UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

FERRAMENTA DE APOIO A TESTE DE SOFTWARE BASEADO NA TABELA DE DECISÃO

DESIRÉE FERNANDA HOPPE

BLUMENAU
2015

2015/2-03

DESIRÉE FERNANDA HOPPE

FERRAMENTA DE APOIO A TESTE DE SOFTWARE BASEADO NA TABELA DE DECISÃO

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Marcel Hugo, Mestre – Orientador

BLUMENAU
2015

2015/2-03

FERRAMENTA DE APOIO A TESTE DE SOFTWARE

BASEADO NA TABELA DE DECISÃO

Por

DESIRÉE FERNANDA HOPPE

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Marcel Hugo, Mestre – Orientador, FURB
Membro:	Prof. Matheus Carvalho Viana, Doutor – FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB

Blumenau, 08 de dezembro de 2015

Dedico este trabalho aos meus pais, por todo o apoio e incentivo.

AGRADECIMENTOS

A Deus, por ter me guiado até aqui.

À minha família, por sempre apoiar minhas decisões.

Ao meu orientador Marcel Hugo, por ter acreditado na conclusão deste trabalho.

À minha gestora Silva Aparecida da Silva, por me auxiliar na tomada das decisões, visando maior aplicabilidade para o trabalho.

Há uma força motriz mais poderosa que o vapor, a eletricidade e a energia atômica: a vontade.

Albert Einstein

RESUMO

O desenvolvimento de sistemas mais complexos fez com que as empresas de tecnologia aumentassem o investimento em teste de software. Por conta disto, tornou-se fundamental incluir as atividades relacionadas a teste de software no processo de desenvolvimento. Também foi necessário capacitar pessoas para validar os requisitos e verificar o funcionamento das aplicações, garantindo o funcionamento correto dos sistemas. Porém as atividades relacionadas ao processo de teste podem representar um alto custo para a empresa e, por isso, é necessário automatizar partes deste processo. Baseando-se nestas informações, este trabalho foi desenvolvido com o objetivo de criar uma ferramenta de apoio ao teste de software a partir do uso de tabelas de decisão. A automatização da criação de tabelas de decisão reduz o tempo gasto pelo analista de teste nessa atividade e cria artefatos de teste mais abrangentes, reduzindo o risco de falha humana. A ferramenta foi desenvolvida utilizando as linguagens Java Server Pages (JSP), JavaScript e Java, o banco de dados MySQL para a persistência de dados e o Apache Tomcat para servidor web. A *Integrated Development Environment* (IDE) utilizada no desenvolvimento do sistema foi o Eclipse Mars. A ferramenta obteve como resultado a criação de tabelas de decisão completas e reduzidas, a partir das variáveis e regras informadas pelo usuário. Além disto, a ferramenta permite gerar os casos de teste a partir das condições criadas na tabela. A ferramenta foi validada por potenciais usuários, que concluíram que os requisitos propostos e as regras de negócio definidas foram atendidos.

Palavras-chave: Teste de software. Tabela de decisão. Casos de teste.

ABSTRACT

The development of complex systems made technology companies to increase investment in software testing. To ensure the correct operation of applications, it became essential to include the activities related to software testing in development process, in addition to enable people to validate application requirements and test application operation. However, the activities related to software testing process can represent a high cost to the company and, because of that, it is necessary automate parts of this process. According to this information, this work has been carried out in order to create a support tool for software testing based on decision tables. Automating the creation of decision tables reduces the time spent by the test analyst in this activity and create more comprehensive test artifacts, reducing the risk of human fault. The tool was developed using Java Server Pages (JSP), JavaScript and Java languages, MySQL Database for data persistence and Apache Tomcat Application Server. The Integrated Development Environment (IDE) used in the system development was Eclipse Mars. The result reached by the tool is the creation of completed and reduced decision tables from the variables and rules provided by the user. Moreover, the developed tool allows generating test cases from conditions created in the table. The tool has been validated by potential users, which concluded that the proposed requirements and the defined business rules have been met.

Key-words: Software testing. Decision table. Test cases.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

GWT – *Google Web Toolkit*

IDE – *Integrated Development Environment*

JSP – Java Server Pages

TI – Tecnologia da Informação

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o investimento por parte das empresas de Tecnologia da Informação (TI) em testes de software aumentou consideravelmente. Inicialmente, esse processo era executado pela própria equipe de desenvolvimento. Hoje existem equipes focadas exclusivamente em testes de software, com o objetivo de garantir o funcionamento correto do sistema. Isso acontece, pois, segundo Rios e Moreira Filho (2013, p. 9), "(...) houve uma mudança significativa na abrangência e complexidade das aplicações, onde fatores, tais como segurança e desempenho, passam a ser relevantes, tornando a atividade de testar cada vez mais especializada".

Com o crescimento das equipes de testes em empresas de TI, tornou-se necessária a criação de processos mais robustos, permitindo que estes profissionais executem suas funções com maior precisão. "A indústria tem despertado para a extrema importância da atividade de teste que, por um lado, pode contribuir para a melhoria da qualidade de um determinado produto e, por outro, representar um custo significativo dentro dos orçamentos empresariais" (DELAMARO; MALDONADO; JINO, 2007, p.).

Ao longo dos anos, diversas técnicas de teste foram desenvolvidas para garantir que as especificações e os requisitos de um sistema sejam atendidos. Esse processo inclui a criação de cenários de testes para avaliar as funcionalidades da aplicação, validando se o que foi especificado foi desenvolvido corretamente (HEINEBERG, 2008). Dentre as técnicas, destacam-se os testes por tabela de decisão, cujo objetivo é elaborar casos de testes que representem regras e ações a partir de uma tabela (MOLINARI, 2008, p.155).

De acordo com Delamaro, Maldonado e Jino (2007, p. 7), "um ponto importante para o sucesso no teste de um software é a automatização". Esses autores também comentam que técnicas para a geração automática de casos de teste são muito desejadas, permitindo também a automação do processo de testes. Apesar disso, ainda existem poucos investimentos em ferramentas que automatizem atividades desta área.

A necessidade em automatizar as atividades relacionadas ao processo de teste, aliada ao aumento no número de profissionais nesta área, mostra que muitas melhorias podem ser desenvolvidas para aprimorar o trabalho dessas equipes. Dentro do contexto apresentado, a ferramenta desenvolvida neste trabalho gera a tabela de decisão utilizando variáveis e regras definidas pelo usuário. Após a geração da tabela, a ferramenta analisa as condições de testes geradas e permite ao usuário excluir as condições consideradas equivalentes. Além disso, a

ferramenta permite gerar os casos de teste a partir da tabela, utilizando um modelo previamente definido pelo usuário.

1.1 PROBLEMA

As tabelas de decisão são criadas para analisar as combinações entre as variáveis e facilitar a criação dos casos de testes. Devido à falta de ferramentas especializadas em auxiliar atividades de análise de teste, muitas empresas executam esses processos de forma manual, incluindo a criação de tabelas de decisão.

Criar tabelas de decisão manualmente exige muito tempo e é um processo bastante complexo, pois os sistemas possuem muitas possibilidades de testes. Até mesmo em sistemas de baixa complexidade, em que existe apenas uma pequena quantidade de estados por variáveis, a quantidade total de possibilidades a serem abrangidas em casos de testes é muito grande (PRIMESOFT, 2014). Além disto, a criação manual também apresenta um risco considerável de não abranger todas as combinações necessárias para os testes.

1.2 JUSTIFICATIVA

O desenvolvimento de sistemas mais complexos exige que casos de testes mais abrangentes sejam criados. Porém a busca por produtividade nas empresas de TI exige, também, que os processos sejam executados com maior rapidez. Com isso, percebe-se a necessidade em automatizar a criação de tabelas de decisão, diminuindo o tempo gasto pelo analista de testes nessa atividade.

Com base nessas informações, a ferramenta desenvolvida neste trabalho gera a tabela de decisão a partir de variáveis e regras definidas pelo usuário, contemplando todas as combinações possíveis e apresentando uma ação para cada combinação. Após a tabela gerada, a ferramenta permite que o usuário exclua os testes considerados equivalentes e exporte a tabela para posterior acompanhamento. A consulta às informações geradas na tabela auxilia o analista de teste na criação manual dos casos de teste, mas a ferramenta também possibilita ao analista de teste gerar os casos de testes a partir de um modelo padrão que ele pode cadastrar no sistema.

1.3 OBJETIVOS

O objetivo geral do trabalho é o desenvolvimento de uma ferramenta de apoio ao teste de software a partir do uso de tabelas de decisão.

Os objetivos específicos do trabalho proposto são:

a) automatizar a criação das tabelas de decisão, utilizando variáveis e regras definidas pelo usuário;
b) facilitar a identificação das condições equivalentes dentro da tabela de decisão;
c) facilitar a criação de casos de teste por meio da consulta às informações da tabela gerada;
d) permitir gerar os casos de teste a partir da tabela de decisão, utilizando um modelo padrão;
e) permitir exportar os artefatos gerados para arquivos nos formatos: pdf e xlsx.

1.4 ESTRUTURA

Este trabalho está estruturado em quatro capítulos.

O primeiro capítulo apresenta a introdução, o problema, a justificativa, os objetivos geral e específicos e a estrutura do trabalho.

O segundo capítulo aborda assuntos relacionados a qualidade de software, apresentando técnicas de análise e de teste de software, descrevendo o funcionamento de tabelas de decisão e detalhando os trabalhos correlatos ao trabalho desenvolvido.

O terceiro capítulo apresenta o desenvolvimento da ferramenta, sua estrutura, interface, a metodologia utilizada e a aplicabilidade da ferramenta, além dos resultados obtidos.

O quarto capítulo contém a conclusão obtida no desenvolvimento do trabalho e as sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os temas aplicados no desenvolvimento do trabalho, tais como Qualidade e Teste de Software, Análise de Testes, Tabela de Decisão e trabalhos correlatos.

2.1 QUALIDADE E TESTE DE SOFTWARE

Existem muitas maneiras de definir teste de software. Porém, em essência, teste de software pode ser definido como o processo de execução do software de maneira controlada, visando avaliar se o comportamento obtido está de acordo com o que foi especificado (RIOS, 2010, p. 232). Assim sendo, o principal objetivo da atividade de teste de software é garantir que o produto que está sendo desenvolvido e o formato utilizado em seu desenvolvimento estejam em conformidade com a especificação (DELAMARO; MALDONADO; JINO, 2007, p. 1).

Segundo Pezzè e Young (2008, p. 64), a estrutura do processo de qualidade deve buscar a completude, identificando oportunidades e prezando pelo custo-benefício. Ou seja, é necessário que este processo seja completo, de modo que as falhas possam ser identificadas o quanto antes, gerando o menor custo possível. Um processo de qualidade bem elaborado deve selecionar e organizar atividades ao longo de todo o processo de desenvolvimento, aumentando a eficiência das atividades e respeitando o custo-benefício (PEZZÈ; YOUNG, 2008, p. 65).

As atividades específicas da garantia de qualidade costumam ser agrupadas no processo de qualidade. Apesar disso, é necessário reconhecer que a qualidade deve estar presente em outras atividades do processo (PEZZÈ; YOUNG, 2008, p. 63). Para que isso ocorra, é aconselhável que as atividades de testes sejam conduzidas durante todo o processo de desenvolvimento do software, englobando diferentes técnicas (DELAMARO; MALDONADO; JINO, 2007, p. 2). De acordo com Pezzè e Young (2008, p. 63) “a qualidade de software resulta de um conjunto completo de atividades interdependentes, entre as quais a análise e os testes são necessários, mas estão longe de ser suficientes”.

Ao contrário do que se pensa, o principal objetivo da atividade de teste de software não é mostrar que um sistema está correto, mas sim detectar a presença de defeitos, caso eles existam (DELAMARO; MALDONADO; JINO, 2007, p. 6). “Quando a atividade de teste é realizada de maneira criteriosa e embasada tecnicamente, o que se tem é uma certa ‘confiança’ de que o software se comporta corretamente para grande parte do seu domínio de entrada” (DELAMARO; MALDONADO; JINO, 2007, p. 6).

Ao longo dos anos, muitas técnicas foram desenvolvidas para auxiliar a execução dos testes. Os testes de software podem ser divididos em duas técnicas: testes de caixa preta e testes de caixa branca. Os testes de caixa preta, também chamados de testes funcionais, validam as funcionalidades do software e garantem a aderência aos requisitos, sem se basear no código ou na estrutura interna do sistema (RIOS, 2010, p. 353). Os testes de caixa branca avaliam o comportamento interno do software, validando o código, os componentes, as configurações e outros elementos técnicos (RIOS, 2010, p. 353).

Molinari (2008, p. 58) define teste funcional como “o mais importante teste, pois ele se caracteriza por mostrar se a aplicação funciona ou não em tudo aquilo que ela se propõe a atender em termos de funcionalidades.” Os testes funcionais podem ser executados em diferentes níveis, sendo que os mais comuns são os testes de unidade, de integração, de sistema e de aceitação do usuário (MOLINARI, 2008, p. 149).

2.2 ANÁLISE DE TESTE

Antes de iniciar a execução dos testes, é necessário analisar os requisitos do sistema e criar os artefatos de análise de teste. Rezende (2005, p. 269) afirma que existe um grande aumento de produtividade ao executar os testes seguindo um roteiro ou especificação. “Na prática, torna-se muito difícil testar sem requerimentos, mas não é impossível” (MOLINARI, 2008, p. 41).

De acordo com Delamaro, Maldonado e Jino (2007, p. 9), é possível encontrar todos os defeitos de um software por meio de testes funcionais, submetendo o sistema a todos os valores de entrada possíveis, porém o domínio de entrada pode ser muito grande, inviabilizando essa alternativa. A elaboração da análise dos testes torna essa atividade possível, pois divide todos os possíveis valores de entrada do software em subconjuntos que tenham grande probabilidade de encontrar os erros (DELAMARO; MALDONADO; JINO, 2007, p. 5).

Uma das maiores dificuldades dentro do processo de teste é transcrever o que deve ser testado e qual será a abrangência do teste (MOLINARI, 2008, p. 127). Por conta disto, foram desenvolvidas técnicas para auxiliar a análise dos testes, sem envolver a execução do sistema. De acordo com Pezzè e Young (2008, p. 71), “técnicas de análise, que não envolvem a execução real do código fonte do programa, desempenham um papel importante nos processos gerais de qualidade de software”.

Dentre as diversas técnicas utilizadas, destacam-se os testes baseados em modelos. O objetivo desse teste é criar especificações de casos de teste utilizando como modelo o

comportamento esperado e revelando as diferenças existentes entre a modelagem e o comportamento real do sistema (PEZZÈ; YOUNG, 2008, p. 266). Os testes baseados em modelos abrangem os testes de estruturas de decisão, nos quais a técnica de tabela de decisão está inserida.

A tabela de decisão serve para modelar os testes baseando-se nas regras de negócio (VIEIRA, 2010) e costuma ser o primeiro artefato de análise de teste a ser criado. Molinari (2008, p. 156) explica que, após a tabela ser definida e estabilizada, ela se transforma em casos de teste. O caso de teste é o principal artefato de análise de teste e representa, detalhadamente, o que deve ser testado (MOLINARI, 2008, p. 56). Rios (2005, p.117) explica que os casos de teste descrevem o teste que será executado, com foco nos objetivos, e costumam estar vinculados aos casos de uso do sistema.

2.3 TABELA DE DECISÃO

A tabela de decisão é uma técnica de análise de teste, que tem por objetivo apresentar combinações de situações que devem ser verdadeiras para que uma ação específica seja tomada. (RIOS, 2005, p.111). De acordo com Campos (2009), a tabela de decisão é uma boa maneira de lidar com combinações de valores de entrada, pois combina as variáveis de impacto retiradas da análise de negócio, formando condições que deverão ser testadas.

A tabela de decisão está incluída no grupo de técnicas de teste do tipo funcional, em que o objetivo é testar diretamente a aplicação, analisando as entradas e saídas (MOLINARI, 2005, p. 149). Além disso, a tabela de decisão é uma técnica baseada em modelo, ou seja, a criação da tabela de decisão baseia-se nos modelos de comportamento do software, que são obtidos dos requisitos do sistema (MARTINS, 2010, p.6).

“O objetivo principal dessa técnica é verificar se as possíveis combinações do sistema estão sendo manipuladas de acordo com o previsto. (...), ou seja, é uma técnica que tem por objetivo auxiliar na criação de casos de teste” (VIEIRA, 2010). Mas é necessário tomar cuidado na execução desse processo, pois cada condição presente na tabela deve se transformar em um caso de teste e “se a quantidade de regras e ações envolvidas for muito grande, teremos uma quantidade ainda maior de casos de teste, os quais se tornam inviáveis de testar” (MOLINARI, 2008, p. 157).

O fato de a técnica criar todas as combinações possíveis entre as principais variáveis do sistema torna os testes muito mais abrangentes. De acordo com Campos (2009), a principal vantagem de utilizar tabela de decisão é que, geralmente, ela cria combinações complexas e que não foram exercidas durante os testes.

A tabela de decisão pode ser aplicada a qualquer momento do processo, com um enfoque maior para os casos em que existem muitas regras de negócios e uma regra interfere na outra (MOLINARI, 2008, p. 155). Campos (2009) completa dizendo que a técnica de tabela de decisão pode ser aplicada a todas as situações, principalmente nos casos em que a execução do software depende de muitas decisões lógicas. A Figura 1 apresenta um exemplo de tabela de decisão criado por Vieira (2010) para testar o sistema de uma empresa de processamento de cartões de crédito. Vieira (2010) explica que o objetivo dessa tabela de decisão é validar os titulares e vendedores, além de definir as ações que devem ser tomadas de acordo com cada perfil de cliente.

Verifica-se que a tabela é formada por quatro variáveis, e os valores válidos são definidos como sim (S) ou não (N). A tabela possui três ações, que serão escolhidas de acordo com regras pré-definidas. Ao combinar todas as variáveis e seus valores válidos, foram geradas 16 condições de teste. Exemplo: a condição de teste 1 apresenta um caso em que o valor válido para todas as condições (variáveis) é definido como “sim” e a ação escolhida é “aprovado”.

Figura 1 - Exemplo de tabela de decisão

Condições	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Conta existe?	S	S	S	S	S	S	S	S	N	N	N	N	N	N	N	N
Conta ativa?	S	S	S	S	N	N	N	N	S	S	S	S	N	N	N	N
Possui limite?	S	S	N	N	S	S	N	N	S	S	N	N	S	S	N	N
Endereço existente?	S	N	S	N	S	N	S	N	S	N	S	N	S	N	S	N
Ações																
Aprovado?	S	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Ligar para o titular?	N	S	S	S	N	S	S	S	N	N	N	N	N	N	N	N
Ligar para o vendedor?	N	N	N	N	S	S	S	S	S	S	S	S	S	S	S	S

Fonte: Vieira (2010).

Ao analisar a Figura 1, percebe-se que algumas condições de teste são equivalentes, gerando testes repetidos. Vieira (2010) afirma que as colunas que possuem valores semelhantes e não afetam as ações tomadas podem ser mescladas, a fim de reduzir a quantidade de testes. A Figura 2 apresenta a tabela de decisão elaborada por Vieira (2010) com base na tabela de decisão da Figura 1, porém eliminando as condições equivalentes e, assim, reduzindo a tabela para apresentar apenas as condições que realmente precisam ser testadas.

Figura 2 - Exemplo de tabela de decisão reduzida

Condições	1	2	3	5	6	7	9
Conta existe?	S	S	S	S	S	S	N
Conta ativa?	S	S	S	N	N	N	-
Possui limite?	S	S	N	S	S	N	-
Endereço válido?	S	N	-	S	N	-	-
Ações							
Aprovado?	S	N	N	N	N	N	N
Ligar para o titular?	N	S	S	N	S	S	N
Ligar para o vendedor?	N	N	N	S	S	S	S

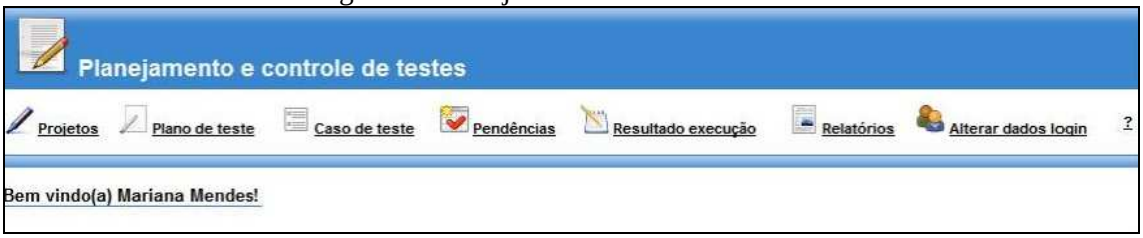
Fonte: Vieira (2010).

2.4 TRABALHOS CORRELATOS

Podem ser citados como trabalhos correlatos a dissertação realizada na Pontifícia Universidade Católica do Rio de Janeiro (PUC-RIO), pelo aluno Luiz Rodolfo Neves Caldeira (2010), para conclusão do mestrado pelo Programa de Pós-Graduação em Informática da PUC-RIO e a monografia realizada pela aluna Bruna Tatiane Bonecher (2008) para conclusão do curso de Ciências da Computação pela Universidade Regional de Blumenau (FURB). Também pode-se citar como trabalho correlato a ferramenta *online* desenvolvida por Lima (ca. 2010) para geração de tabelas de decisão.

O trabalho de Bonecher (2008) apresenta uma ferramenta web para automatizar as atividades e os artefatos do processo de teste, contemplando as etapas de planejamento e controle dos testes. O processo é baseado em outros processos existentes na literatura, como o OpenUP. Ela utiliza o padrão de documentação sugerido pela norma internacional IEEE-829. Na Figura 3 é apresentada a tela principal da ferramenta desenvolvida por Bonecher (2008), utilizada para planejamento e controle dos testes.

Figura 3 - Planejamento e controle de testes

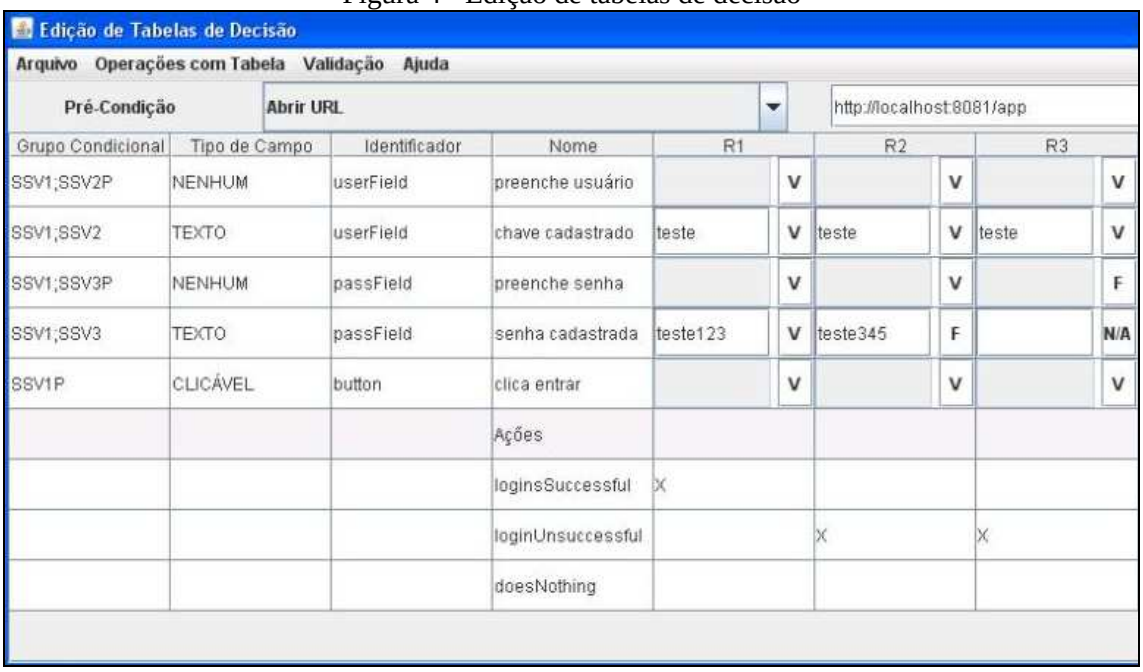


Fonte: Bonecher (2008).

Caldeira (2010) apresenta um processo e um conjunto de ferramentas para a geração semi-automática de *scripts* de teste funcional para sistemas web, a partir de casos de uso e tabelas de decisão, reduzindo o tempo gasto para gerar testes automatizados. A partir dos casos de uso e com o auxílio de uma ferramenta, monta-se manualmente uma tabela de decisão. Os casos de testes semânticos são gerados a partir destas tabelas de decisão. Outra

ferramenta é responsável por gerar os *scripts* de testes a partir dos casos de testes semânticos. Na Figura 4 é apresentada a tela de edição de tabelas de decisão, onde o usuário pode cadastrar ou editar as condições, ações e regras.

Figura 4 - Edição de tabelas de decisão



Fonte: Caldeira (2010).

A ferramenta desenvolvida por Lima (2010) gera a tabela de decisão utilizando as variáveis e regras informadas pelo usuário.A Figura 5 apresenta a tabela de decisão gerada, utilizando três variáveis.

Figura 5 - Tabela de decisão gerada pela ferramenta x|decision

Entradas/Saídas	Classes	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17	T18
Idade	18	Sim	Sim	Sim	Sim	Sim	Sim	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não
	24 a 29	Não	Não	Não	Não	Não	Não	Sim	Sim	Sim	Sim	Sim	Sim	Não	Não	Não	Não	Não	Não
	30 ou mais	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não	Sim	Sim	Sim	Sim	Sim	Sim
Sexo	F	Sim	Sim	Sim	Não	Não	Não	Sim	Sim	Sim	Não	Não	Não	Sim	Sim	Sim	Não	Não	Não
	M	Não	Não	Não	Sim	Sim	Sim	Não	Não	Não	Sim	Sim	Sim	Não	Não	Não	Sim	Sim	Sim
Tempo de carteira	menos de 1 ano	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não
	1 a 4 anos	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não
	5 anos ou mais	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim	Não	Não	Sim
Desconto	Sem desconto	-	-	-	X	X	X	-	-	-	X	-	-	-	-	-	-	-	-
	R$100,00	X	X	X	-	-	-	X	-	-	-	X	-	-	-	-	X	-	-
	R$300,00	-	-	-	-	-	-	-	X	X	-	-	X	X	-	-	-	X	-
	R$500,00	-	-	-	-	-	-	-	-	-	-	-	-	-	X	X	-	-	X

Fonte: Lima (2010).

3 DESENVOLVIMENTO DA FERRAMENTA

Neste capítulo são descritas as particularidades técnicas e detalhamento da ferramenta, apresentando as suas características, os requisitos funcionais e os não funcionais, o diagrama de caso de uso e suas descrições, o diagrama de entidade de relacionamento, implementação e resultados e discussões.

3.1 LEVANTAMENTO DE INFORMAÇÕES

A ferramenta desenvolvida neste trabalho tem como principal objetivo gerar tabelas de decisão, de modo que as informações contidas na tabela gerada auxiliem o analista de teste na criação de casos de teste mais abrangentes. Inicialmente, a tabela apresenta todas as condições de teste possíveis, porém, para auxiliar o usuário, a ferramenta identifica quais condições de testes são consideradas repetitivas e permite que o usuário as elimine da tabela. Além disso, a ferramenta permite que o usuário automatize a criação dos casos de teste a partir da tabela de decisão, utilizando um formato padrão pré-definido pelo usuário, ou exporte a tabela de decisão para posterior acompanhamento.

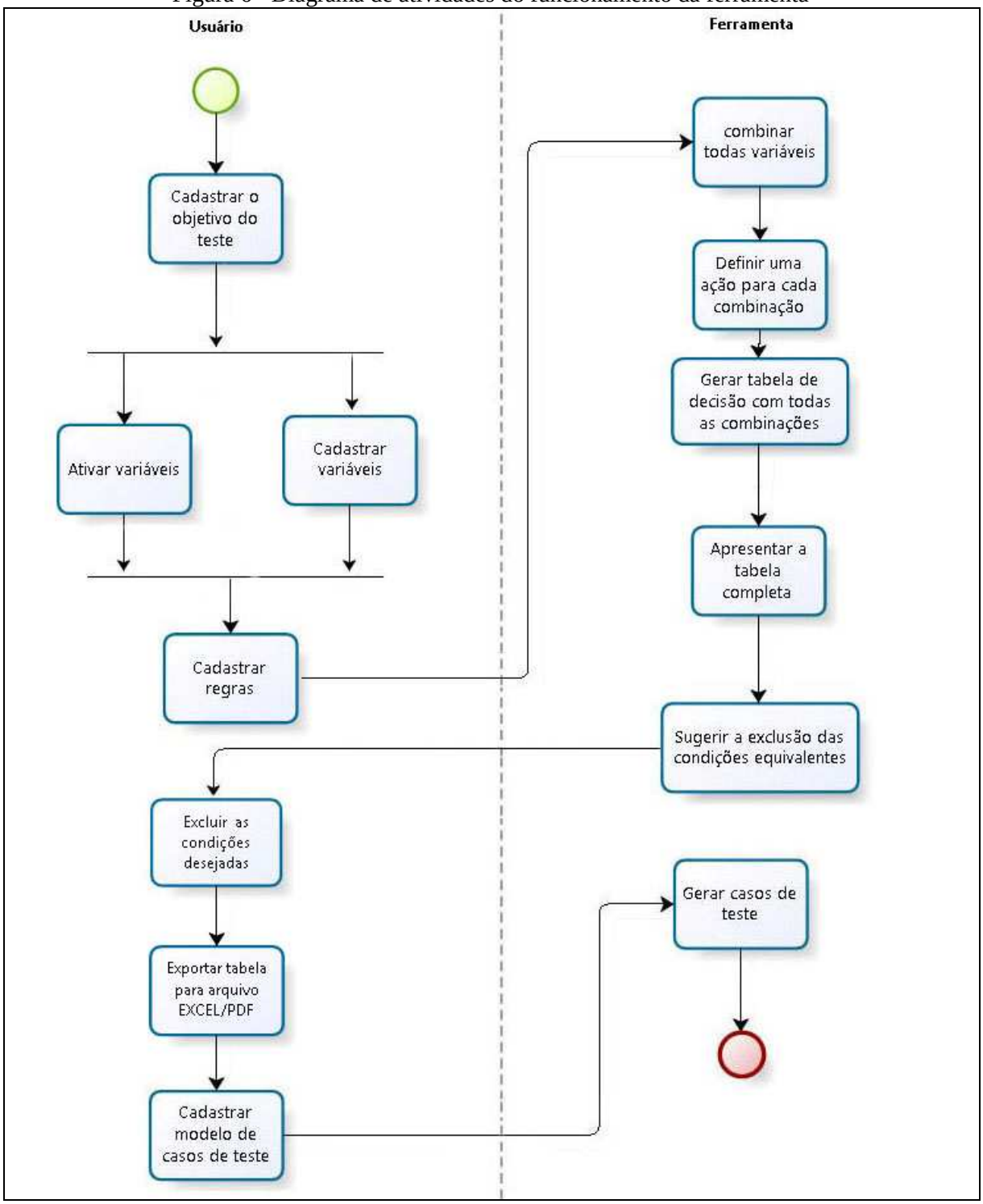
Para gerar a tabela de decisão, o usuário deve cadastrar as variáveis que serão utilizadas no teste. Essas variáveis devem ser escolhidas pelo usuário conforme o impacto delas no teste do software. Para cada variável, o usuário deve cadastrar o nome, o tipo e um ou mais valores de teste. O usuário também poderá ativar ou desativar variáveis utilizadas em testes anteriores, pois todas as variáveis utilizadas são armazenadas no banco de dados e apresentadas em uma tela para consulta. Após o cadastro das variáveis de teste, a ferramenta apresenta a tela de cadastro de regra. Por meio dessa tela, o usuário define qual ação deve ser apresentada de acordo com combinações entre os valores de teste das variáveis.

Um dos maiores problemas na criação manual de tabelas de decisão é o tempo que o analista de testes gasta para preencher a tabela, pois é necessário combinar todos os valores de teste de cada variável. Para aprimorar este processo, a ferramenta cria as combinações e apresenta qual ação deverá ser tomada em cada condição de teste. A ferramenta também identifica quais testes são equivalentes e podem ser excluídos, além de permitir que o usuário exclua outras condições individualmente, caso ele considere necessário.

A ferramenta considera como testes equivalentes aqueles que possuem ações iguais utilizando variáveis semelhantes. Para evitar testes duplicados e, ao mesmo tempo, garantir a execução de todos os testes necessários, a ferramenta apresenta ao usuário a possibilidade de reduzir a tabela de decisão ao nível recomendado.

O nível recomendado reduz a tabela de decisão de modo que todos os valores de teste de todas as variáveis sejam apresentados ao menos uma vez. Valores de teste pertencentes à regra, que possuem alto impacto no teste, são apresentados mais de uma vez, enquanto valores de baixo impacto, que não pertencem a regra, mas que fazem parte do teste que está sendo criado, são apresentados apenas uma vez.

Figura 6 - Diagrama de atividades do funcionamento da ferramenta



Um valor de teste de baixo impacto se repetirá apenas quando a variável que ele pertencer possuir uma quantidade de valores de teste menor do que a quantidade cadastrada para alguma outra variável. Neste caso, os valores das variáveis com menor quantidade se repetirão apenas para completar a tabela, combinando com os valores pertencentes às variáveis com maior quantidade. Além disto, a ferramenta permite que o usuário reduza a tabela ao nível mínimo, ou seja, menor quantidade de condições de teste possíveis. Nesta funcionalidade, a ferramenta reduzirá a tabela de modo que exista apenas uma condição de teste para cada regra cadastrada. Esta funcionalidade foi desenvolvida para tratar as situações em que o teste deve ser executado, porém possui um impacto muito baixo, sendo suficiente testar uma vez cada regra cadastrada. Também pode ser usado nos casos em que o tempo disponível para executar o teste de determinada funcionalidade é muito pequeno, não sendo possível executar mais de um teste para cada regra.

Para facilitar a geração dos casos de testes, a ferramenta permite que o usuário transforme a tabela de decisão em casos de teste, utilizando um formato padrão definido por ele. Nesse processo, cada condição de teste da tabela transforma-se em um caso de teste. Por fim, o usuário poderá exportar a tabela para um arquivo xlsx ou pdf. A Figura 6 apresenta o funcionamento da ferramenta em alto nível.

3.2 ESPECIFICAÇÃO

Esta seção apresenta os requisitos funcionais e não funcionais da ferramenta, sua rastreabilidade com os casos de uso, o modelo de casos de uso e o modelo de entidade relacionamento (MER).

3.2.1 Requisitos funcionais

O Quadro 1 apresenta os requisitos funcionais previstos para a ferramenta, bem como sua rastreabilidade, ou seja, vinculação com os casos de uso associados.

Quadro 1 – Requisitos Funcionais

Requisitos Funcionais	Caso de Uso
RF01: O sistema deverá permitir ao usuário o cadastramento do objetivo do teste.	UC01
RF02: O sistema deverá permitir ao usuário o cadastramento de variáveis.	UC02
RF03: O sistema deverá permitir ao usuário o cadastramento de valores de teste para as variáveis.	UC02
RF04: O sistema deverá armazenar as variáveis e valores de teste no banco de dados e apresentar ao usuário todas as variáveis e valores de teste armazenados.	UC02
RF05: O sistema deverá permitir ao usuário ativar e desativar as variáveis cadastradas para utilizar em seus testes.	UC02
RF06: O sistema deverá permitir ao usuário excluir variáveis.	UC02
RF07: O sistema deverá permitir ao usuário o cadastramento de regras para definir qual ação deverá ser tomada em cada condição de teste.	UC03
RF08: O sistema deverá armazenar as regras no banco de dados e apresentar ao usuário todas as regras armazenadas.	UC03
RF09: O sistema deverá permitir ao usuário excluir as regras que não serão utilizadas no teste.	UC03
RF10: O sistema deverá combinar todos os valores de teste de todas as variáveis para gerar a tabela de decisão.	UC04
RF11: O sistema deverá determinar qual ação será tomada em cada condição de teste da tabela, com base nas regras cadastradas.	UC04
RF12: O sistema deverá gerar a tabela de decisão contemplando todas as condições de teste possíveis.	UC04
RF13: O sistema deverá apresentar a tabela de decisão com todas as condições de teste e respectivas ações.	UC04
RF14: O sistema deverá identificar as condições de teste equivalentes.	UC05
RF15: O sistema deverá permitir ao usuário reduzir a tabela de decisão ao tamanho recomendado, desconsiderando as condições de teste equivalentes.	UC05
RF16: O sistema deverá permitir ao usuário reduzir a tabela de decisão ao tamanho mínimo, apresentando apenas uma condição de teste por regra.	UC05
RF17: O sistema deverá permitir ao usuário excluir qualquer condição de teste da tabela.	UC05
RF18: O sistema deverá permitir ao usuário cadastrar um modelo padrão (*template*) para gerar casos de teste.	UC06
RF19: O sistema deverá transformar cada condição de teste da tabela de decisão em um caso de teste, utilizando o *template* definido pelo usuário.	UC06
RF20: O sistema deverá permitir que o usuário exporte a tabela para um arquivo Excel ou PDF.	UC07

3.2.2 Requisitos não funcionais

O Quadro 2 apresenta os requisitos não funcionais previstos para o sistema.

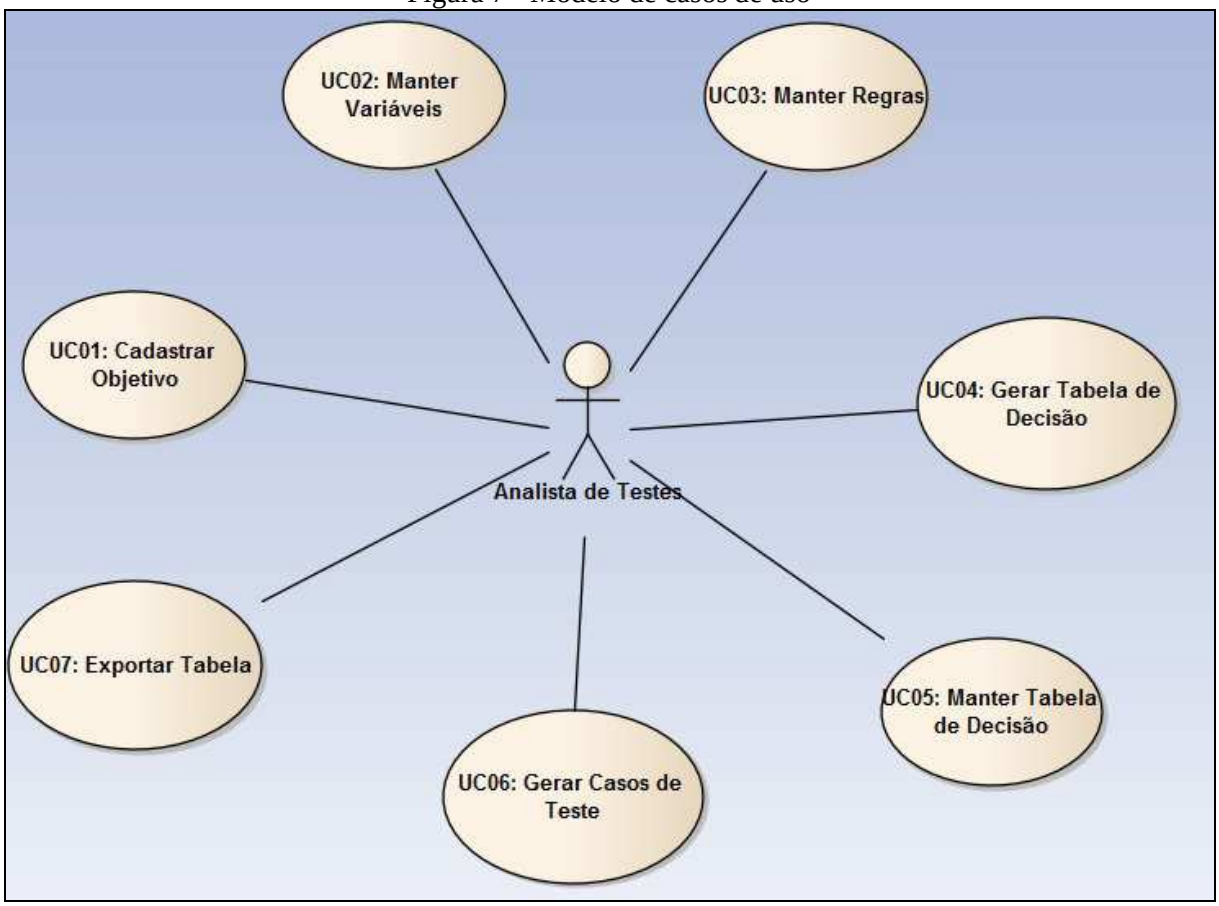
Quadro 2 - Requisitos não funcionais

Requisitos Não Funcionais
RNF01: O sistema web deverá utilizar o banco de dados MySQL.
RNF02: O sistema web deverá ser implementado em JavaScript e JSP.
RNF03: O sistema web deverá suportar as versões mais atualizadas dos navegadores Google Chrome e Mozilla Firefox.

3.2.3 Modelo de casos de uso

A Figura 7 apresenta o modelo de casos de uso do sistema, composto por sete casos de uso e um ator. Para melhor entendimento do projeto, o detalhamento dos casos de uso encontra-se no Apêndice A.

Figura 7 - Modelo de casos de uso

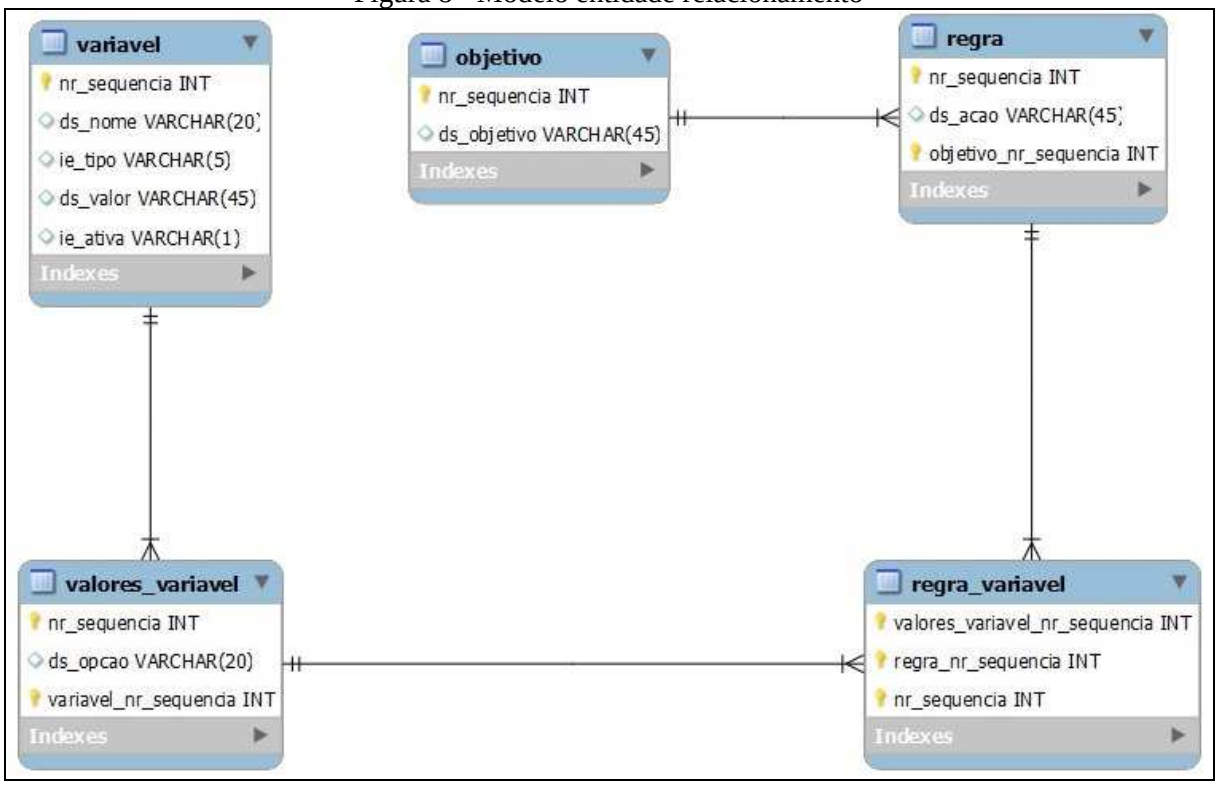


3.2.4 Modelo entidade relacionamento

A Figura 8 apresenta o modelo entidade relacionamento (MER) da ferramenta desenvolvida. O dicionário de dados pode ser visualizado no Apêndice B. A seguir, é apresentada uma breve descrição das entidades criadas para o desenvolvimento do sistema:

a) `variavel`: entidade que armazena as variáveis cadastradas;
b) `valores_variavel`: entidade que armazena os valores cadastrados para cada variável;
c) `regra_variavel`: entidade que armazena as variáveis vinculadas a cada regra;
d) `regra`: entidade que armazena as regras que serão consideradas para definir as ações na tabela de condição;
e) `objetivo`: entidade que armazena o objetivo do teste.

Figura 8 - Modelo entidade relacionamento



3.3 IMPLEMENTAÇÃO

A seguir são apresentados os softwares e as técnicas utilizadas no desenvolvimento do sistema, além da operacionalidade da implementação.

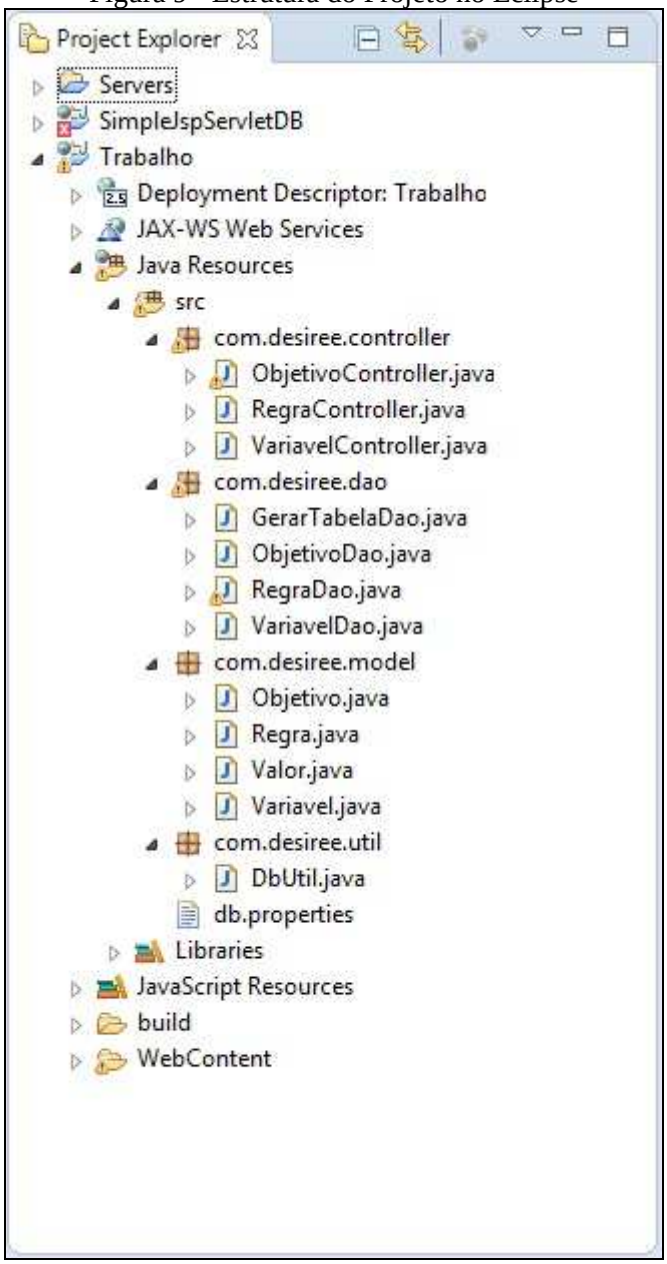
3.3.1 Técnicas e softwares utilizados

Os softwares utilizados no desenvolvimento deste trabalho são livres e disponíveis no mercado. A ferramenta foi desenvolvida utilizando as linguagens Java, JavaScript e JSP, sendo que a IDE utilizada para o desenvolvimento foi o Eclipse Mars. Os dados cadastrados no sistema são armazenados em uma base de dados do banco MySQL Workbench.

3.3.1.1 Eclipse Mars

O Eclipse é uma das IDEs mais completas e comuns para desenvolver softwares na linguagem Java. A versão Mars é a versão mais recente da IDE e possui várias novas funcionalidades, como, por exemplo, visões hierárquicas para projetos aninhados, melhorias nas buscas de texto e possibilidade de customização de perspectivas. A Figura 9 apresenta a estrutura de arquivos criada no Eclipse Mars para o desenvolvimento do sistema.

Figura 9 - Estrutura do Projeto no Eclipse



3.3.1.2 MySQL

O MySQL é um sistema gerenciador de banco de dados relacional. O Quadro 3 apresenta a conexão com o banco de dados MySQL realizada por meio de uma classe Java.

Quadro 3 – Conexão entre o banco de dados e a aplicação

Conexão – MySQL

```
package com.desiree.util;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;
```

```
public class DbUtil {
   private static Connection connection = null;

   public static Connection getConnection() {
      if (connection != null)
         return connection;
      else {
         try {
            Properties prop = new Properties();
            InputStream inputStream =
DbUtil.class.getClassLoader().getResourceAsStream("/db.properties");
            prop.load(inputStream);
            String driver = prop.getProperty("driver");
            String url = prop.getProperty("url");
            String user = prop.getProperty("user");
            String password = prop.getProperty("password");
            Class.forName(driver);
            connection =
               DriverManager.getConnection(url, user, password);
         } catch (ClassNotFoundException e) {
            e.printStackTrace();
         } catch (SQLException e) {
            e.printStackTrace();
         } catch (FileNotFoundException e) {
            e.printStackTrace();
         } catch (IOException e) {
            e.printStackTrace();
         }
         return connection;
      }
   }
}
```

Durante a implementação da ferramenta, foram desenvolvidas classes para acesso ao banco de dados. O Quadro 4 apresenta a implementação do método de inserção de variáveis, localizado na classe `VariavelDao`, que tem a função de inserir no banco de dados as novas variáveis.

Quadro 4 - Método para inserir variáveis no banco de dados

Método para inserir variáveis

```
public void insert(Variavel variavel) {
   try {
      PreparedStatement preparedStatement = connection
         .prepareStatement("insert into variavel
          (ds_nome,ie_tipo,ds_valor, ie_ativa) values (?,?,?,?)");
      if (variavel.getNome() != null) {
         preparedStatement.setString(1, variavel.getNome());
      } else {
         preparedStatement.setNull(1, java.sql.Types.VARCHAR);
      }
      if (variavel.getIeTipo() != null) {
         preparedStatement.setString(2, variavel.getIeTipo());
      } else {
         preparedStatement.setNull(2, java.sql.Types.VARCHAR);
      }
      if (variavel.getValores() != null) {
         preparedStatement.setString(3, variavel.getValores());
      } else {
         preparedStatement.setNull(3, java.sql.Types.VARCHAR);
      }
      if (variavel.getIeAtiva() != null) {
         preparedStatement.setString(4, variavel.getIeAtiva());
      } else {
         preparedStatement.setNull(4, java.sql.Types.VARCHAR);
      }
      preparedStatement.execute();
```

```
      insertValor(retornaUltimaVariavel(), variavel.getValores());
      preparedStatement.close();

   } catch (SQLException e) {
      e.printStackTrace();
   }
}
```

A seguir, é apresentado o método responsável por gerar a tabela de decisão, utilizando o conteúdo armazenado no banco de dados. Este método está localizado na classe `GerarTabelaDao`. O Quadro 5 apresenta o início da implementação do método. Neste trecho do código, foram criadas as variáveis que serão utilizadas para a geração da tabela de decisão e executados os comandos para selecionar do banco de dados a quantidade de casos de teste e tipos que a tabela terá.

Quadro 5 - Método para gerar tabela de decisão – Parte 1

Método para gerar tabela de decisão– Parte 1

```
public ArrayList<Object> gerar() {
   ArrayList<Object> retorno = new ArrayList<>();
   String[] columnNames;
   String[] variaveis;

   String qtCasosSQL = "  select  count(*) qt_variaveis\n"
         + "from    variavel a,\n"
         + "           valores_variavel b\n"
         + "where b.nr_variavel = a.nr_sequencia and a.ie_ativa ='S'
\n"
         + "group by ds_nome ";

   int qtCasos = 1;
   int qtTipos = 0;

   try (PreparedStatement preparedStatement =
      connection.prepareStatement(qtCasosSQL)) {
      ResultSet rs = preparedStatement.executeQuery();
      while (rs.next()) {
         if (rs.getInt("QT_VARIAVEIS") != 0) {
            qtTipos++;
            qtCasos *= rs.getInt("QT_VARIAVEIS");
         }
         }
         preparedStatement.close();
   } catch (Exception e) {
      System.err.println(e.getLocalizedMessage());
   }
   columnNames = new String[qtCasos + 1];
   columnNames[0] = "-";
   for (int i = 1; i <= qtCasos; i++) {
      columnNames[i] = "C" + (i);
   }
```

No Quadro 6 é apresentado o trecho do código em que são selecionadas todas as variáveis ativas, ordenadas pela sequência, além dos valores de cada variável.

Quadro 6 - Método para gerar tabela de decisão - Parte 2

Método para gerar tabela de decisão – Parte 2

```
variaveis = new String[qtTipos];

   String variaveisSQL =
      " select ds_nome from variavel where ie_ativa = 'S' order by
nr_sequencia ";
```

```
      HashMap<String, ArrayList<Object>> valoresVariavel = new
HashMap<>();

      try (PreparedStatement preparedStatement =
         connection.prepareStatement(variaveisSQL)) {
         ResultSet rs = preparedStatement.executeQuery();
         int aux = 0;
         while (rs.next()) {
            variaveis[aux++] = rs.getString("DS_NOME");
         }
      preparedStatement.close();
      } catch (Exception e) {
         System.err.println(e.getLocalizedMessage());
      }

      for (String nomeVariavel : variaveis) {
         String selectOpcoes = " select  b.ds_opcao\n"
         + " from    variavel a,\n"
         + " valores_variavel b\n"
         + " where   b.nr_variavel = a.nr_sequencia\n"
         + " and     a.ds_nome = ?";
         try (PreparedStatement preparedStatement =
            connection.prepareStatement(selectOpcoes)) {
            if (nomeVariavel != null) {
               preparedStatement.setString(1, nomeVariavel);
            } else {
               preparedStatement.setNull(1, java.sql.Types.VARCHAR);
            }
            ResultSet rs = preparedStatement.executeQuery();
            ArrayList<Object> opcoes;
            if (valoresVariavel.containsKey(nomeVariavel)) {
               opcoes = valoresVariavel.get(nomeVariavel);
            } else {
               opcoes = new ArrayList<>();
            }
            while (rs.next()) {
               opcoes.add(rs.getString("DS_OPCAO"));
            }
            valoresVariavel.put(nomeVariavel, opcoes);
         } catch (Exception e) {
            System.err.println(e.getLocalizedMessage());
         }
      }
```

O Quadro 7 demonstra o trecho do código em que se inicia a montagem da tabela. Para a primeira variável inserida na tabela, a ferramenta divide a quantidade de casos de teste, calculado na parte 1 do método, pela quantidade de valores que a variável possui. O resultado desse cálculo será o número de vezes que cada valor da primeira variável se repetirá. Para as variáveis seguintes, a ferramenta divide a quantidade de repetições da variável anterior pela quantidade de valores que a variável atual possui, sem levar em consideração a quantidade de casos que a tabela possui.

Quadro 7 - Método para gerar tabela de decisão - Parte 3

Método para gerar tabela de decisão – Parte 3

```
Object[][] linhas = new Object[qtTipos + 2][qtCasos + 1];
int qtRepeticao = 0;

boolean iePrimeira = true;
for (int i = 0; i < variaveis.length; i++) {
   linhas[i][0] = variaveis[i];
   ArrayList<Object> opcoes = valoresVariavel.get(variaveis[i]);
   if (iePrimeira) {
```

```
      qtRepeticao = qtCasos / opcoes.size();
      iePrimeira = false;
      for (Object opcao : opcoes) {
         int qtAdd = 0;
         for (int j = 1; j <= qtCasos; j++) {
            if (qtAdd < qtRepeticao && linhas[i][j] == null) {
               qtAdd++;
               linhas[i][j] = opcao;
            }
         }
      }
} else {
   qtRepeticao /= opcoes.size();
      int qtFors = qtCasos / opcoes.size();
      for (int x = 0; x < qtFors; x++) {
         for (Object opcao : opcoes) {
            int qtAdd = 0;
            for (int j = 1; j <= qtCasos; j++) {
               if (qtAdd < qtRepeticao && linhas[i][j] == null) {
                  qtAdd++;
                  linhas[i][j] = opcao;
               }
            }
         }
      }
   }

}
```

No Quadro 8 é possível visualizar a parte final do método, em que as duas últimas linhas são inseridas na tabela. A penúltima linha contém a ação de cada condição de teste. A última linha apresenta o botão excluir, acompanhado do número da condição de teste, para possibilitar ao usuário a exclusão manual de condições de teste.

Quadro 8 - Método para gerar tabela de decisão - Parte 4

Método para gerar tabela de decisão – Parte 4

```
linhas[linhas.length - 2][0] = "Acao";
   linhas[linhas.length - 1][0] = "-";
   Object[] valoresCaso = new String[linhas.length - 1];
   for (int i = 1; i < linhas[0].length; i++) {
      for (int j = 0; j < valoresCaso.length; j++) {
         valoresCaso[j] = linhas[j][i];
      }
      linhas[linhas.length - 2][i] = verificaRegra(valoresCaso);
      linhas[linhas.length - 1][i] = "excluir" + i;
   }

   retorno.add(columnNames);
   retorno.add(linhas);
   return retorno;
}
```

3.3.2 Operacionalidade da implementação

Esta subseção demonstra o funcionamento da implementação por meio da utilização do sistema em nível de usuário. A ferramenta não possui controle de usuários, portanto, ao abri-la, é apresentada a tela principal para iniciar a especificação de testes, conforme Figura 10. Ao iniciar a especificação de testes, o usuário deve informar o objetivo do teste que será criado. Este objetivo será armazenado no banco de dados e apresentado junto com a tabela de

decisão que será gerada posteriormente. A Figura 11 apresenta a tela de cadastro de objetivo. No exemplo apresentado, o objetivo do teste é validar o acesso ao sistema da universidade.

Figura 10 - Tela inicial

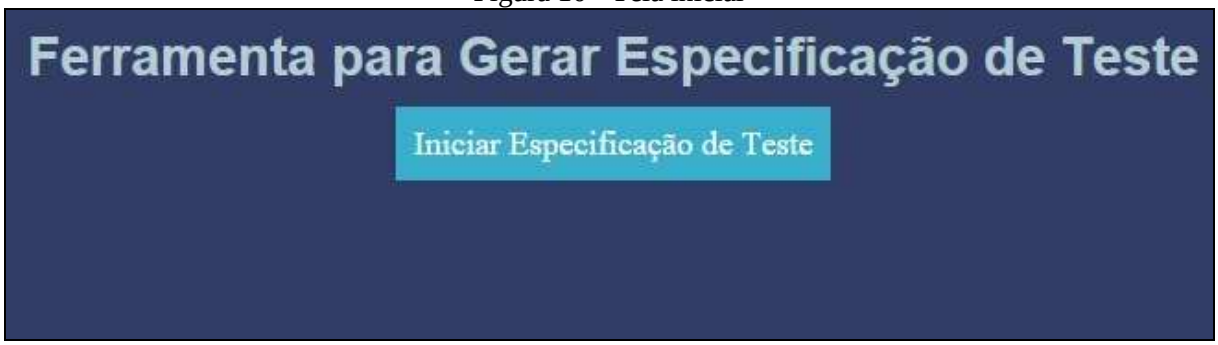
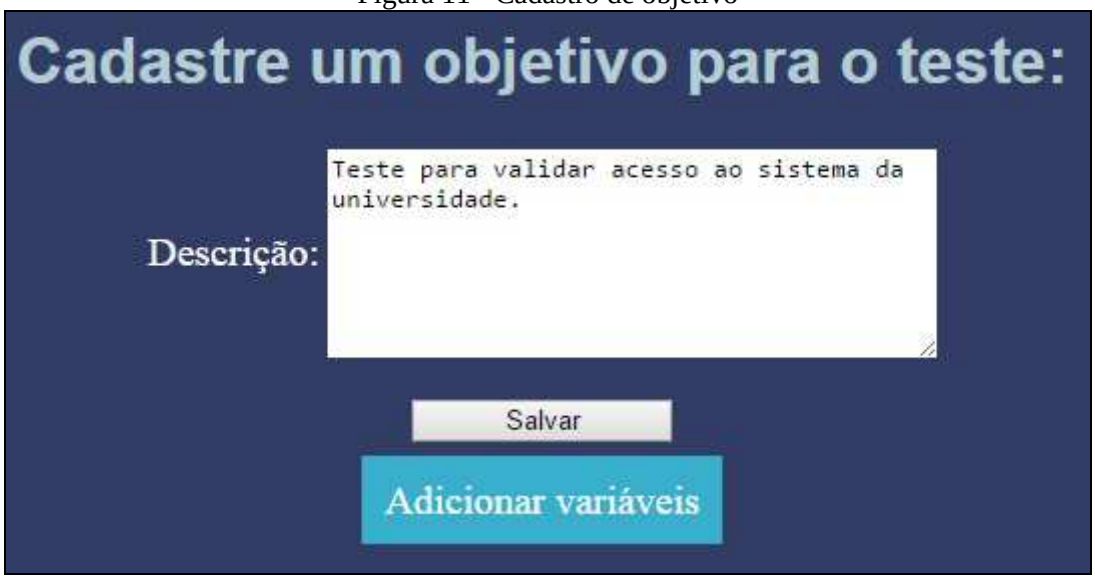


Figura 11 - Cadastro de objetivo



Com o objetivo do teste cadastrado, o usuário poderá navegar para a tela de inserção e manutenção de variáveis. Por meio desta tela, ele poderá cadastrar novas variáveis, alterar, excluir, ativar ou desativar variáveis existentes, conforme apresentado na Figura 12. Essa tela também informa ao usuário a quantidade de condições de teste que serão criadas caso ele gere a tabela de decisão com todas as variáveis que estão ativas. Esta funcionalidade foi implementada a fim de facilitar ao usuário a decisão de quantas variáveis e valores utilizar em seu teste. No exemplo, existem quatro variáveis cadastradas, sendo que apenas três delas estão ativas e serão utilizadas ao gerar a tabela.

Após ter escolhido as variáveis que serão utilizadas no teste, o usuário poderá acessar a tela de cadastro e manutenção de regras, conforme Figura 13. Nessa tela, o usuário deverá selecionar uma ou mais variáveis, acompanhadas de um valor válido correspondente, para compor uma regra. Para cada combinação de variáveis e valores, ele deverá informar uma ação. Essas combinações serão levadas em consideração no momento de inserir as ações na tabela de decisão.

Figura 12 - Cadastro de variáveis

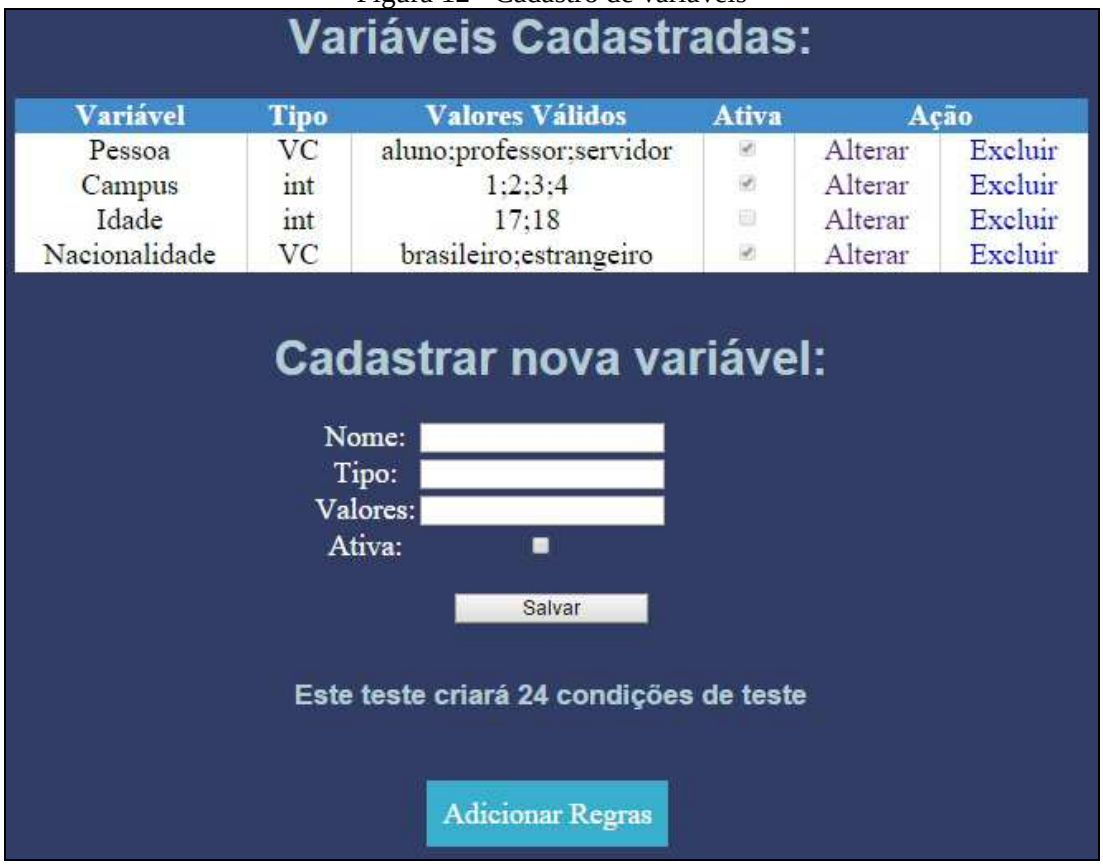
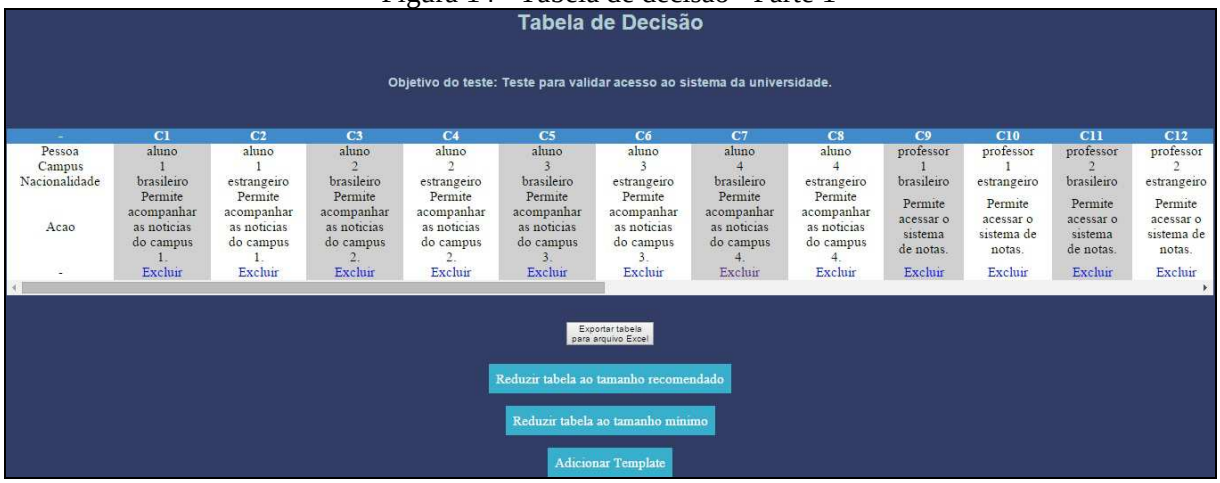


Figura 13 - Cadastro de regra

Para gerar a tabela de decisão, a ferramenta levará em conta todas as regras cadastradas no banco de dados. Com o intuito de evitar que o usuário cadastre regras repetidas e facilitar a visualização das regras cadastradas, essa tela também apresenta uma tabela com as regras existentes no banco de dados. Caso o usuário não queira utilizar alguma destas regras, poderá excluí-las individualmente. No exemplo apresentado na Figura 13, existem seis regras cadastradas.

A partir da tela de manutenção de regras, o usuário poderá gerar a tabela de decisão. A tabela será gerada com todas as condições de testes possíveis. A ferramenta permite que o usuário exclua condições de teste individualmente, conforme necessidade. A Figura 14 apresenta a primeira parte da tabela de decisão gerada e a Figura 15 apresenta a continuação da tabela.

Figura 14 - Tabela de decisão - Parte 1



-	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
Pessoa	aluno	aluno	aluno	aluno	aluno	aluno	aluno	aluno	professor	professor	professor	professor
Campus	1	1	2	2	3	3	4	4	1	1	2	2
Nacionalidade	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro
Acao	Permite acompanhar as noticias do campus 1.	Permite acompanhar as noticias do campus 1.	Permite acompanhar as noticias do campus 2.	Permite acompanhar as noticias do campus 2.	Permite acompanhar as noticias do campus 3.	Permite acompanhar as noticias do campus 3.	Permite acompanhar as noticias do campus 4.	Permite acompanhar as noticias do campus 4.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.
-	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir

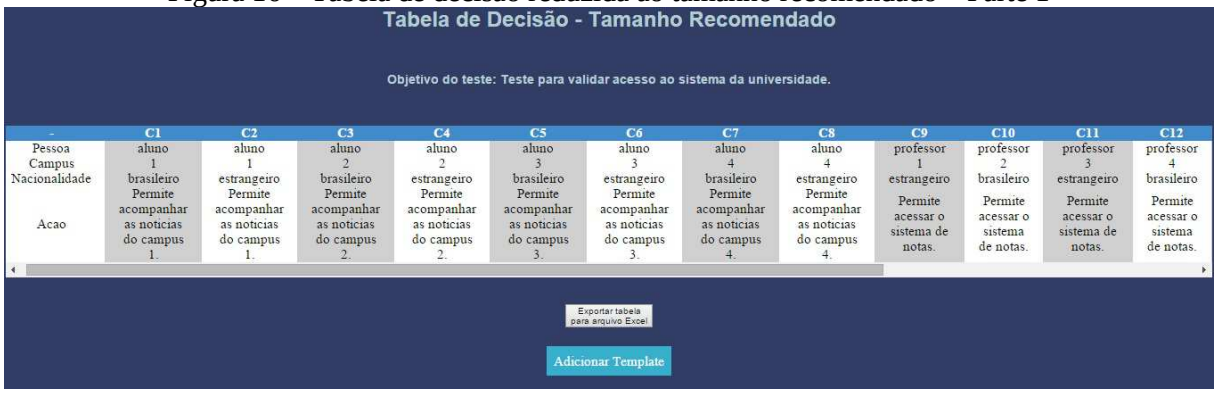
Figura 15 - Tabela de decisão - Parte 2

C13	C14	C15	C16	C17	C18	C19	C20	C21	C22	C23	C24
professor	professor	professor	professor	servidor	servidor	servidor	servidor	servidor	servidor	servidor	servidor
3	3	4	4	1	1	2	2	3	3	4	4
brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro
Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.
Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir	Excluir

Após gerar a tabela de decisão, a ferramenta identifica quais testes são equivalentes e permite que o usuário exclua todos estes testes de uma só vez, reduzindo a tabela ao tamanho recomendado ou ao tamanho mínimo, conforme explicado na seção 3.1. As Figuras 16 e 17 apresentam a tabela de decisão gerada anteriormente, porém reduzida ao tamanho

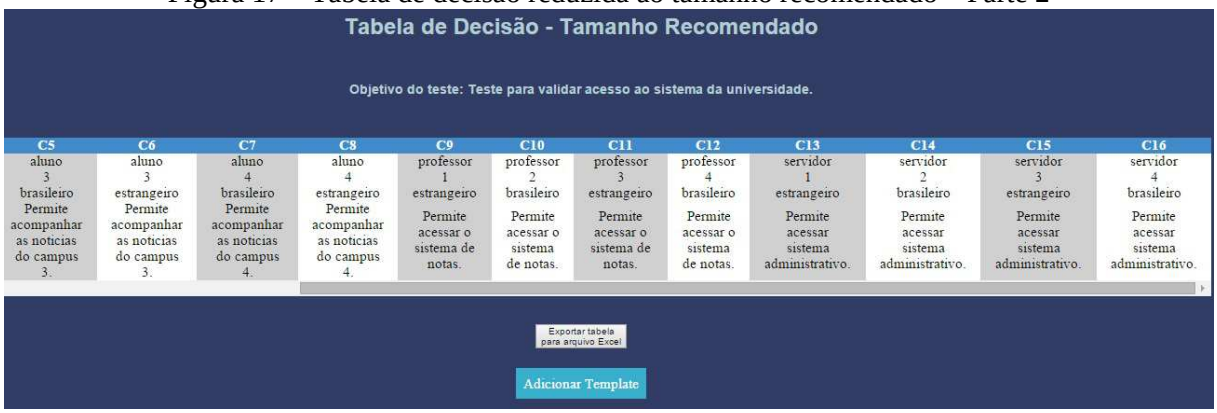
recomendado, contendo dezoito condições de teste. A Figura 18 apresenta a tabela de decisão reduzida ao tamanho mínimo, contendo seis condições de teste.

Figura 16 – Tabela de decisão reduzida ao tamanho recomendado – Parte 1



-	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
Pessoa	aluno 1	aluno 1	aluno 2	aluno 2	aluno 3	aluno 3	aluno 4	aluno 4	professor 1	professor 2	professor 3	professor 4
Campus	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	estrangeiro	brasileiro	estrangeiro	brasileiro
Nacionalidade	Permite	Permite	Permite	Permite	Permite	Permite	Permite	Permite				
Acao	acompanhar as noticias do campus 1.	acompanhar as noticias do campus 1.	acompanhar as noticias do campus 2.	acompanhar as noticias do campus 2.	acompanhar as noticias do campus 3.	acompanhar as noticias do campus 3.	acompanhar as noticias do campus 4.	acompanhar as noticias do campus 4.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.

Figura 17 – Tabela de decisão reduzida ao tamanho recomendado – Parte 2



C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16
aluno 3	aluno 3	aluno 4	aluno 4	professor 1	professor 2	professor 3	professor 4	servidor 1	servidor 2	servidor 3	servidor 4
brasileiro	estrangeiro	brasileiro	estrangeiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro
Permite acompanhar as noticias do campus 3.	Permite acompanhar as noticias do campus 3.	Permite acompanhar as noticias do campus 4.	Permite acompanhar as noticias do campus 4.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar o sistema de notas.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.	Permite acessar sistema administrativo.

Figura 18 – Tabela de decisão reduzida ao tamanho mínimo

-	C1	C2	C3	C4	C5	C6
Pessoa	aluno 1	aluno 2	aluno 3	aluno 4	professor 1	servidor 1
Campus	brasileiro	brasileiro	brasileiro	brasileiro	estrangeiro	estrangeiro
Nacionalidade	Permite	Permite	Permite	Permite		
Acao	acompanhar as noticias do campus 1.	acompanhar as noticias do campus 2.	acompanhar as noticias do campus 3.	acompanhar as noticias do campus 4.	Permite acessar o sistema de notas.	Permite acessar sistema administrativo.

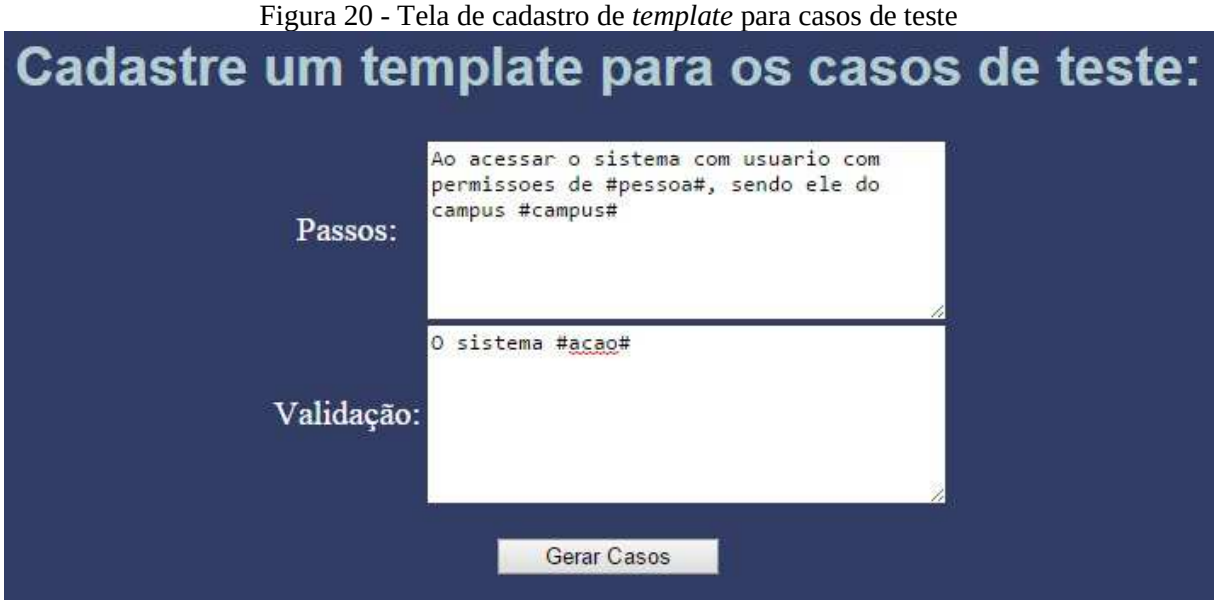
Na tela da tabela de decisão, o usuário ainda poderá exportar a tabela gerada para arquivos no formato Excel ou PDF. A Figura 19 apresenta trecho da tabela de decisão exportada para um arquivo no formato xlsx.

Figura 19 – Trecho de tabela de decisão exportada para arquivo xlsx

	A	B	C	D	E	F
1	-	C1	C2	C3	C4	C5
2	Pessoa	aluno	aluno	aluno	aluno	aluno
3	Campus	1	1	2	2	
4	Nacionalidade	brasileiro	estrangeiro	brasileiro	estrangeiro	brasileiro
5	Acao	Permite acompanhar as noticias do campus 1.	Permite acompanhar as noticias do campus 1.	Permite acompanhar as noticias do campus 2.	Permite acompanhar as noticias do campus 2.	Permite acompanhar as noticias do ca 3.
6	-					
7						
8						

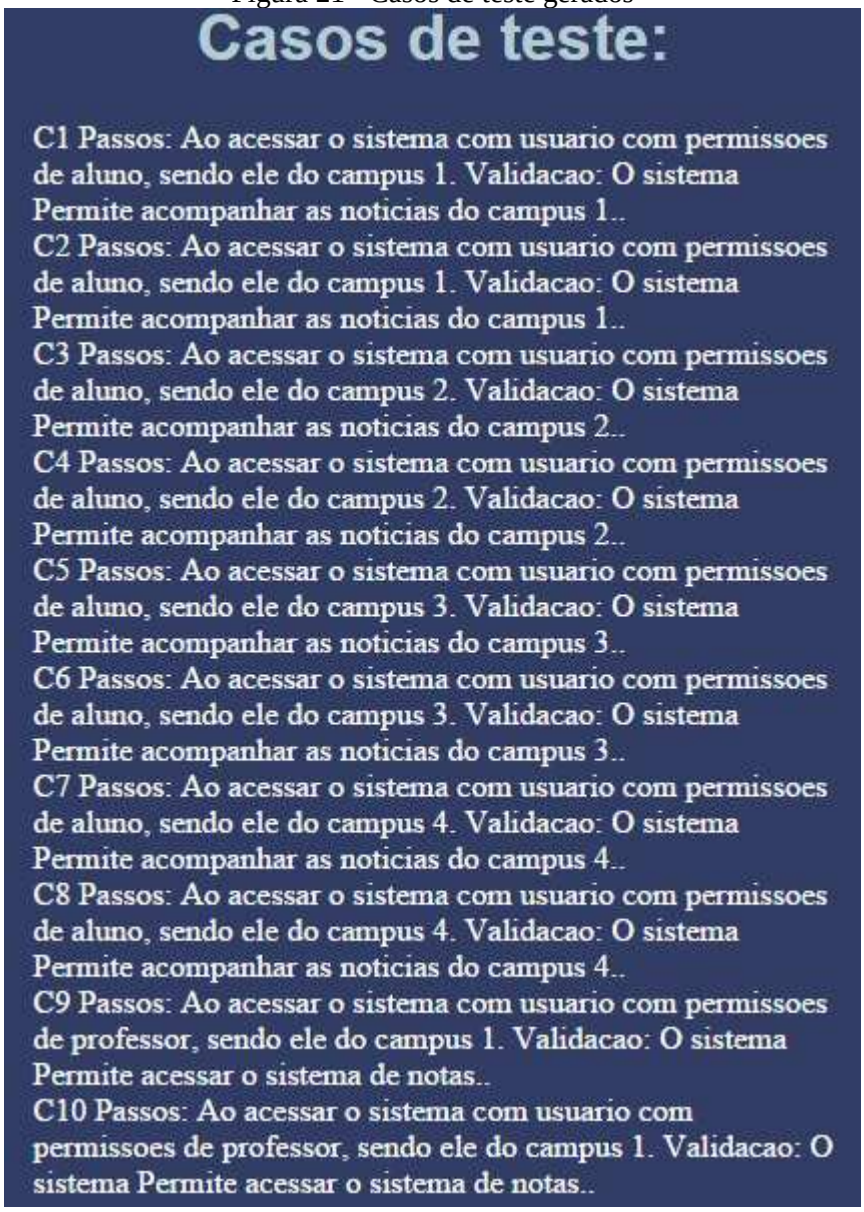
A ferramenta foi construída com o objetivo de gerar as tabelas de decisão. Porém, para auxiliar o usuário na criação dos casos de teste, foi desenvolvida uma funcionalidade para gerar os casos de teste utilizando os dados da tabela de decisão e um *template* informado pelo usuário. O texto do *template* será utilizado para transformar cada coluna da tabela de decisão em um caso de teste. Para apresentar o valor de teste da variável nos casos de teste, o usuário deverá informar o nome da variável entre dois sustenidos. A Figura 20 mostra um exemplo de *template* cadastrado para gerar os casos de teste.

Figura 20 - Tela de cadastro de *template* para casos de teste



A Figura 21 apresenta os casos de teste gerados a partir da tabela de decisão, utilizando o *template* da figura anterior.

Figura 21 - Casos de teste gerados



3.4 RESULTADOS E DISCUSSÕES

Todos os objetivos apresentados neste trabalho foram atendidos. A ferramenta desenvolvida gera tabelas de decisão a partir de variáveis e regras definidas pelo usuário, identifica as condições equivalentes dentro da tabela gerada e permite que o usuário exclua estas condições de uma só vez. Para facilitar o armazenamento das tabelas de decisão e a posterior consulta, a ferramenta permite que a tabela de decisão seja exportada em arquivos nos formatos pdf e xlsx. Além disto, a ferramenta possibilita ao usuário gerar os casos de teste a partir da tabela de decisão gerada, utilizando um modelo padrão pré-definido.

Logo no início do desenvolvimento da ferramenta, notou-se que o GWT (*Google Web Toolkit*) não era a ferramenta ideal para a construção deste trabalho. O principal motivo da

desistência do uso do GWT está no fato de a ferramenta estar sendo descontinuada pela Google. Para facilitar a construção e manutenção do software, optou-se pela substituição do GWT pelo JSP.

O sistema foi testado e validado por analistas de teste da Senior Sistemas, de Blumenau, Santa Catarina. Os analistas validaram os requisitos propostos e as regras de negócio aplicadas ao sistema, e concluíram que ambos foram atendidos. Além disto, informaram que o sistema é intuitivo, rápido e de fácil manuseio.

Como melhorias para o trabalho desenvolvido, os analistas de teste sugeriram a possibilidade de estruturar os testes do sistema, de modo que o usuário crie projetos dentro da ferramenta para separar quais testes pertencem a cada projeto. Outra sugestão apresentada é criar uma funcionalidade para mostrar o percentual de cobertura dos testes das tabelas reduzidas.

Ao comparar a ferramenta desenvolvida com os trabalhos correlatos, percebe-se que todas as ferramentas possuem como objetivo a automatização de atividades do processo de teste de software e a redução do tempo gasto pelo analista de teste nestas atividades. Porém os trabalhos diferem-se na aplicação desenvolvida. O trabalho de Caldeira (2010) propõe a criação manual de tabelas de decisão para, posteriormente, gerar os casos de teste, e o trabalho de Bonecher (2008) está focado na automação do planejamento e controle dos testes.

A ferramenta desenvolvida por Lima (ca. 2010) comporta-se de modo muito semelhante à ferramenta desenvolvida neste trabalho. A principal diferença entre ambas está no fato de a ferramenta desenvolvida por Lima não preencher automaticamente a ação que será tomada em cada condição testada. Por conta disto, a ferramenta desenvolvida por Lima também não identifica quais condições da tabela são equivalentes e não possibilita ao usuário gerar os casos de testes utilizando um modelo padrão.

O Quadro 9 apresenta a comparação entre a ferramenta desenvolvida neste trabalho e os trabalhos correlatos apresentados na seção 2.5.

Quadro 9 - Comparação com trabalhos correlatos

Características	Caldeira	Bonecher	Lima	Trabalho atual
Linguagem de Programação	?	Pascal	?	Java / JSP
Banco de dados	?	MySQL	?	MySQL
Ambiente	*Desktop*	Web	Web	Web
Automatiza plano de testes?	Não	Sim	Não	Não
Automatiza tabela de decisão?	Não	Não	Sim	Sim
Automatiza casos de testes?	Sim	Não	Não	Sim
Permite exportar os artefatos?	Não	Não	Sim	Sim

4 CONCLUSÕES

O processo de teste de software engloba diferentes atividades, executadas por analistas e testadores. Uma das principais atividades é a análise de teste. Nesta atividade, o analista de teste cria diferentes artefatos, dentre eles a tabela de decisão e os casos de teste. A criação manual destes artefatos consome muito tempo e corre o risco de não abranger todos os testes necessários para o sistema.

Os objetivos propostos neste trabalho foram alcançados. Foi desenvolvida uma ferramenta que gera a tabela de decisão baseando-se nas variáveis e regras previamente cadastradas pelo usuário. A ferramenta identifica as condições de testes equivalentes, permite que o usuário as exclua e gere os casos de teste por meio de um *template* padrão. Ao automatizar estas atividades, reduz-se o risco de falha humana no esquecimento de condições de testes importantes e diminui-se o tempo gasto pelo analista de testes por reduzir a quantidade de trabalho manual.

O desenvolvimento e a conclusão deste trabalho propiciaram à autora um conhecimento mais aprofundado sobre as técnicas de teste existentes e os processos mais utilizados para trabalhar com teste de software. Além disso, foi possível aplicar os conhecimentos adquiridos durante o curso de graduação, sendo necessário realizar toda a análise do problema, a definição dos requisitos, o desenvolvimento da ferramenta e a validação e verificação com usuários.

4.1 EXTENSÕES

A seguir são apresentadas algumas funcionalidades como sugestões de implementações futuras:

a) implementar a gestão de usuários, de modo que as variáveis e regras cadastradas para um usuário não sejam apresentadas para os outros usuários;
b) implementar a estruturação dos testes, permitindo que o usuário cadastre projetos e crie testes separados para cada projeto que for atuar;
c) armazenar todas as tabelas de decisão e casos de teste gerados pelo usuário no banco de dados, para posterior manutenção;
d) apresentar o percentual de cobertura do teste das tabelas de decisão reduzidas;
e) permitir a manutenção dos casos de teste gerados, permitindo que o usuário realize alterações específicas em cada caso de teste gerado, conforme necessidade.

REFERÊNCIAS

BONECHER, Bruna Tatiane. **Ferramenta web de apoio ao planejamento e controle de teste de software**. 2008. 83 p. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2008. Disponível em: <http://www.bc.furb.br/docs/MO/2009/335630_1_1.pdf>. Acesso em: 1 abr. 2015.

CALDEIRA, Luiz Rodolfo Neves. **Geração semi-automática de massas de testes funcionais a partir da composição de casos de uso e tabelas de decisão**. 2010. 96 p. Dissertação (Mestrado em Informática) – Programa de Pós-Graduação em Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2010. Disponível em: <http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0821374_10_pretextual.pdf >. Acesso em: 18 mar. 2015.

CAMPOS, Fabrício. **Técnicas de modelagem de teste (parte 1).** [S.l.], 2009. Disponível em: <http://pt.slideshare.net/FabricioFFC/tcnicas-de-modelagem-de-teste-parte-1>. Acesso em: 18 mar. 2015.

DELAMARO, Márcio Eduardo; MALDONADO, José Carlos; JINO, Mario. **Introdução ao teste de software**. Rio de Janeiro: Campus, 2007. 394 p, il.

HEINEBERG, Ricardo. **Técnicas de teste funcional (caixa preta).** [S.l.], 2008. Disponível em: <http://testandosoftware.blogspot.com.br/2008/06/tcnicas-de-teste-funcional-caixa-preta.html>. Acesso: em 20 mai. 2015.

LIMA, Júlio. **x|decision.** [S.l.], [ca 2010]. Disponível em: <http://xdecision.co/>. Acesso em: 11 de out. 2015.

MARTINS, Eliane. **Testes baseados na especificação**: modelos de estado. [S.l.], 2010. Disponível em <http://www.ic.unicamp.br/~eliane/Cursos/Transparencias/VVTestes/Aula14-testescxp2-FSM.pdf>. Acesso: em 10 out. 2015.

MOLINARI, Leonardo. **Testes Funcionais de Software**. Florianópolis: Visual Books, 2008.

MOREIRA FILHO, Trayahú Rodrigues; RIOS, Emerson. **Projeto & engenharia de software**: teste de software. Rio de Janeiro: Alta Books, c2003. 210p, il.

PEZZÈ, Mauro; YOUNG, Michael. **Teste e análise de software**: processos, princípios e técnicas. Porto Alegre: Bookman, 2008. x, 512 p, il.

PRIMESOFT. **Teste de Software:** conheça a técnica do pairwise (matrizes ortogonais)**.** [S.l.], 2014. Disponível em: <http://www.primesoft.com.br/post-conheca-a-tecnica-do-pairwise.php>. Acesso em: 20 mai. 2015.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação**.3. ed.rev. e ampl. Rio de Janeiro : Brasport, 2005. xxii, 316 p, il.

RIOS, Emerson. **Análise de riscos em projetos de teste de software**. São Paulo : Alta Books, c2005. 131 p, il.

RIOS, Emerson. **Documentação de teste de software**: dissecando o padrão IEEE 829. 2. ed. Niterói: Imagem Art, 2010.

RIOS, Emerson; MOREIRA FILHO, Trayahú Rodrigues. **Teste de Software.** 3. ed. Rio de Janeiro: Alta Books, 2013.

VIEIRA, Luiz Gustavo Schroeder. **Técnicas avançadas de tabelas de decisão.** [S.l.], 2010. Disponível em: <http://www.testavo.com.br/2010/08/tecnicas-avancadas-tabelas-de-decisao.html>. Acesso em: 17 mar. 2015.

APÊNDICE A – Descrição dos Casos de Uso

Este Apêndice apresenta a descrição dos casos de uso previstos no diagrama apresentado na seção 3.2.3.

No Quadro 10 apresenta-se o caso de uso `Cadastrar Objetivo`.

Quadro 10 – Descrição do caso de uso UC01

Caso de uso	UC01 – Cadastrar Objetivo
Descrição	Permite que o usuário cadastre um objetivo para o teste que será criado. O objetivo informado será armazenado no banco de dados e vinculado a tabela de decisão que será gerada.
Ator	Analista de testes
Fluxo principal	1. Analista de testes acessa o sistema 2. Sistema apresenta a tela de cadastro de objetivo 3. Analista de testes informa o objetivo 4. Sistema armazena o objetivo no banco de dados

No

Quadro 11 apresenta-se o caso de uso `Manter Variáveis`.

Quadro 11 - Descrição do caso de uso UC02

Caso de uso	UC02 – Manter Variáveis
Descrição	Este caso de uso tem como objetivo permitir que o usuário cadastre as variáveis que serão utilizadas nos testes, acompanhadas do tipo e de valores que deverão ser testados. Todas as variáveis cadastradas serão armazenadas no banco de dados para que possam ser reutilizadas futuramente. As variáveis cadastradas serão apresentadas ao usuário, e este poderá ativar, desativar, alterar e excluir variáveis, conforme necessidade do teste que será criado.
Ator	Analista de testes
Fluxo principal	1. Analista de testes acessa a tela de cadastro de variáveis 2. Sistema apresenta a lista de variáveis cadastradas 3. Sistema apresenta os campos para cadastro de novas variáveis 4. Analista de testes informa nome, tipo e valores para a nova variável 5. Analista de testes informa se variável estará ativa no próximo teste 6. Sistema armazena a nova variável no banco de dados 7. Sistema apresenta a nova variável na lista de variáveis cadastradas
Fluxo alternativo	No passo 2, usuário poderá optar por ativar ou desativar variáveis cadastradas: 2.1 Ativar variável 2.2 Desativar variável
Cenário – alteração	No passo 2, o usuário poderá alterar variável: 2.1 Analista de testes seleciona uma variável para alterar 2.2 Analista de testes realiza as alterações desejadas 2.3 Analista de testes pressiona o botão alterar 2.4 Sistema grava as alterações no banco de dados 2.5 Sistema apresenta variável alterada na lista de variáveis
Cenário – exclusão	No passo 2, o usuário poderá excluir variável: 2.1 Analista de testes seleciona uma variável para excluir 2.2 Analista de testes pressiona o botão excluir

	2.3 Sistema exclui variável do banco de dados 2.4 Variável excluída não é mais apresentada na lista de variáveis

No Quadro 12 apresenta-se o caso de uso `Manter Regras`.

Quadro 12 - Descrição do caso de uso UC03

Caso de uso	UC03 – Manter Regras
Descrição	Permite ao usuário incluir e excluir as regras que serão consideradas na tabela de decisão O usuário poderá definir uma ação para cada combinação da tabela, e cada ação deverá ser formada por uma ou mais variáveis.
Ator	Analista de testes
Pré-condição	As regras devem estar cadastradas no sistema.
Fluxo principal	1. Analista de testes acessa a tela de cadastro de regra 2. Sistema apresenta regras cadastradas 3. Analista de testes seleciona variável desejada 4. Analista de testes seleciona valor válido para a variável desejada 5. Analista de testes define qual ação deverá ser tomada para a combinação de teste

No Quadro 13 apresenta-se o caso de uso `Gerar Tabela de Decisão`.

Quadro 13 - Descrição do caso de uso UC04

Caso de uso	UC04 – Gerar Tabela de Decisão
Descrição	Sistema gera a tabela de decisão contendo todas as condições de teste possíveis, combinando os valores de teste que foram informados para todas as variáveis e definindo qual ação deverá ser tomada para cada condição gerada.
Ator	Analista de testes
Pré-condição	As variáveis e regras devem estar cadastradas no sistema.
Fluxo principal	1. Analista de testes pressiona o botão “Gerar Tabela de Decisão” 2. Sistema cria combinações entre todos os valores de teste informados para as variáveis e definindo uma ação para cada combinação 3. Sistema apresenta a tabela de decisão completa

No Quadro 14 apresenta-se o caso de uso `Manter Tabela de Decisão`.

Quadro 14 - Descrição do caso de uso UC05

Caso de uso	UC05 – Manter Tabela de Decisão
Descrição	Sistema identifica as condições de teste equivalentes e permite que o usuário reduza a tabela de decisão para o tamanho recomendado ou para o tamanho mínimo. Sistema também permite que o usuário exclua qualquer condição da tabela.
Ator	Analista de testes
Pré-condição	A tabela de decisão deve ter sido gerada.
Fluxo principal	1. Sistema identifica as condições de teste equivalentes 2. Analista de teste reduz tabela de decisão 3. Analista de teste exclui condições da tabela
Fluxo alternativo	No passo 2, usuário poderá optar por reduzir a tabela de decisão ao tamanho recomendado ou ao tamanho mínimo: 2.1 Reduzir tabela ao tamanho recomendado 2.2 Reduzir tabela ao tamanho mínimo

No Quadro 15 apresenta-se o caso de uso `Gerar Casos de Teste`.

Quadro 15 - Descrição do caso de uso UC06

Caso de uso	UC06 – Gerar Casos de Teste
Descrição	Sistema permite que o usuário defina um modelo padrão (*template*) para os casos de teste, e transforma cada condição de teste da tabela em um caso de teste.
Ator	Analista de testes
Pré-condição	A tabela de decisão deve ter sido gerada.
Fluxo principal	1. Analista de teste cadastra o *template* 2. Sistema transforma condições de teste em casos de teste 3. Sistema apresenta os casos de teste

No Quadro 16 apresenta-se o caso de uso `Exportar Tabela`.

Quadro 16 - Descrição do caso de uso UC07

Caso de uso	UC07 – Exportar Tabela
Descrição	Permite ao usuário exportar a tabela gerada para arquivos no formato Excel ou PDF, auxiliando a criação dos casos de teste por meio da consulta às informações da tabela gerada.
Ator	Analista de testes
Pré-condição	Tabela de decisão gerada pelo sistema.
Fluxo principal	1. Sistema apresenta a tabela de decisão 2. Analista de testes pressiona o botão “Exportar Tabela para PDF” 3. Sistema apresenta a tabela gerada em um arquivo PDF
Fluxo alternativo	1. No passo 2, o analista de teste pode optar por exportar a tabela de decisão para um arquivo no formato Excel: Analista de testes pressiona o botão “Exportar Tabela para Excel” 2. Sistema apresenta a tabela gerada em um arquivo Excel

APÊNDICE B – Dicionário de Dados

Este Apêndice apresenta a descrição das tabelas criadas no banco de dados para armazenar as informações geradas pela ferramenta. Os tipos de dados utilizados na definição dos atributos são:

a) `integer`: armazena caracteres numéricos inteiros;

b) `varchar`: armazena dados alfanuméricos com até 255 caracteres.

O Quadro 17 apresenta a descrição da tabela objetivo.

Quadro 17 - Descrição da tabela objetivo

Entidade: Objetivo					
Descrição: Armazena o objetivo do teste					
Atributo	**Tipo**	**Descrição**	**Tamanho**	**Chave Primária**	**Chave Estrangeira**
nr_sequencia	int	código do objetivo	-	sim	não
ds_objetivo	varchar	descrição do objetivo	45	não	não

O Quadro 18 apresenta a descrição da tabela regra.

Quadro 18 - Descrição da tabela regra

Entidade: Regra					
Descrição: Armazena as regras					
Atributo	**Tipo**	**Descrição**	**Tamanho**	**Chave Primária**	**Chave Estrangeira**
nr_sequencia	int	código da regra	-	sim	não
ds_acao	varchar	ação da regra	45	não	não
objetivo_nr_sequencia	int	objetivo vinculado à regra	-	não	sim

O Quadro 19 apresenta a descrição da tabela regra_variavel.

Quadro 19 - Descrição da tabela regra_variavel

Entidade: Regra_variavel					
Descrição: Armazena as variáveis pertencentes a cada regra cadastrada					
Atributo	**Tipo**	**Descrição**	**Tamanho**	**Chave Primária**	**Chave Estrangeira**
valores_variavel_nr_sequencia	int	código do valor pertencente à variável e vinculado à regra	-	não	sim
regra_nr_sequencia	int	código da regra	-	não	sim
nr_sequencia	int	código do vinculo entre regra	-	sim	não

		e valor variável			

O Quadro 20 apresenta a descrição da tabela variavel.

Quadro 20 - Descrição da tabela variavel

Entidade: Variavel					
Descrição: Armazena as variáveis					
Atributo	**Tipo**	**Descrição**	**Tamanho**	**Chave Primária**	**Chave Estrangeira**
nr_sequencia	int	código da variável	-	sim	não
ds_nome	varchar	nome da variável	20	não	não
ie_tipo	varchar	tipo da variável	5	não	não
ds_valor	varchar	valores válidos da variável	45	não	não
ie_ativa	varchar	situação da variável	1	não	não

O Quadro 21 apresenta a descrição da tabela valores_variavel.

Quadro 21 - Descrição da tabela valores_variavel

Entidade: Valores_variável					
Descrição: Armazena os valores pertencentes a cada variável					
Atributo	**Tipo**	**Descrição**	**Tamanho**	**Chave Primária**	**Chave Estrangeira**
nr_sequencia	int	código do valor da variável	-	sim	não
ds_opcao	varchar	nome da opção de valor da variável	20	não	não
variável_nr_sequencia	int	código da variável	-	não	sim