Lucas Cabral Soares e Maria Eduarda Amaral Muniz
{ maria.amaral, lucas.cabral } @sga.pucminas.br

Documento de Visão para o Sistema de GraphTest

21 de Setembro de 2025

Proposta dos alunos Lucas Cabral Soares e Maria Eduarda Amaral Muniz ao curso de Engenharia de Software como projeto de Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo dos professores Danilo de Quadros Maia Filho, Leonardo Vilela Cardoso, Raphael Ramos Dias Costa e orientação acadêmica do professor Cleiton Silva Tavares.

OBJETIVOS

O presente documento tem como objetivo descrever a visão do sistema GraphTest, que será desenvolvido como parte do Trabalho de Conclusão de Curso. A plataforma busca apoiar engenheiros de software, pesquisadores e estudantes na aplicação de testes estruturais (caixa branca) e funcionais (caixa preta), fornecendo meios automatizados e interativos para geração de Grafos de Fluxo de Controle (GFC) e Grafos de Causa-Efeito (GCE).

O sistema visa atender às necessidades de reduzir a complexidade da geração manual de grafos de teste, auxiliar na identificação de casos de teste mais eficientes para diferentes critérios de cobertura (comandos, decisões, condições e caminhos), apoiar a derivação de tabelas de decisão e casos de teste funcionais a partir de grafos de causa-efeito e fornecer relatórios visuais e métricas que facilitem a análise da qualidade dos testes.

ESCOPO

A plataforma oferecerá recursos que auxiliam o usuário na realização tanto de testes estruturais quanto funcionais em um ambiente único e integrado. No contexto de testes estruturais, será possível importar código-fonte em linguagem Java, para que a aplicação gere automaticamente o GFC. A partir desse grafo, o sistema calculará complexidade ciclomática e fornecerá o esqueleto de teste estrutural. Já no contexto de testes funcionais, a plataforma permitirá que o usuário modele graficamente as causas e efeitos extraídos da especificação, resultando em um

GCE. O usuário pode converter o grafo em uma tabela de decisão e, posteriormente, gerar o esqueleto de teste funcional. Como diferencial em relação a sistemas concorrentes, que geralmente tratam apenas de cobertura de código (por exemplo, *JaCoCo* ou Cobertura) ou apenas de geração de tabelas de decisão (como *Tcases*), esta plataforma unificará as duas abordagens em uma solução acadêmica e acessível, favorecendo tanto o ensino quanto a prática profissional.

FORA DO ESCOPO

A plataforma não terá como objetivo executar testes automatizados diretamente em frameworks como *JUnit*, *PyTest* ou *Selenium*, visto que sua finalidade principal consiste na geração e análise de casos de teste, e não em sua execução.

Além disso, não será oferecido suporte a outras linguagens de programação além de *Java*, uma vez que esta foi escolhida por representar a linguagem de maior domínio e familiaridade da equipe de desenvolvimento.

Por fim, não haverá integração direta com ambientes de desenvolvimento integrados (IDEs) como *Eclipse* ou *IntelliJ IDE*, considerando que a interação do usuário com o sistema ocorrerá exclusivamente por meio da interface web da plataforma.

GESTORES, USUÁRIOS E OUTROS INTERESSADOS

Nome	Cleiton Silva Tavares
Qualificação	Gestor/Orientador
Responsabilidades	Auxiliar no acompanhamento do desenvolvimento da plataforma e fornecer orientações acadêmicas.

Nome	Danilo de Quadros Maia Filho
Qualificação	Gestor/Orientador
Responsabilidades	Auxiliar no acompanhamento do desenvolvimento da plataforma e fornecer orientações acadêmicas.

Nome	Leonardo Vilela Cardoso
Qualificação	Gestor/Orientador
Responsabilidades	Auxiliar no acompanhamento do desenvolvimento da plataforma e fornecer orientações acadêmicas.

Nome	Raphael Ramos Dias Costa
Qualificação	Gestor/Orientador
Responsabilidades	Auxiliar no acompanhamento do desenvolvimento da plataforma e fornecer orientações acadêmicas.

Nome	Lucas Cabral Soares
Qualificação	Estudante/Desenvolvedor
Responsabilidades	Responsável pelo desenvolvimento da plataforma ao longo da disciplina de TCC I.

Nome	Maria Eduarda Amaral Muniz
Qualificação	Estudante/Desenvolvedor
Responsabilidades	Responsável pelo desenvolvimento da plataforma ao longo da disciplina de TCC I.

Qualificação	Estudante de Engenharia de Software
Responsabilidades	Participa de disciplinas relacionadas a teste de software, aprendendo fundamentos de teste estrutural e funcional. Utiliza a plataforma como ferramenta de apoio ao aprendizado, explorando exemplos práticos.
Como poderá usar o sistema?	Gerando GFCs a partir de pequenos trechos de código fornecidos em aula e construindo GCEs a partir de especificações, a fim de compreender a derivação de casos de teste.

Qualificação	Professor Universitário de Teste de Software
Responsabilidades	Leciona disciplinas de qualidade e teste de software, criando exercícios, exemplos e avaliações. Precisa de ferramentas didáticas para ilustrar conceitos de grafos de fluxo de controle e grafos de causa-efeito.
Como poderá usar o sistema?	Utilizando a plataforma para gerar GFCs a partir de trechos de código de exemplo e construir GCEs a partir de especificações, de modo a ilustrar em sala de aula como casos de teste podem ser derivados.

Qualificação	Engenheiro de Software Profissional
Responsabilidades	Responsável por validar qualidade de código e especificações em projetos reais. Atua no planejamento e execução de testes em nível de unidade, integração e sistema.
Como poderá usar o sistema?	Importando código do projeto para gerar automaticamente o GFC, avaliando a cobertura dos testes já implementados, e utilizando o módulo funcional para derivar casos de teste adicionais a partir de requisitos.

LEVANTAMENTO DE NECESSIDADES

1. Automatização da geração de GFCs. Atualmente, professores e engenheiros realizam esse processo de forma manual, o que demanda tempo e aumenta as chances de falhas. Ao automatizar a construção desses grafos, o sistema reduz o esforço e garante maior precisão.

2. Derivação de casos de teste funcionais a partir de especificações. A geração manual de tabelas de decisão é suscetível a erros e pode não contemplar todas as combinações necessárias. O uso de GCEs e a derivação automática de tabelas proporcionam maior confiabilidade no processo.

3. Consolidação das métricas de cobertura estrutural e funcional em um único ambiente. Hoje, essas análises são feitas em ferramentas diferentes, dificultando a visão integrada

dos resultados. Com a plataforma, será possível avaliar de forma unificada a cobertura de código e os casos de teste funcionais.

4. Visualização interativa e didática das estruturas de teste. A ausência de ferramentas que ilustram de maneira clara os GFCs e GCEs. A plataforma permitirá que esses elementos sejam explorados de forma visual e compreensível.

FUNCIONALIDADES DO PRODUTO

Necessidade: Automatização da geração de grafos de fluxo de controle	
Funcionalidade	**Categoria**
1. Importar código-fonte e realizar parsing automático retornando a lista de métodos encontrados para que o usuário possa escolher exatamente a partir de qual método o grafo será gerado.	Crítico
2. Gerar GFC a partir de um método pré-selecionado	Crítico
3. Identificar blocos de comandos e decisões, criando nós e arestas do GFC	Crítico
4. Calcular automaticamente a complexidade ciclomática do método a partir do GFC gerado, registrando o valor associado ao grafo e ao método correspondente.	Crítico
5. Gerar esqueleto do método de teste, anotado com @Test, contendo asserções vazias organizadas pelos caminhos independentes do GFC, permitindo que o usuário complete os valores necessários para cobrir cada caminho.	Útil

Necessidade: Derivação de casos de teste funcionais a partir de especificações	
Funcionalidade	**Categoria**
1. Modelar causas e efeitos por meio de interface gráfica dedicada	Crítico
2. Suportar operadores lógicos (AND/OR/NOT) e restrições (E, I, O, R, M) no grafo	Crítico
3. Converter o grafo de causa-efeito em tabela de decisão	Crítico

4. Validar consistência do modelo (causas sem uso, efeitos inalcançáveis)	Útil
5. gerar esqueleto do método de teste, já com a anotação @Test e com asserções vazias referentes aos efeitos identificados no GCE, deixando para o usuário apenas preencher os valores concretos.	Útil

Necessidade: Visualização interativa e didática das estruturas de teste	
Funcionalidade	**Categoria**
1. Fornecer layouts automáticos de grafos	Importante
2. Adicionar anotações/legendas automáticas (rótulos de nós, decisões, caminhos)	Importante

INTERLIGAÇÃO COM OUTROS SISTEMAS

O sistema estabelecerá uma integração interna entre o parser de código Java, o mecanismo de derivação de grafos e o módulo de visualização. O parser AST extrairá a estrutura sintática do código, que servirá como insumo direto para os algoritmos de construção do GFC. Em seguida, bibliotecas de renderização gráfica serão utilizadas para exibir o grafo de forma interativa, permitindo inspeção, navegação e refinamento sem necessidade de exportação externa.

RESTRIÇÕES

Restrições de produto

- **Requisitos de eficiência:**
    - A geração do GFC para trechos de até 500 linhas deve ocorrer em até 2 segundos em ambiente de referência.
    - A construção do GCE com até 20 causas/efeitos deve ocorrer em até 2 segundos em ambiente de referência.
- **Requisitos de confiabilidade:**
    - O cálculo de complexidade ciclomática do GFC deve ser apresentada com precisão de duas casas decimais.
    - A geração dos grafos deve ser determinística: a mesma entrada resulta no mesmo grafo e mesma complexidade ciclomática.
- **Requisitos de segurança e privacidade:**

    - O arquivo *.java* importado pelo usuário é processado apenas em memória e descartado por padrão; qualquer forma de persistência depende de consentimento explícito.
    - *Logs* não devem armazenar conteúdo do código submetido; apenas metadados técnicos (timestamp, tamanho aproximado, rótulos de artefato).
- **Requisitos de usabilidade e acessibilidade:**
    - Interface responsiva e com contraste adequado, suporte a navegação por teclado e textos em PT-BR.

Restrições organizacionais

- **Requisitos de implementação:**
    - O sistema será desenvolvido em *TypeScript (Angular)* no *front-end* e *Java* (*Spring Boot*) no *back-end*.
    - A autenticação na plataforma será implementado com *Spring Security*
    - A análise de código-fonte Java utilizará a biblioteca *JavaParser* para extração da *AST* (*Abstract Syntax Tree*) e geração do GFC.
    - A renderização e manipulação visual dos grafos será feita com *Cytoscape.js*, uma biblioteca open source licenciada sob *MIT License*.
    - Devem ser adotadas bibliotecas open source com licenças compatíveis (*MIT, Apache-2.0* ou equivalentes) para visualização de grafos e exportação de relatórios.

Restrições externas

- **Requisitos de portabilidade:**
    - Execução em navegadores modernos (*Chrome/Edge/Firefox* versões recentes). Não há dependência de IDEs.
- **Requisitos de interoperabilidade:**
    - Suporte a importação local de arquivos *.java*.
    - Os arquivos exportados poderão ser utilizados em ferramentas externas de análise, visualização ou documentação.
- **Requisitos legais/licenciamento:**
    - Observância às licenças de terceiros e proibição de armazenamento de código confidencial sem autorização do usuário.

DOCUMENTAÇÃO

A documentação do projeto incluirá um manual do usuário, explicando de forma detalhada o funcionamento das principais funcionalidades, bem como um guia de instalação para deploy

local ou em nuvem. Também será elaborado um arquivo README destinado a desenvolvedores, descrevendo requisitos técnicos e instruções para evolução da plataforma. Além desses materiais, a documentação incorporará todos os diagramas produzidos ao longo do desenvolvimento — diagramas de sequência, diagramas de estados, diagramas de componentes, diagramas de arquitetura e o diagrama de classes — garantindo a representação completa dos fluxos internos, dos modelos de dados e da dinâmica operacional do sistema. Essa documentação permitirá que usuários finais tenham clareza sobre a utilização e a evolução do sistema.