UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

ISSN 0103-2569

SELECTT: FERRAMENTA PARA AUTOMATIZAÇÃO DO PROCESSO DE SELEÇÃO DE TÉCNICAS DE TESTE DE _SOFTWARE_ CONCORRENTE

FELIPE MOREIRA MOURA
SILVANA MORITA MELO
SIMONE DO ROCIO SENGER DE SOUZA

Nº 423

RELATÓRIOS TÉCNICOS

São Carlos – SP
Jan./2018

Instituto de Ciências Matemáticas e de Computação

ISSN – 0103-2569

SeleCTT: Ferramenta para automatização do processo de seleção de técnicas de teste de *software* concorrente

Felipe Moreira Moura
Silvana Morita Melo
Simone do Rocio Senger de Souza

RELATÓRIOS TÉCNICOS DO ICMC

São Carlos
12 de janeiro de 2018

Resumo

Neste relatório técnico é apresentada a ferramenta *SeleCTT* que automatiza o processo de seleção de técnicas de teste de *software* concorrentes, com o objetivo de auxiliar profissionais e pesquisadores nesse campo a identificar técnicas de teste de software concorrente adequadas a seu projeto de *software*. Fundamentos da Engenharia de *Software* Empírica (ESE) são usados para possibilitar a extração de informação relevante ao processo de seleção de técnicas, permitindo que desenvolvedores consigam respostas para questões sobre qual técnica é mais efetiva, menos custosa e que pode revelar mais defeitos em um determinado cenário. Como contribuição científica, a ferramenta proposta além de disponibilizar acesso às informações sobre as técnicas que compõem o estado atual da área de teste de software concorrente, auxilia o processo de tomada de decisão sobre a escolha da técnica mais adequada a um projeto de software, com menos esforço e baseada em fundamentação teórica e conhecimento de especialistas que têm trabalhado amplamente no estudo e aplicação de técnica de teste neste domínio de aplicação.

Palavras-chave: Programas Concorrentes; Engenharia de *Software*; Corpo de Conhecimento; Teste de *Software*, Teste de Programas Concorrentes, Seleção de Técnicas de teste de *Software*.

Sumário

Lista de Tabelas

Lista de Figuras

CAPÍTULO 1: INTRODUÇÃO

Antes do surgimento dos multicomputadores e multiprocessadores, tradicionalmente na área de ciência da computação, um software era desenvolvido buscando resolver um problema de maneira sequencial, além de ser executado por um único processador em um único computador, onde apenas uma instrução de máquina é executada por vez. Embora essa abordagem funcione e seja útil a diversos tipos de aplicações, ela impõe limitações que podem inviabilizar projetos de software ou algoritmos que requerem uma enorme capacidade computacional (BARNEY, 2017).

Com o intuito de solucionar esse problema, uma alternativa é o desenvolvimento de aplicações paralelas em contrapartida à programação sequencial. Os princípios nesta área envolvem: divisão do problema a ser resolvido e execução paralela de diferentes módulos, a fim de se obter redução no tempo de execução, maior tolerância a falhas e soluções mais elegantes para problemas intrinsecamente paralelos (GRAMA, GUPTA, et al., 2003).

A programação de alto desempenho (paralela/concorrente) visa resolver problemas de projetos de software ou algoritmos que requerem uma enorme capacidade computacional, e na era do Big Data (JIN, WAH, et al., 2015) é imprescindível a presença de programas que executem em paralelo; com objetivo de minimizar custos e otimizar o tempo de execução.

O desenvolvimento de software concorrente é mais complexo que o desenvolvimento de software sequencial, pois envolve um maior número de etapas no processo de desenvolvimento e a necessidade de tratar iteração entre processos/threads, a comunicação e o compartilhamento de memória. A interação entre processos (ou threads) pode ser realizado por meio de passagem de mensagem (envio e recebimento) ou através de acesso a memória compartilhada (sendo local ou global).

Programas concorrentes são não determinísticos, essa característica dificulta a reprodução de uma situação em que um programa falhe ou não apresente o resultado esperado, pois o resultado é probabilístico e não determinista (ARORA, BHATIA e SINGH, 2016). Sendo assim, para uma mesma entrada podem existir diferentes saídas com o mesmo ou diferentes resultados (CARVER e TAI, 2005).

Essas características tornam o teste de software concorrente mais complexo que o teste de software tradicional (sequencial). Segundo Myers (1979) utilizar da exaustão para cobrir

todas as possibilidades e caminhos é uma tarefa quase impossível, de elevado custo e com resultados não tão eficientes em relação ao tempo necessário, principalmente no caso em que a técnica de teste adotada não for a mais adequada para o projeto em desenvolvimento que está sendo testado.

A atividade de teste corresponde em alguns projetos em mais da metade dos custos de desenvolvimento. No entanto, a perda monetária relacionada a softwares defeituosos alcança a casa de bilhões de dólares ao ano, sugerindo uma deficiência por parte da indústria na execução da atividade de teste, necessitando de medidas mais eficientes com o objetivo de diminuir o prejuízo anual (VEGAS e BASILI, 2005).

Revisões sistemáticas podem ser uma alternativa eficaz destinada a apoiar profissionais da área, pois fazem um panorama das pesquisas em um tópico, neste caso na área de teste de software concorrente, analisando as técnicas que têm sido propostas neste contexto, apoiando o desenvolvimento de diretrizes baseadas em evidências para os profissionais com objetivo de auxiliar no processo de tomada de decisão (KITCHENHAM, BRERETON, et al., 2009). No geral, a revisão sistemática é utilizada como um meio de identificar, avaliar e interpretar todas as pesquisas disponíveis e relevantes para um determinado assunto de interesse (ARORA, BHATIA e SINGH, 2016).

Atualmente profissionais na área de teste tendem a repetir os mesmos padrões durante o desenvolvimento de um projeto de software. Esse comportamento é mais agravante no caso de projetos de software concorrente, devido a complexidade inerente de desenvolvimento, e na dificuldade da equipe de teste em adotar novas técnicas de teste, que até então nunca antes foram utilizadas. Além disso, "desenvolvedores tendem a não compartilhar o conhecimento que adquirirem com outras pessoas. Esse comportamento reduz a chance de aprender sobre as experiências de outros desenvolvedores" (VEGAS e BASILI, 2005).

A maioria de informações sobre técnicas de teste de software encontram-se em artigos científicos ou mesmo em livros e relatórios técnicos e dificultam a escolha das técnicas de testes mais adequadas para determinado projeto. Nesse contexto, Vegas e Basili (2005) propõem um esquema de caracterização das técnicas disponíveis, a fim de criar um repositório que contenha apenas informações relevantes ao processo de seleção de técnicas.

Neste repositório, estão descritas todas as técnicas com o mesmo padrão, para que uma decisão possa ser tomada sobre a adoção ou não de uma técnica de teste baseando-se no processo de seleção (VEGAS e BASILI, 2005). Através da seleção, é possível resolver um problema fundamental: "A identificação apropriada das características relevantes para seleção"

(VEGAS e BASILI, 2005). No entanto, o trabalho proposto por Vegas e Basili (2005) é exclusivo para softwares sequenciais, não se adequando ao contexto de software concorrente. Por isso instanciou-se o presente projeto no contexto de programação concorrente.

Além disso, este trabalho relaciona-se com o trabalho de doutorado, em andamento, de Melo (2017), o qual tem por objetivo investigar, através de um framework, as principais técnicas de teste de software que têm sido propostas para a área de programação concorrente, a fim de oferecer um mecanismo de apoio à caracterização e seleção sistemática dessas técnicas. Este framework é desenvolvido seguindo os princípios da Engenharia de Software

Experimental (ESE). O presente projeto desenvolve a infraestrutura computacional de apoio para o projeto de doutorado de Melo (2017), a fim de automatizar o processo de seleção e minimizar os erros e falhas humanas durante o processo.

Em geral, os programas concorrentes apresentam resultados com comportamento não determinístico (ARORA, BHATIA e SINGH, 2016), ou seja, uma mesma entrada pode gerar resultados diferentes na saída, um comportamento não presente em programas sequenciais.

Algumas outras características cruciais presentes em programas concorrentes são relacionadas a comunicação e sincronização entre computadores, processadores ou threads, que podem resultar erros específicos desse tipo de aplicação.

Essas, dentre outras características tornam a atividade de teste nesse contexto ainda mais complexa, se comparada ao contexto de software tradicional (sequencial). Nesse sentido, diversas abordagens têm sido propostas tentando combater esses desafios. Uma grande variedade de técnicas tem sido proposta nos últimos anos para apoiar o teste de programas concorrentes (SOUZA, BRITO, et al., 2011).

Nesse contexto, este trabalho propõe uma infraestrutura computacional para automatizar o processo de seleção de técnicas de teste de software concorrentes, auxiliando profissionais e pesquisadores nesse campo a identificar técnicas adequadas a seu projeto de software. Além disso, espera-se que a ferramenta auxilie o processo de tomada de decisão no processo de desenvolvimento do software, proporcionando ganhos em tempo de condução do processo de teste e auxilie na garantia de qualidade das aplicações concorrentes, com a utilização de técnicas de teste de software.

CAPÍTULO 2: SeleCTT

2.1. Considerações Iniciais

Este capítulo tem como objetivo apresentar a infraestrutura da ferramenta *SeleCTT*. Primeiramente, é descrito o panorama geral do projeto, detalhando os objetivos, a arquitetura, os módulos presentes, as tecnologias adotadas durante o desenvolvimento da ferramenta e o servidor em que a ferramenta se encontra hospedada.

É descrito também o processo de desenvolvimento do corpo de conhecimento, com enfoque no processo de caracterização das técnicas e identificação de informações úteis ao processo de seleção. Um processo sistemático de seleção é proposto, utilizando as informações a respeito das técnicas e cálculos matemáticos a fim de ranquear as técnicas de teste mais adequadas a um projeto de software específico.

São apresentados os resultados um exemplo de aplicação da metodologia. E por fim, listamos algumas dificuldades e limitações encontradas durante o desenvolvimento do projeto.

2.2. Especificações

2.2.1. Perspectiva do *Software*

SeleCTT é uma página web, com *cross-browser*, que retorna resultados sobre seleção de técnicas de teste de software concorrente através da utilização de informações extraídas de um corpo de conhecimento a fim de encontrar a melhor correspondência. O sistema recebe um conjunto de parâmetros de entrada, informações sobre o projeto de software concorrente a ser desenvolvido, processa essa entrada em conjunto com um banco de dados (base de conhecimento), com o objetivo de retornar uma lista com as técnicas de teste mais adequadas. Esse sistema deve garantir precisão e a confiabilidade nos resultados dado as informações de entrada.

2.2.2. Classes de Usuários e Características

Existem duas classes de usuários.

- **Administradores:** que são responsáveis pelo cadastramento, alteração e remoção de usuários. Além disso, são responsáveis pelo controle das informações inseridas no corpo de conhecimento e o gerenciamento do sistema.

- **Usuários:** que tem como função a utilização da *SeleCTT*, portanto podem fornecer informações sobre novas técnicas de teste de software concorrente, entrar com informações sobre seu projeto a fim de selecionar dentre as técnicas existentes quais as mais adequadas a este projeto de software.

2.2.3. Diagramas

Os diagramas ilustrando o funcionamento da ferramenta *SeleCTT* encontram-se nas figuras abaixo. Para a elaboração dos diagramas utilizou-se como referência as obras de (PRESSMAN, 2010, cap. 5 e 6) e (SOMMERVILLE, 2007, cap. 4).

A Figura 1 ilustra o diagrama de domínio, que descreve os atributos necessários das classes da ferramenta para o seu funcionamento. A Figura 2 ilustra o diagrama de casos de uso, que descreve as principais funcionalidades da ferramenta, como: autenticação de usuário, inserção de novas técnicas, inserção de informações de software concorrente, gerenciamento do sistema. A Figura 3 ilustra o diagrama de atividades, que descreve o fluxo das atividades da ferramenta, como: controle de acesso, inserção de técnicas, administração do sistema e seleção.

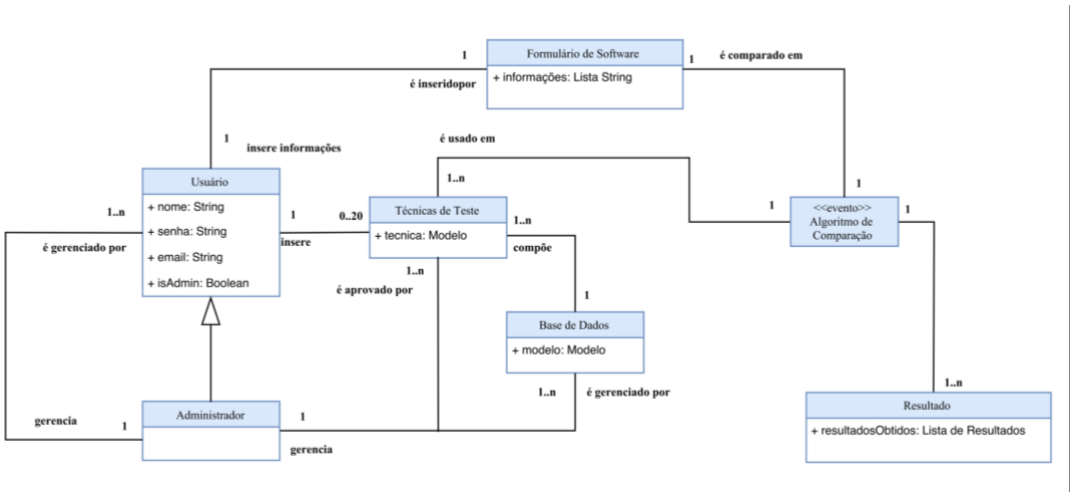

Figura 1: Diagrama de domínio da ferramenta *SeleCTT*.

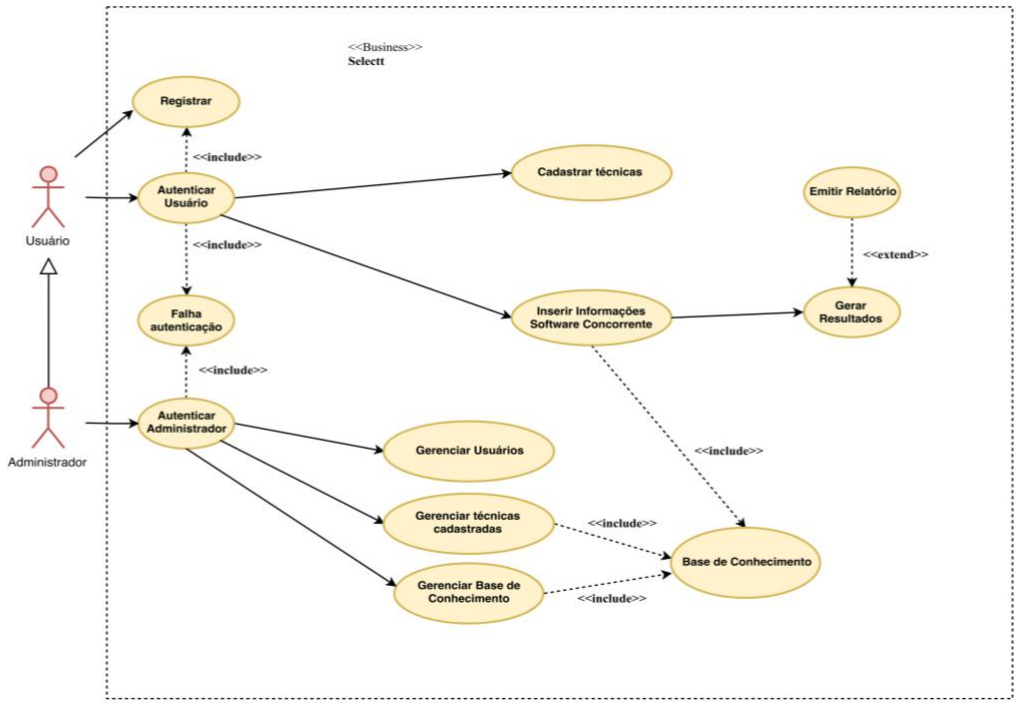

Figura 2: Diagrama de casos de uso da ferramenta *SeleCTT*.

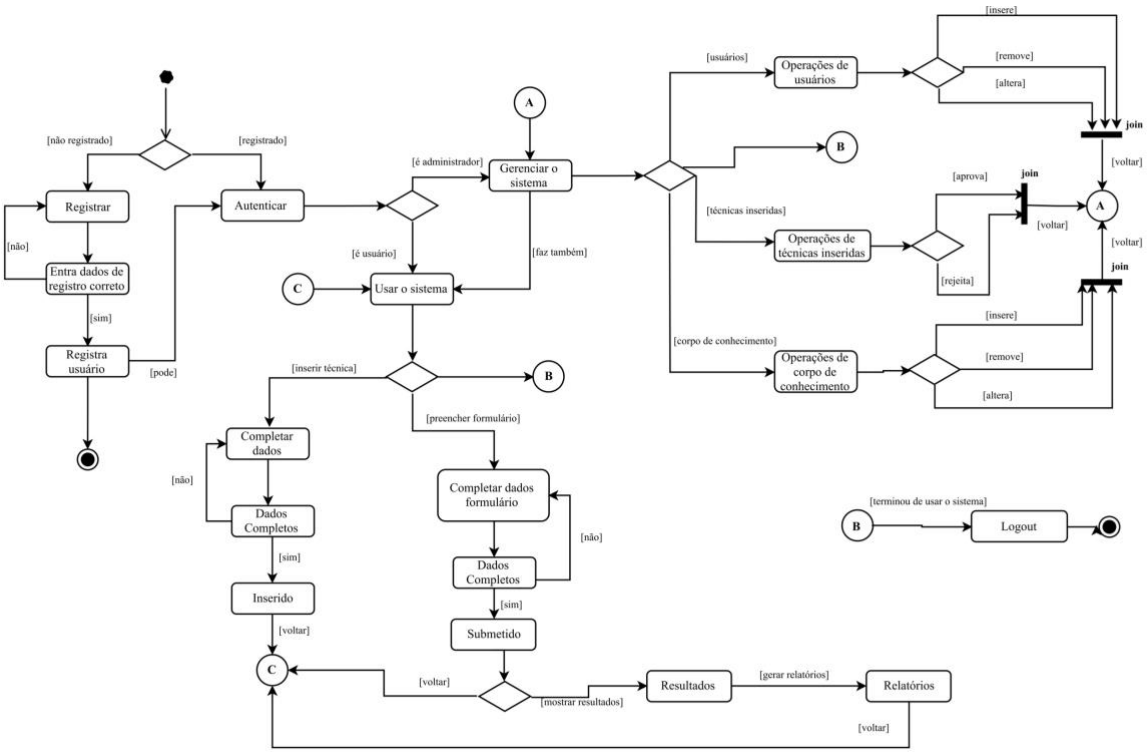

Figura 3: Diagrama de atividades da ferramenta *SeleCTT*.

2.2.4. Ambiente de Operação

A *SeleCTT* opera em um ambiente web hospedado em um servidor web do Laboratório de Engenharia de Software do Instituto de Ciências Matemáticas e de Computação (LABES-ICMC/USP), disponível no endereço: http://www.labes.icmc.usp.br/~selectt/.

O sistema é *cross-browser*, podendo ser executado em diferentes web-browsers como *Safari, Mozilla Firefox e Google Chrome*.

2.3. Projeto

A *SeleCTT*, ou *Infrastructure for **Sele**ction of **C**oncurrent **T**esting **T**echniques* (Infraestrutura para Seleção de Técnicas de Teste de *Software* Concorrente), é uma infraestrutura *web* para automatização do processo de seleção de técnicas de teste de software concorrente.

Os métodos utilizados na ferramenta *SeleCTT*, no processo de automatização de seleção das técnicas de teste de software concorrente mais adequadas, são baseados nos estudos de Dias-Neto e Travassos (2009) e Vegas e Basili (2005). Esses estudos propõem um processo de seleção de técnicas de teste baseando-se nas informações do projeto em desenvolvimento, conforme discutido da seção 2.4.1. Porém, esses trabalhos consideram apenas o contexto de software sequencial, não considerando as características da programação concorrente que afetam a atividade de teste.

No entanto, a ferramenta desenvolvida neste projeto é composta por um corpo de conhecimento com as técnicas de teste de software concorrente de diversas categorias, e não apenas técnicas de teste baseado em modelos, como em Dias-Neto e Travassos (2009).

2.4. Arquitetura da ferramenta

A ferramenta *SeleCTT* tem como principal objetivo auxiliar a tomada de decisão na técnica de teste de software concorrente mais adequada a um projeto em desenvolvimento (independente da fase em que se encontra o projeto), através dos atributos do projeto que são fornecidos como entrada à ferramenta de teste. A partir dessa entrada, são cruzados os dados do projeto com os dados contidos no corpo de conhecimento através da técnica de seleção, gerando uma lista ordenada e com classificação das técnicas de teste mais adequada para este projeto. Uma visão geral da arquitetura da ferramenta *SeleCTT* é dada pela Figura 4.

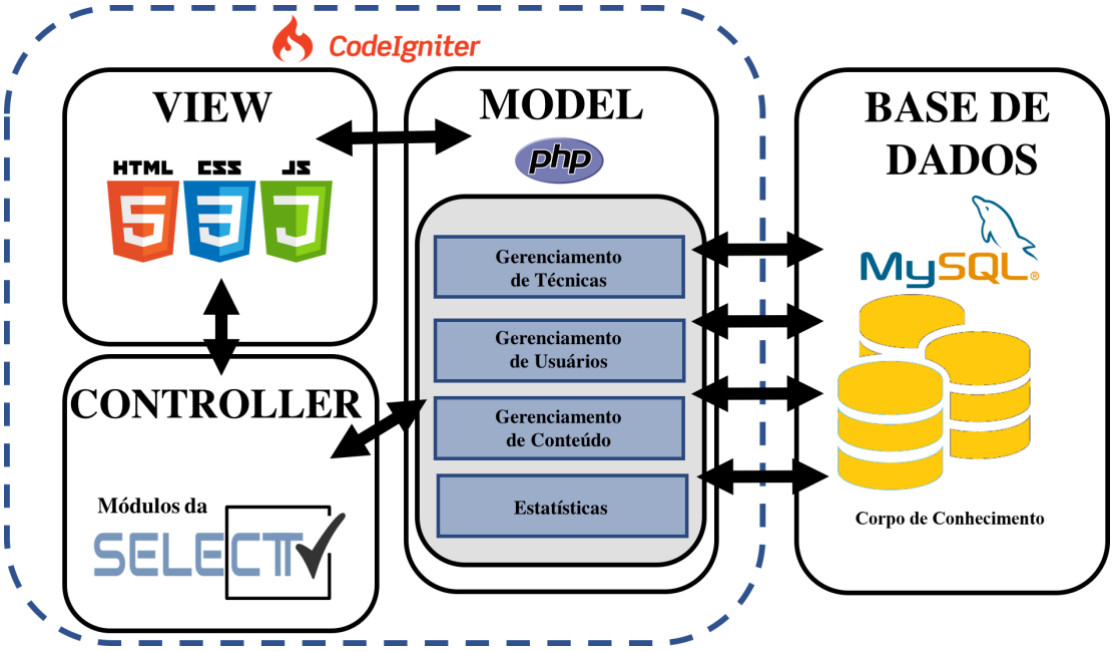

Figura 4: Arquitetura da ferramenta *SeleCTT*.

A ferramenta *SeleCTT* está dividida em quatro módulos, que são definidos:

(a) Módulo de Inserção: que recebe como entrada informações a respeito do projeto de software a ser desenvolvido, também recebe como entrada informações para inserção de técnicas de teste de *software* concorrente no corpo de conhecimento.

(b) Módulo de Administração do sistema: responsável pelo gerenciamento do corpo de conhecimento, dos usuários e do conteúdo presentes nas páginas.

(c) Módulo de Controle: Responsável pelo controle dos atributos e critérios utilizados para a caracterização das técnicas de teste.

(d) Módulo de apresentação dos Resultados: responsável por comparar as informações extraídas do módulo (a) utilizando critérios de adequação definidos no modulo (c) a fim de fornecer uma lista de recomendação de técnicas ranqueadas de acordo com seu nível de relevância ao projeto.

Na construção da infraestrutura web foi desenvolvido tanto o *front-end* como o *back-end* da ferramenta *SeleCTT*.

Para o *front-end*, as páginas da ferramenta são compostas por *HTML5*, *CSS3*, proporcionando um código limpo e de fácil implementação, além do suporte pela maioria dos *browsers* no mercado. Também utilizou *JavaScript*, permitindo páginas dinâmicas, no qual

certas requisições são feitas por AJAX na API desenvolvida pela ferramenta para solicitar informações do servidor de forma assíncrona, sem prejudicar a experiência do usuário.

Para o *back-end* utilizou-se do *framework* do *CodeIgniter*[1], para desenvolvimento de aplicações PHP, possibilitando o desenvolvimento de aplicações de forma mais rápida e eficaz, além de proporcionar uma rápida curva de aprendizado. Este *framework* possui uma arquitetura MVC (*Model-View-Controller)*, a qual separa a aplicação em três camadas distintas, mas complementares entre si, para a implantação de interfaces de usuários. As camadas dessa arquitetura são compostas por: a camada de representação para o usuário (*view*), a camada de manipulação de dados e regras (*model*) e a camada de controle (*controller*) que faz a ponte entres as outras duas camadas.

Além disso, o *Codeigniter* contém um ótimo conjunto de bibliotecas para tarefas comuns, uma interface simples e uma estrutura lógica para acesso a estas bibliotecas além de ter suporte a conexão com banco de dados, o que possibilita a utilização e adoção do *MySQL* para armazenar os dados da ferramenta.

2.4.1. Corpo de Conhecimento

O Corpo de Conhecimento da ferramenta *SeleCTT* atua como uma base de dados, contendo todas as técnicas de teste de software concorrente da revisão sistemática em Souza e Brito (2017). O corpo de conhecimento foi desenvolvido utilizando *MySQL*[2], um sistema de gerenciamento de banco de dados (SGBD) *open source*. Uma ilustração de armazenamento das técnicas de teste no corpo de conhecimento encontra-se na Figura 5.

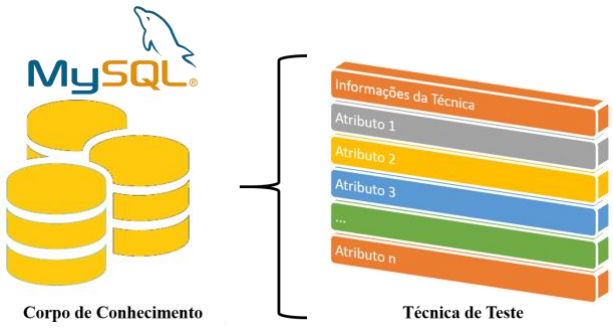

Figura 5: Caracterização das técnicas de teste e seus atributos.

[1] Disponível em: https://codeigniter.com/

[2] Para maiores informações acesse: https://www.mysql.com

Para cada técnica presente no corpo de conhecimento, estão presentes campos com as informações de identificação da técnica, e vinte e um campos com atributos que são utilizados para o cálculo de seleção da melhor técnica. A Tabela 1 contém uma lista com todos os atributos presentes no corpo de conhecimento e estão classificados por categorias. Além disso, cada um desses atributos possui como texto o valor de armazenamento na base de dados. Os atributos são classificados em quatro categorias, como descrito a seguir:

1) Modelo de programação: agrupa os atributos que representam o modelo de programação utilizado para o teste de software.

2) Características gerais do teste: agrupa os atributos gerais às diferentes técnicas de teste, não apenas técnicas para software concorrente.

3) Características de teste concorrente: agrupa os atributos que descrevem aspectos de tecnologias de teste de software específicas para software concorrente.

4) Suporte à ferramentas de teste: agrupa os atributos que contém as informações da ferramenta de teste de software.

Para cada técnica no corpo de conhecimento são considerados os atributos apresentados na Tabela 1 nas técnicas de teste de software concorrente, apenas variando o conteúdo armazenado na base de dados em cada um desses atributos. Indo além, existe a possibilidade de um desses atributos não conter valor, neste caso específico, constará no campo como: "Sem informação", portanto não existe possibilidade de um campo conter o valor nulo, sempre existirá um valor para o atributo, independentemente do tipo de conteúdo.

O corpo de conhecimento preliminar é composto por 109 técnicas de teste de software concorrente, podendo ser expandido com a inserção de novas técnicas de teste de acordo com a evolução literatura técnica e científica na área.

Qualquer usuário da ferramenta com uma conta válida (registrado e com o e-mail confirmado) pode acrescentar uma técnica a base de dados, incrementando o número de técnicas presentes no corpo de conhecimento. Porém, as técnicas adicionadas a base de dados, por usuários que não são administradoras da ferramenta, aparecem como pendentes e não entram no corpo de conhecimento em produção. Como essa nova técnica consta como pendente, no corpo de conhecimento temporário, ela não aparecerá no cálculo de seleção da técnica de teste mais apropriada para determinado projeto de software concorrente.

Técnicas pendentes na ferramenta apenas poderão fazer parte do corpo de conhecimento em produção se forem aprovadas por um administrador. Esse sistema de pendência garante a confiabilidade das técnicas de teste de software concorrente presentes no corpo de conhecimento, pois passam por uma revisão antes de serem aprovadas, garantindo que conteúdos não adequados não sejam adicionados ao repositório, e afetem a natureza de funcionamento da ferramenta.

Tabela 1: Atributos de caracterização de técnicas de teste de software concorrente.

Categoria	Atributo	Descrição	Exemplo de valores
Modelo de programação	Plataforma de Execução	Plataforma de execução que o *software* sob teste executa.	Linux, Windows, Android
	Paradigma de Desenvolvimento	Paradigma concorrente para a interação do processo.	Shared memory, message passing, hybrid
	Linguagens de programação/Bibliotecas de tempo de de execução	Linguagem de programação e/ou bibliotecas de tempo de execução que foi desenvolvido o *software* em teste.	Java, C/MPI, C/Phtreads, Fortran, Ada, Erlang
Características gerais de testes	Técnica de teste	Tipo de técnica de teste proposto.	Structural, functional, fault-based, model-based
	Geração de dados de teste	Mecanismo para testar a geração de dados de teste.	Execution trace, reachability testing, threading schedules, location pairs
	Nível de teste	Nível de teste que a técnica foi aplicada.	Unit, integration, system, acceptance, regression
	Mecanismo de intercalação de sincronização	Tipo de mecanismo de intercalação e sincronização aplicado na técnica.	Reachability testing, Execution trace, Linearizability
	Entradas necessárias	Entrada esperada para um caso de teste.	Code, requirements, models, bitecode
	Saídas	Saída esperada para um caso de teste.	Percentage of coverage, execution time, bugs revealed
	Atributos de qualidade	Características de qualidade do *software* que a técnica é capaz de avaliar.	Effectiveness, efficiency, performance, scalability
	Tipo de estudo empírico	Tipo de estudo empírico aplicado para validar a técnica de teste.	Case study, controlled experiment
Características de teste concorrente	Análise de teste	Tipo de análise utilizada pela técnica de teste.	Static, dynamic, hybrid
	Paradigma concorrente para a interação do processo	Contexto de desenvolvimento/Objetivo do software sob teste	High performance computing, distributed applications, mobile, embedded systems
	Mecanismo de *replay*	Tipo de mecanismo de repetição usado para re-executar o programa durante o teste.	Monitoring, regression scheduler, record-replay, SYN-sequences
	Representação do programa	A representação do programa que captura a informação relevante para o teste.	State space graph, petri nets, prediction model
	Instrumentação	Contexto/objetivo do software concorrente.	HPC, distributed systems, mobile, embedded
	Redução do espaço de estados	A técnica utilizada para tratar o problema de explosão do estado.	Dynamic partial order reduction, fitness function, preemption bounding
	Erros concorrente	O tipo de bugs concorrentes identificados pela técnica de teste.	Data race, deadlock, atomicity violation, livelock
Suporte a ferramentas de teste	Existência de ferramenta de apoio	Nome da ferramenta, se a técnica apresentar uma ferramenta.	Contest, CHESS, JavaPathFinder
	Custo da ferramenta	O custo associado à ferramenta.	Academic, shareware, open source
	Plataforma da ferramenta	A plataforma de execução que a ferramenta opera.	Linux, Windows, Android

2.4.2. Módulo de Inserção

O módulo de inserção da ferramenta *SeleCTT* é responsável por receber, fornecer e registrar informações sobre técnicas de teste. Uma visão geral da interface desse módulo é apresentada no Apêndice A.

Todos os usuários registrados e com o e-mail confirmado tem acesso ao módulo de Inserção. Este módulo tem duas opções de funcionamento. A Figura 6 ilustra as duas opções de funcionamento do módulo de Inserção.

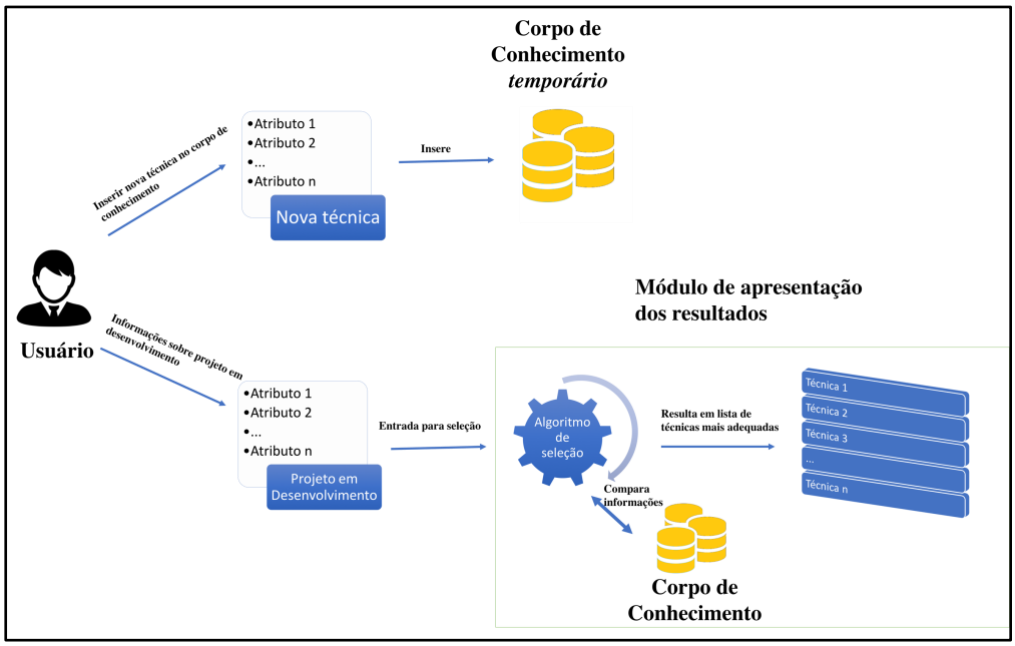

Figura 1: Exemplo de execução do módulo de Inserção, (a) seleção de técnicas e (b) adição de técnicas ao corpo de conhecimento.

A primeira opção, está relacionada com a entrada de informações a respeito do projeto de software a ser desenvolvido através de um formulário. Os resultados das melhores técnicas de teste para determinado projeto estão relacionados com as informações de entrada no módulo de inserção, pois estas informações serão utilizadas no algoritmo de seleção para serem cruzadas com os atributos das técnicas presentes no corpo de conhecimento. A segunda opção, está relacionada com a entrada através de um formulário, que contém as informações dos atributos de uma nova técnica de teste de *software* concorrente para inserção no corpo de conhecimento. No entanto, a inserção desta nova técnica, utilizando esse módulo, aparecerá como pendente, portanto a inserção desta nova técnica acontecerá no corpo de conhecimento temporário, que necessita da aprovação de um administrador para realmente se consolidar no repositório e ser considerada no cálculo de adequação durante o processo de seleção.

2.4.3. Módulo de Administração do Sistema

Apenas usuários administradores têm acesso a esse módulo, o módulo de Administração do Sistema da ferramenta *SeleCTT* possibilita o gerenciamento do corpo de conhecimento, os usuários registrados, o conteúdo da página e também funciona como um *dashboard*, apresentando os dados estatísticos de uso do sistema, a interface desse módulo encontra-se no Apêndice A.

O dashboard representa um painel de indicadores, como por exemplo: quantidade de técnicas presentes, quantidade de usuários, entre outros. Os indicadores deste projeto estão relacionados com as de técnicas de teste de software concorrente e o comportamento do usuário ao utilizar a ferramenta.

A Figura 7 é ilustrado a interação entre o módulo de administração do sistema, o corpo de conhecimento, o sistema de gerenciamento de usuários e conteúdo e o sistema de geração de dados estatísticos.

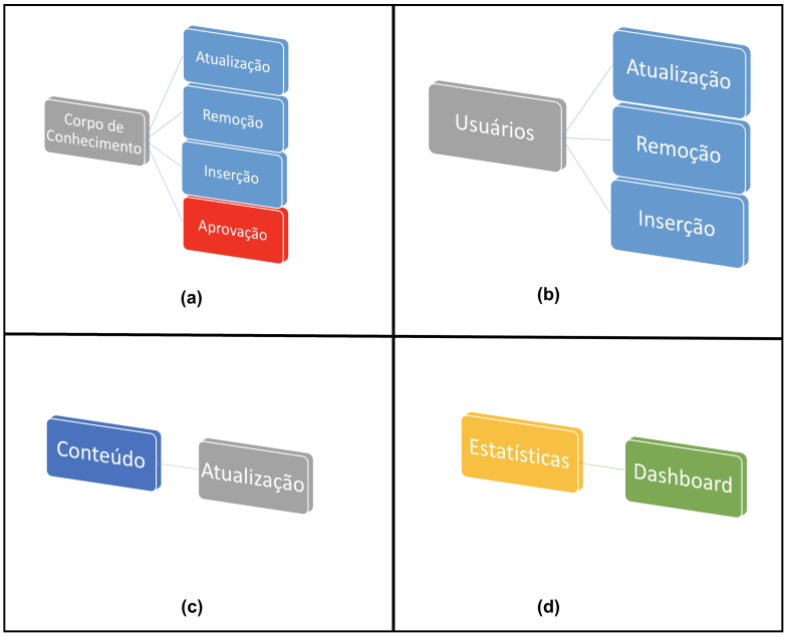

Figura 7: Módulo de administração do sistema. (a) o corpo de conhecimento, o sistema de gerenciamento de (b) usuários e (c) conteúdo e (d) o sistema de geração de dados estatísticos.

Dentre as atividades providas por esse módulo estão:

- Gerenciamento do corpo de conhecimento: responsável por controlar a inserção de novas técnicas, a edição de novos atributos, atualização de atributos e informações das técnicas presentes, remoção das técnicas e alteração das pendências nas técnicas, com objetivo de fazer parte do corpo de conhecimento em produção ou vice-versa.

- Gerenciamento dos usuários: responsável por controlar a inserção de novos usuários, a edição de usuários cadastrados, remoção de usuários, aprovação de usuários com verificação de e-mail pendente ou submissão novamente do e-mail de verificação, por fim, também gerencia as permissões, alterando o usuário para administrador ou vice-versa.

- Gerenciamento do conteúdo da ferramenta responsável pela edição do conteúdo presente nas páginas comuns (Página Inicial, Página de Publicações, Página de Pessoas, etc).

- Geração e apresentação de dados estatísticos responsável pelas informações de estatísticas das áreas que este módulo gerencia e dos outros módulos da ferramenta. As estatísticas monitoradas são: estatísticas sobre os usuários da ferramenta e das técnicas presentes no corpo de conhecimento. Indo além, também contém informações estatísticas geradas pelo módulo de controle e pelo módulo de inserção.

2.4.4. Módulo de Controle

O módulo de Controle da ferramenta SeleCTT gerencia os pesos que entram no cálculo da seleção da(s) técnica(s) de teste mais adequada(s) a um determinado projeto de software concorrente. Os pesos para cada um dos vinte e um atributos, que compões parte da informação de uma técnica de teste de software concorrente, são representados na Tabela 2.

Os pesos presentes em cada atributo nesse módulo são de natureza global na ferramenta. O gerenciamento dos pesos, de cada um dos vinte e um, consiste em atualizar os valores dos pesos, pois fazem parte de cada técnica de teste de software concorrente no corpo de conhecimento, sendo fundamentais para o cálculo da seleção da(s) técnica(s) mais adequada(s).

Um survey foi conduzido a fim de avaliar os atributos propostos para caracterização das técnicas de teste de software concorrente e estabelecer pesos a cada atributo durante e o processo de seleção. Os resultados do survey a respeito da relevância de cada atributo e os pesos estabelecidos são descritos na Tabela 2 (MELO, SOUZA, et al., 2017).

O survey foi desenvolvido em três etapas. A primeira etapa focou em reunir informações necessárias para avaliar os conhecimentos dos participantes e atribuir peso em suas respostas através de sua experiência na área. A segunda etapa focou em classificar a importância de cada atributo para o processo de caracterização de técnicas de teste. Os valores admitidos para a avaliação são: Sim (se o atributo for relevante) e Não (caso o atributo não seja relevante). A terceira etapa focou em reunir a relevância de cada atributo em seis faixas como: 1) Nenhuma

relevância, 2) Relevância muito baixa, 3) Baixa relevância, 4) Relevância média, 5) Alta relevância, ou 6) Relevância muito alta (MELO, SOUZA, et al., 2017).

Para condução do survey foram enviados e-mails para 145 pesquisadores e profissionais de teste que têm proposto técnicas de teste de software concorrente, desses 27 responderam ao survey. Considerando a experiência relatadas pelos participantes, eles podem ser classificados como especialistas na área com grande número de publicações na área e participação em projetos na indústria. O resultado considerando o nível de adequação e o nível de relevância, para cada um dos atributos presentes nas técnicas de teste de software concorrente, são apresentados na Tabela 2. Os atributos que estão abaixo de um limite de 65%, para adequação ou relevância foram considerados com peso 0.

Tabela 2: Porcentagem de adequação e nível de relevância para os atributos de caracterização. Fonte: Extraído de (MELO, SOUZA, *et al.*, 2017, p.6).

Categoria	Atributo	Adequação (%)	Relevância (%)	Peso (%)
Modelo de programação	Plataforma de Execução	89	70	6,80
	Paradigma de Desenvolvimento	63	61	0,00
	Linguagem de programação/Bibliotecas de tempo de execução	68	59	5,72
Características gerais de teste	Técnica de teste	96	76	7,38
	Geração de dados de teste	89	69	6,70
	Nível de teste	94	71	6,89
	Mecanismo de intercalação de sincronização	76	66	6,89
	Entradas necessárias	76	66	6,41
	Saídas	81	69	6,70
	Atributos de qualidade	94	80	7,77
	Tipo de estudo empírico	34	37	0,00
Características de teste concorrente	Análise de teste	100	66	6,41
	Paradigma concorrente para a interação do processo	80	71	5,92
	Mecanismo de *replay*	68	62	6,02
	Representação do programa	65	54	0,00
	Instrumentação	69	54	0,00
	Redução do espaço de estados	68	61	5,92
	Erros concorrente	100	80	7,77
Suporte a ferramentas de teste	Existência de ferramenta de apoio	0	0	0,00
	Custo da ferramenta	79	69	6,70
	Plataforma da ferramenta	0	0	0,00

2.4.5. Módulo de apresentação dos resultados

O módulo de apresentação dos resultados da ferramenta *SeleCTT* é responsável por realizar a seleção das técnicas com base nos atributos do projeto de *software* e apresentar uma lista de recomendação das técnicas mais adequadas. Usuários e administradores têm acesso a este módulo, ambos podem utilizar esse módulo para obtenção da lista com as técnicas de teste concorrente mais adequadas a determinado projeto em desenvolvimento. Uma ilustração desse módulo encontra-se no Apêndice A.

Para obter a lista com os resultados é necessário acessar primeiro o módulo de inserção e fornecer informações relevantes ao projeto em desenvolvimento, ou seja, a caracterização do projeto, a fim de compará-las as características das técnicas de teste e obter os resultados referentes a essa comparação.

Após submeter as informações do projeto, o servidor executa o algoritmo de seleção da ferramenta *SeleCTT* por meio dos critérios relevantes ao projeto. Os resultados desse algoritmo para a adequação das técnicas de teste são classificados e ordenados, ranqueando pela maior porcentagem de adequação, das técnicas mais apropriadas, para o projeto em desenvolvimento.

Na Figura 8 é apresentado o código do algoritmo de seleção. A função recebe como parâmetro as informações do projeto em desenvolvimento e ocorre a inicialização dos pesos e das técnicas presentes no corpo de conhecimento (linha 3 a 9). A partir da inicialização, para cada técnica presente (linha 11 a 24) são comparados os atributos da técnica *n* com os do projeto em desenvolvimento (linha 15 a 21). Cada peso individual é calculado comparando os atributos (linha16). O peso total é a soma de todos os pesos dos atributos (linha 19 e 22). Por fim, ordena-se a lista com as técnicas com maior adequação nas primeiras posições (linha 26), retornando o resultado exibindo-o para o usuário (linha 28).

Na Figura 9 é apresentado o processo de seleção de técnicas de teste usando a ferramenta *SeleCTT*. O fluxo de atividades do processo é composto inicialmente pela caracterização do projeto pela equipe de teste. As informações de caracterização são utilizadas no processo de seleção, através de critérios específicos utilizando o corpo de conhecimento para comparação desses critérios. Após a comparação é realizado a classificação das técnicas de acordo com a pontuação alcançada na adequação dos critérios específicos, os resultados são gerados pela ferramenta para a equipe de teste.

```php
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

    protected function resultadoAdequacaoTecnicas ($projeto)
    {
        // Inicialização das variáveis
        $todosPesos       = getPesos();
        $todasTecnicas    = getTodasTecnicas();
        $resultado        = array ();
        $count            = 0;

        foreach ($todasTecnicas as $tecnica) {
            // Inicialização do peso total
            $pesoTotal = floatval(0.000);
            $atributo = 0;
            foreach ($todosPesos['atributos'] as $peso) {
                $resultado[$count][$atributo] = comparacaoAtributoDoProjetoComTecnica($projeto[$atributo],
                                                                                      $tecnica[$atributo],
                                                                                      $peso);
                $pesoTotal += $resultado[$count][$atributo];
                $atributo++;
            }
            $resultado[$count]['resultado_peso'] = $pesoTotal;
            $count++;
        }
        // Ordena Tecnicas mais adequadas nas primeiras posicoes
        $resultado = ordernaResultado ($resultado);

        return $resultado;
    }
?>
```

Figura 8: Fragmento de código do algoritmo de seleção.

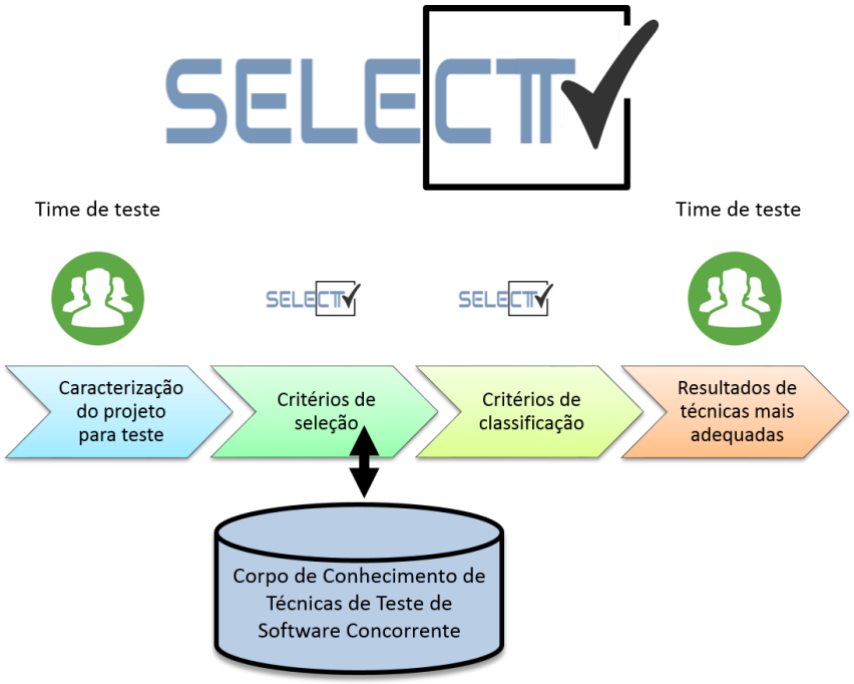

Figura 9: Processo de classificação de seleção das técnicas mais adequadas ao projeto de software. Fonte: Baseado em (DIAS-NETO e TRAVASSOS, 2009).

2.4.5.1 Processo de seleção de técnicas de teste de *software* concorrente

No processo de seleção, as informações sobre um projeto em desenvolvimento são submetidas pelo usuário. Estas informações envolvem os atributos presentes na Tabela 1. O servidor da ferramenta receberá esses atributos na forma de texto por meio do módulo de inserção. No entanto, a partir dessa submissão são comparados os atributos submetido com os atributos presentes nas técnicas do corpo de conhecimento. A comparação é feita considerando os vinte e um atributos, um a um, considerando seus respectivos pesos definidos na Tabela 2. Esta etapa compara os valores presentes nos atributos do projeto em desenvolvimento com as "n" técnicas (sendo n uma das técnicas presentes no corpo de conhecimento) seguindo a relação dada pelo vetor de adequação em (1) onde cada atributo é comparado para cada técnica.

Para o cálculo de adequação considera-se a equação em (1). O valor de adequação $v_{adequação}(i,n)$, sendo i o atributo e n a técnica, é calculado por meio da comparação das características presentes no atributo. Se o atributo possuir mais de uma característica (exemplo: Android e Windows) apenas as características que forem iguais entrarão nesse valor. Nesse caso, comparando o atributo do projeto em desenvolvimento, exemplo Android, com atributo de uma técnica do corpo de conhecimento, exemplo Android e Windows, o resultado será de 50% de adequação para este atributo.

No entanto, a adequação total é dada pela soma do vetor adequação multiplicando-se pelo respectivo peso, essa relação está presente em (1).

$$\text{adequação}_{\text{total}}(n) = \sum_{i=1}^{i=21} v_{\text{adequação}}(i, n) \cdot w_{\text{peso}}(i) \qquad \textbf{(1)}$$

Sendo *i* o índice do atributo e também do peso deste atributo, *i* variando de 1 a 21, sendo *n* o índice da técnica no corpo de conhecimento.

2.5. Exemplos de uso

Esta seção descreve um exemplo prático da aplicação da metodologia proposta e da ferramenta. Relacionando um exemplo de caracterização de atributos para um projeto de teste

e os atributos relacionados à duas possíveis técnicas a serem aplicadas a este projeto como indica a Tabela 3.

Tabela 3: Exemplo de atributos de caracterização das técnicas de teste e do projeto de software.

Categoria	Atributo	Atributos do projeto em desenvolvimento	Técnica 1	Técnica 2
Modelo de programação	Plataforma de Execução	*Windosws*	*Windosws*	*Linux, Windows*
	Paradigma de Desenvolvimento	HPC	HPC	HPC
	Linguagem de programação/Bibliotecas de tempo de execução	Java	Java	Java
Características gerais de teste	Técnica de teste	Teste baseado em modelos	Teste baseado em modelos	Teste baseado em modelos
	Geração de dados de teste	Aleatória	Aleatória	Aleatória
	Nível de teste	Unidade	Sistema	Unidade
	Mecanismo de intercalação de sincronização	Sem informação	Linearizabilidade	Sem informação
	Entradas necessárias	*multithread*	Especificação de classe, *multithread*	*Multithread*
	Saídas	Número de Transições	Número de Transições	Sem informação
	Atributos de qualidade	Eficiência	Eficiência	Eficiência
	Tipo de estudo empírico	Experimento	Experimento	Estudo de Caso
Características de teste concorrente	Análise de teste	Dinâmico	Dinâmico	Dinâmico
	Paradigma de concorrência para a interação do processo	Memória compartilhada	Passagem de mensagem	Memória compartilhada
	Mecanismo de repetição	Exploração sem estado	Exploração sem estado	Agendamento
	Representação do programa	Sem informação	Sem informação	Sem informação
	Instrumentação	Sem informação	Sem informação	CalFuzzer
	Redução do estado do espaço	Sem informação	*Clustering*	Sem informação
	Erros simultâneos	Desencadeamento de erros	Desencadeamento de erros	*deadlock*
Suporte a ferramentas de teste	Nome da ferramenta	*Ballerina*	*Ballerina*	*CalFuzzer*
	Custo da ferramenta	Acadêmico	Acadêmico	Acadêmico
	Plataforma da ferramenta	Sem informação	Sem informação	Sem informação

Tabela 4: Peso estabelecido aos atributos de caracterização para construção do corpo de conhecimento.

Categoria	Atributo	Pesos da Técnica 1	Pesos da Técnica 2
Modelo de programação	Plataforma de Execução	6,80	6,80
	Paradigma de Desenvolvimento	0,00	0,00
	Linguagem de programação/Bibliotecas de tempo de execução	5,72	5,72
Características gerais de teste	Técnica de teste	7,38	7,38
	Geração de dados de teste	6,70	6,70
	Nível de teste	0,00	6,89
	Mecanismo de intercalação de sincronização	0,00	0,00
	Entradas necessárias	6,41	0,00
	Saídas	6,70	6,70
	Atributos de qualidade	7,77	7,77
	Tipo de estudo empírico	0,00	0,00
Características de teste concorrente	Análise de teste	6,41	6,41
	Paradigma de concorrência para a interação do processo	0,00	5,92
	Mecanismo de repetição	6,02	0,00
	Representação do programa	0,00	0,00
	Instrumentação	0,00	0,00
	Redução do estado do espaço	0,00	0,00
	Erros simultâneos	7,77	0,00
Suporte a ferramentas de teste	Nome da ferramenta	0,00	0,00
	Custo da ferramenta	6,7	6,7
	Plataforma da ferramenta	0,00	0,00
	Total de adequação	74,38 %	66,99 %

No entanto, neste exemplo, o resultado de adequação total do projeto em desenvolvimento em comparação com a Técnica 1 e a Técnica 2, resultam em 74,38 % e 66,99% respectivamente. Os cálculos detalhados estão presentes na Tabela 4 onde compara-se um projeto em desenvolvimento com duas técnicas presentes no corpo de conhecimento, portanto conclui-se que a Técnica 1 é mais adequada para este projeto em desenvolvimento dado os atributos de entrada em comparação com o projeto em desenvolvimento.

Os resultados dessa seleção pelo algoritmo serão classificados em: Técnica 1 (primeiro lugar) e Técnica 2 (segundo lugar), sendo retornados para o usuário como indica a Figura 10.

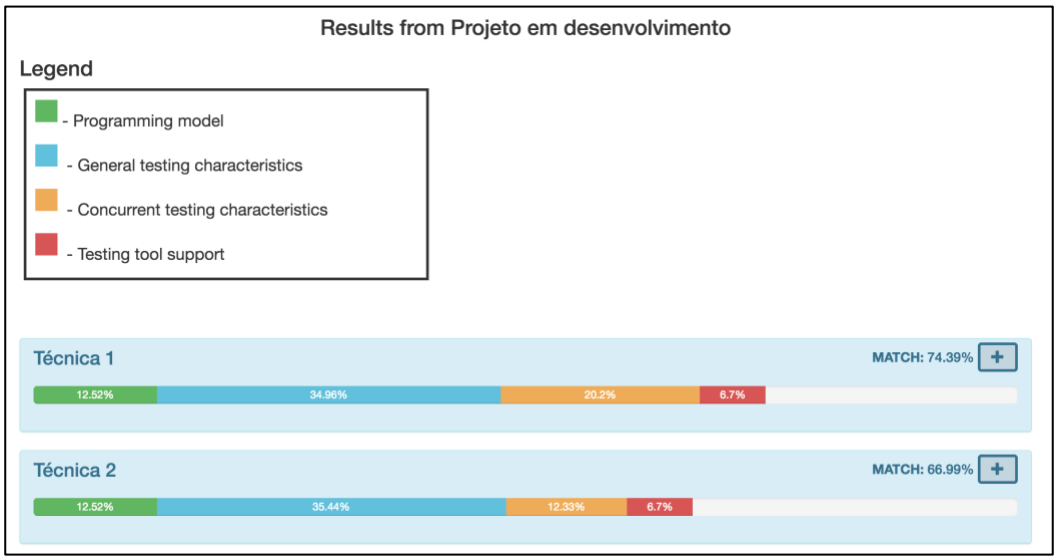

Figura 2: Exemplo de resultado classificado por adequação através do algoritmo de seleção.

Expandindo o resultado para cada uma das técnicas é possível observar a adequação de cada um dos indicados conforme mostra a Figura 11.

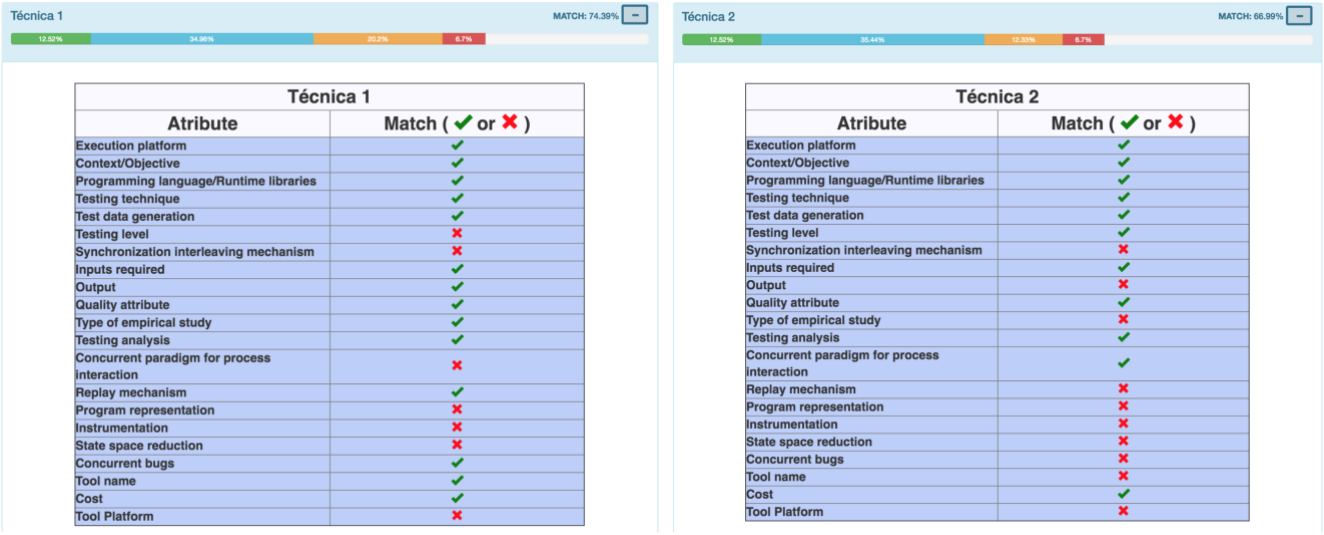

Figura 11: Resultado de adequação de cada atributo para as duas técnicas de exemplo.

2.6. Considerações Finais

Este capítulo descreveu em detalhes o desenvolvimento da ferramenta *SeleCTT*, o modo de funcionamento de seus módulos, os resultados obtidos com este projeto e as dificuldades encontradas durante a implementação dos módulos da ferramenta.

O próximo capítulo tem um enfoque na conclusão e contribuições que se obteve com a ferramenta desenvolvida.

CAPÍTULO 3: CONCLUSÃO

Uma das contribuições essenciais está relacionada ao desenvolvimento de uma infraestrutura *web* que automatiza o processo de seleção, implementando a metodologia proposta e oferecendo uma lista de recomendação de técnicas de teste à equipe de teste.

A classificação de atributos para avaliação e seleção de técnicas de teste de software concorrente. A definição de níveis de adequação e relevância o processo de seleção desses atributos. Tudo isso em conjunto com o projeto de doutorado a que este trabalho está vinculado. Uma visão geral da área de teste de software concorrente, por meio de uma lista de técnicas, ferramentas disponíveis, e análises estatísticas desse domínio.

Além disso, esse projeto ficará disponível na internet, portanto, qualquer pessoa com acesso, e com o mínimo de curiosidade no assunto, de teste de software concorrente, poderá utilizar e beneficiar-se com a ferramenta. A construção de um corpo de conhecimento web que armazena as técnicas de teste de software concorrentes identificadas por meio de uma revisão da literatura técnica. Que contribui para a disponibilização dessa informação aos profissionais de teste tanto na indústria quanto na academia.

O trabalho é desenvolvido em parceria a um projeto de doutorado, e possibilitou automatizar todo o processo proposto, dando maior visibilidade as pesquisas desenvolvidas no Instituto de Ciências Matemáticas e de Computação (ICMC-USP) onde este trabalho foi desenvolvido.

AGRADECIMENTOS

Os autores gostariam de agradecer à FAPESP pelo apoio financeiro sob processo nº 2013/05046-9, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

REFERÊNCIAS

ARORA, V.; BHATIA, R.; SINGH, M. A systematic review of approaches for testing concurrent programs. **Concurrency and Computation: Practice and Experience**, Chichester, UK , v. 28, n. 5, p. 1572-1611, 10 April 2016. ISSN 1532-0634.

BARNEY, B. 1. **Introduction to Parallel Computing**, 2017. Disponivel em: <https://computing.llnl.gov/tutorials/parallel_comp/>. Acesso em: 23 Setembro 2017.

CARVER, R. H.; TAI, K.-C. **Modern Multithreading:** Implementing, Testing, and Debugging Multithreaded Java and C++/Pthreads/Win32 Programs. 1. ed. [S.l.]: Wiley, v. 1, 2005. ISBN 978-0-471-72504-6.

DIAS-NETO, A. C.; TRAVASSOS, G. H. Model-based testing approaches selection for software projects. **Journal of Information and Software Technology**, Butterworth-Heinemann Newton, MA, USA, v. 51 Issue 11, p. 1487-1504, November 2009. ISSN doi>10.1016/j.infsof.2009.06.010.

GRAMA, A. et al. **Introduction to Parallel Computing**. Second Edition. ed. [S.l.]: Pearson, 2003.

KITCHENHAM, B. et al. Systematic literature reviews in software engineering – A systematic literature review. **Information and Software Technology**, v. 51, n. 1, p. 7 - 15, January 2009. ISSN 0950-5849.

MELO, S. M. et al. **How to Test your Concurrent Software:** An Approach for the Selection of Testing Techniques. [S.l.]: [s.n.]. 2017.

MYERS, G. J. **The Art of Software Testing**. 2ª. ed. New York: John Wiley & Sons, Inc., 1979.

SOUZA, S. R. S. et al. **Research in concurrent software testing:** a systematic review. PADTAD '11 Proceedings of the Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging. Toronto, Ontario, Canada: ACM. 2017. p. 1-5.

VEGAS, S.; BASILI , V. A Characterization Schema for Software Testing Techniques. **Empirical Software Engineering**, Hingham, MA, USA, v. 10, n. 4, p. 437-466, October 2005. ISSN doi>10.1007/s10664-005-3862-1.

APÊNDICE A – Visão geral dos módulos

1) Tela de boas-vindas da *SeleCTT*.

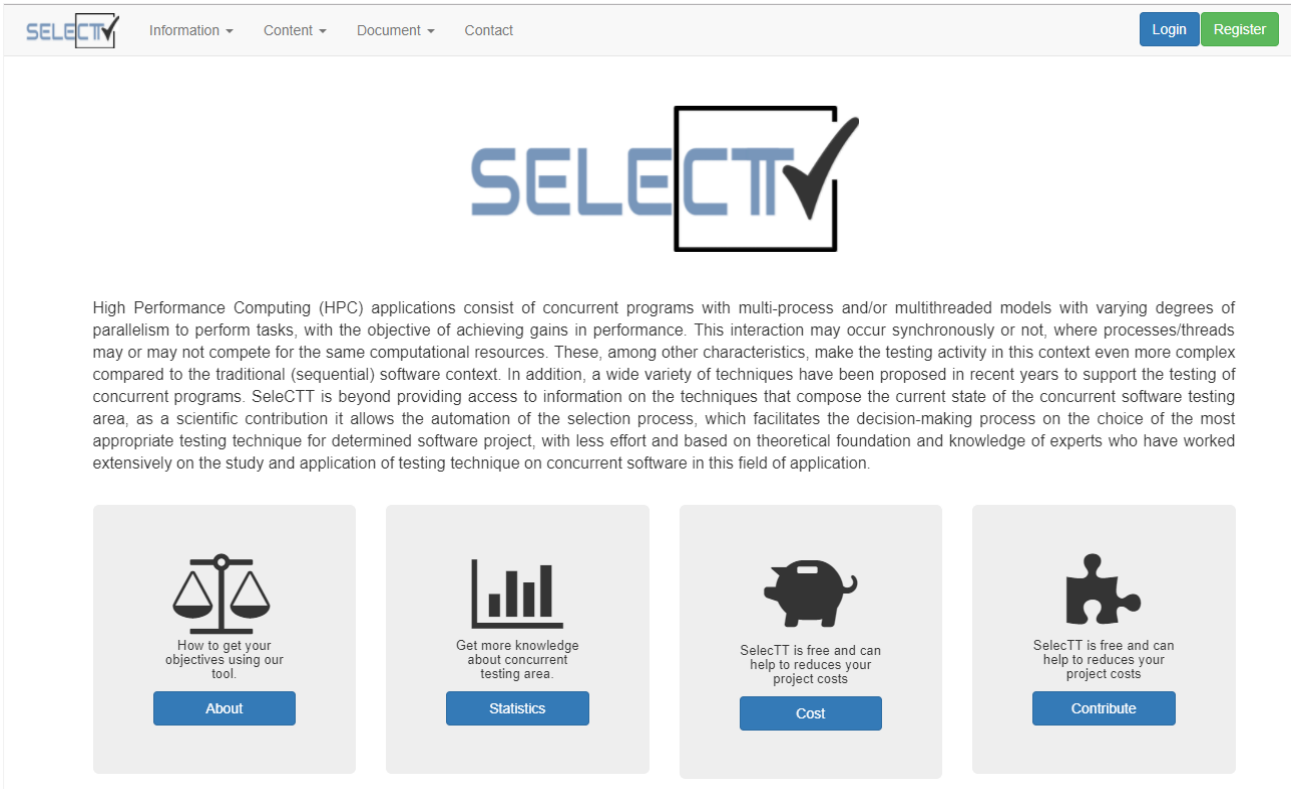

Figura 12: Tela de boas-vindas da SelleCT.

2) Módulo de inserção da *SeleCTT*.

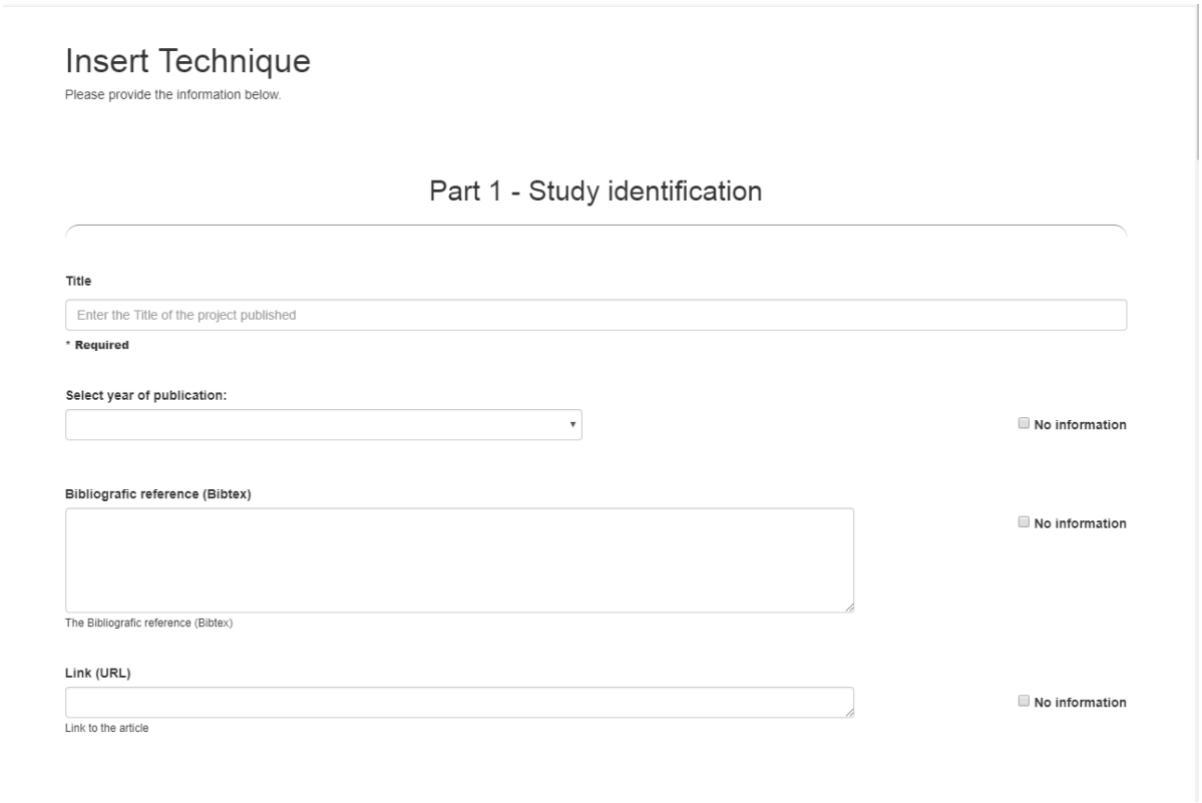

Figura 13: Interface do módulo de inserção.

3) Módulo de Administração do Sistema da *SeleCTT*.

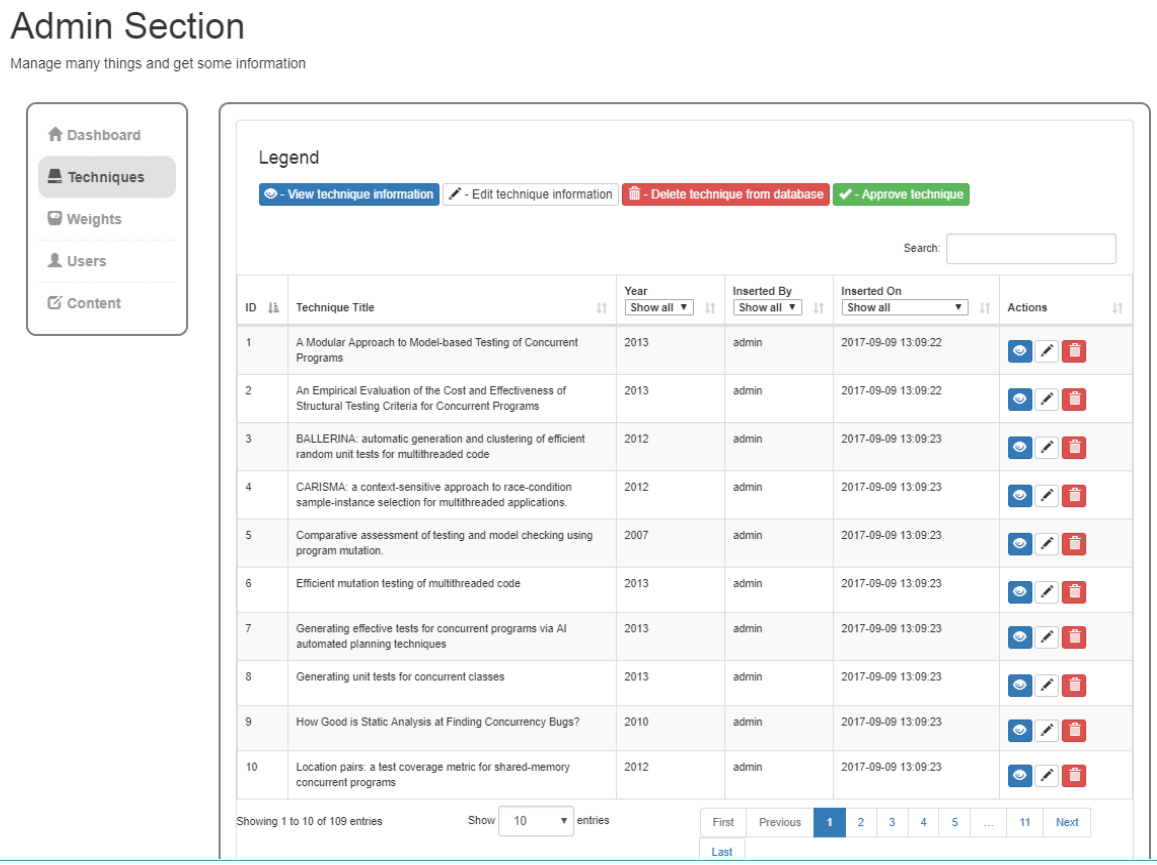

Figura 14: Interface do módulo de Administração de Sistema. Exemplo de gerenciamento de técnicas presentes no corpo de conhecimento.

4) Módulo de Controle da *SeleCTT.*

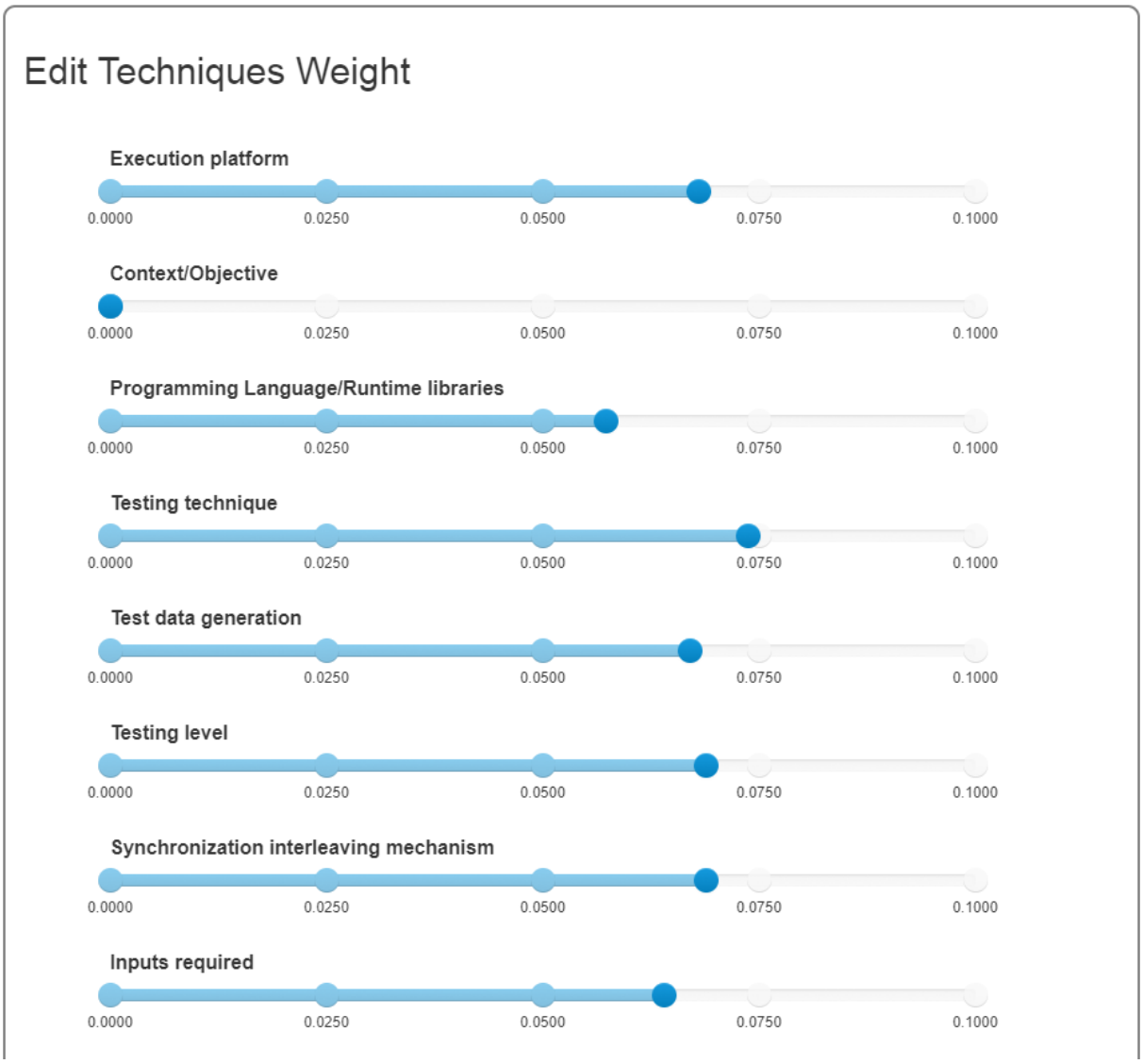

Figura 15: Interface do módulo de Controle. Exemplo dos pesos de alguns atributos.

5) Módulo de Apresentação dos Resultados da *SeleCTT*.

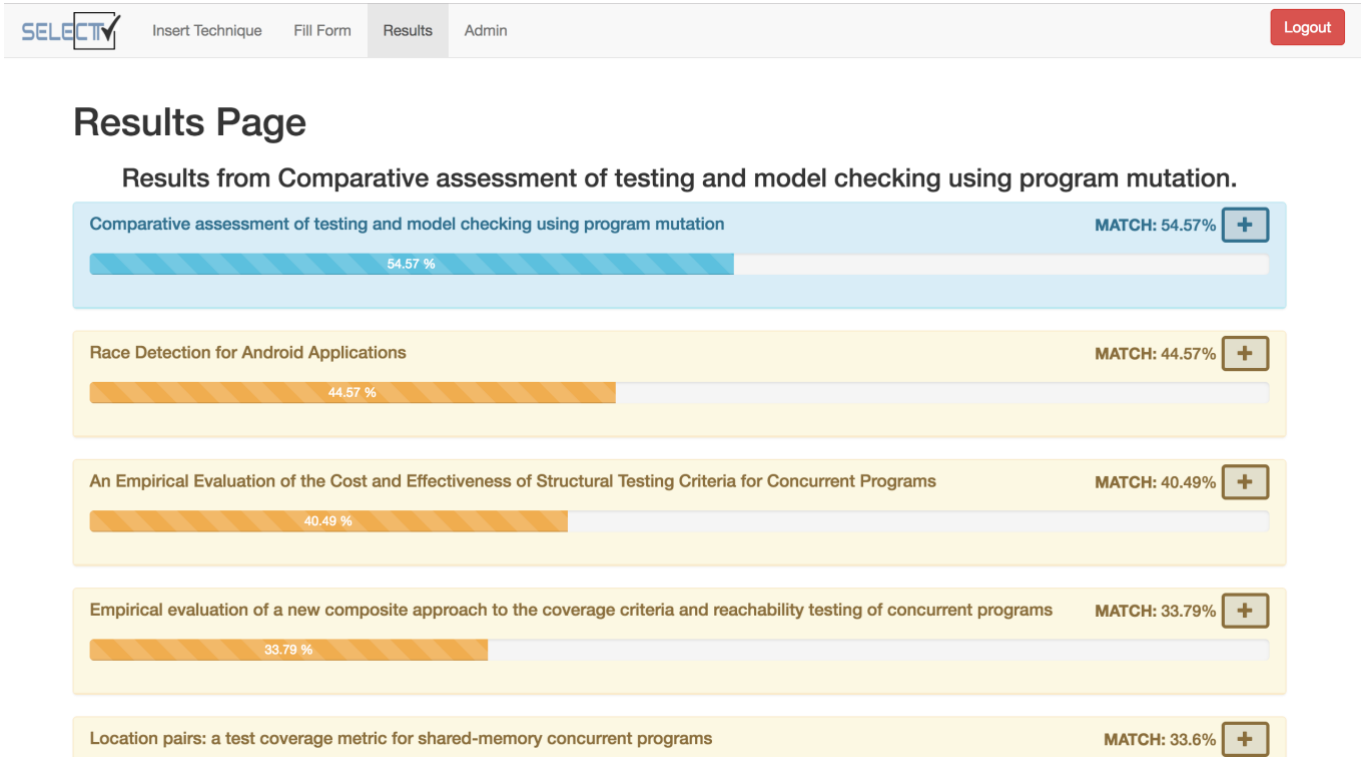

Figura 16: Interface do módulo de Apresentação dos Resultados. Exemplo de recomendação de técnicas para um projeto de *software*.