário Início Pausa	*Horário Fim Pausa*

Task 2

Horário Início	*Horário Fim*

Pausas

Horário Início Pausa	*Horário Fim Pausa*

Atenção, antes de iniciar a codificação detalhada na Atividade II, avalie os Testes de Aceitação entregues pelo colega e responda as duas questões abaixo.

Como você avalia os testes de aceitação entregues para o desenvolvimento da funcionalidade referente a ***Task I***? □ Muito Ruim □ Ruim □ Médio □ Bom □ Muito Bom
Como você avalia os testes de aceitação entregues para o desenvolvimento da funcionalidade referente a ***Task II***? □ Muito Ruim □ Ruim □ Médio □ Bom □ Muito Bom

Após finalizar o Exercício II, responda as questões a seguir:

Qual técnica de Testes de Aceitação você utilizou nas *Taks* do Exercício II □ Cucumber □ Fit
Sobre a ***Task I***, qual grau de dificuldade você atribui para codificar a funcionalidade especificada pelo teste de aceitação? □ Muito Fácil □ Fácil □ Médio □ Difícil □ Muito Difícil
Sobre a ***Task II***, qual grau de dificuldade você atribui para codificar a funcionalidade especificada pelo teste de aceitação? □ Muito Fácil □ Fácil □ Médio □ Difícil □ Muito Difícil

Sobre o resultado da ***Task I***, como você avalia os códigos-fonte produzidos por você? □ Muito Ruim □ Ruim □ Médio □ Bom □ Muito Bom
Sobre o resultado da ***Task II***, como você avalia os códigos-fonte produzidos por você? □ Muito Ruim □ Ruim □ Médio □ Bom □ Muito Bom
Como você avalia os testes de aceitação entregues para o desenvolvimento da funcionalidade referente a ***Task I***? □ Muito Ruim □ Ruim □ Médio □ Bom □ Muito Bom
Como você avalia os testes de aceitação entregues para o desenvolvimento da funcionalidade referente a ***Task II***? □ Muito Ruim □ Ruim □ Médio □ Bom □ Muito Bom
Você utilizaria esta técnica de testes de aceitação como ferramenta para especificação de requisitos em um projeto futuro? □ Com certeza não □ Não □ Talvez □ Sim □ Com certeza sim