Plano de Testes

Informações sobre o documento

Sumário do documento

Chess	*Nome*
Nome do componente	*Plano de Testes*
Autores do documento	*Beatriz Loza, Igor Martire*
Data	*01/05/2017*
Número de páginas	*8*

Sumário de alterações do Documento

Versão	Data	Descrição	Responsável
0.1	*01/05/2017*	*Primeiro esboço do documento criado*	*Igor Martire*
0.2	*DD/MM/AAAA*	*<Documento revisado e editado>*	*Nome*
1.0	*DD/MM/AAAA*	*<Aprovado>*	*Nome*

1 Introdução

O objetivo deste plano de testes é detalhar a metodologia que será utilizada para a realização dos testes unitários, de integração e de aceitação durante o desenvolvimento do projeto. Além disso, objetiva-se também a definição dos responsáveis pela criação e execução dos mesmos.

1.1 Sumário de alto nível

O escopo dos testes descritos neste documento engloba todas as funcionalidades do software desenvolvido, a fim de garantir a qualidade do mesmo no momento de sua entrega.

Contudo, diferentes tipos de testes serão desenvolvidos para cada componente do projeto. Em geral, testes unitários e de integração serão criados para cada funcionalidade e suas combinações. Contudo, quando o todo ou a parte envolver elementos de interface de usuário, o tipo de teste utilizado passará a ser o de aceitação, através do uso do sistema pelo Gerente de Qualidade e pelo cliente.

Os testes unitários e de integração não visam a serem exaustivos, mas procuram validar o funcionamento integral das diferentes partes do projeto. Mediante problemas descobertos durante o uso do sistema, os mesmos passarão a fazer parte das baterias de testes especificados neste documento.

Testes unitários fazem parte das tarefas de desenvolvimento de cada funcionalidade e, assim, cada tarefa de desenvolvimento só é considerada pronta mediante a entrega conjunta de seus testes unitários, desde que dentro do escopo já definido. Testes de integração, por sua vez, serão realizados previamente a cada entrega de software. Por fim, os testes de aceitação serão executados durante o último estágio previsto de entrega.

Nota-se, por fim, que este documento trata-se de um documento vivo, que deverá ser atualizado conforme novos casos de teste forem surgindo ou modificações no processo sejam necessárias.

1.2 Terminologia

Travis CI é uma ferramenta de integração contínua que permite executar verificações automaticamente quando o código é alterado no repositório compartilhado da equipe.

Pytest é uma ferramenta e um framework que permite a criação e execução de testes unitários e de integração em Python.

Testes unitários referem-se a testes de funcionalidades isoladas, enquanto que testes de integração referem-se a testes de funcionalidades de forma que interajam umas com as outras.

2 Ambiente e Configuração dos Testes

Nesta sessão são descritos o ambiente de testes e as configurações necessárias para a execução deles.

2.1 Plataformas

Os testes devem ser executados em todas as plataformas suportadas pelo software. Assim, os testes devem ser realizados tanto em plataformas Windows quanto em plataformas Mac.

2.2 Configurações

O sistema deve possuir a versão 3.6 do Python instalada, bem como o pacote PyGame, que é uma dependência do projeto desenvolvido. Além disso, a ferramenta Pytest deve estar instalada para que a execução dos testes possa ser realizada.

3 Análise e Estratégia de Teste

A análise e a estratégia de teste são descritas minuciosamente nesta sessão, que abrange os objetivos, as dependências, a preparação e finalmente os casos de teste.

3.1 Objetivos dos testes

3.1.1 Objetivo 1 - Facilidade de Uso

- **Condição do teste**: Após iniciada a aplicação, o usuário leigo deve ser capaz de iniciar uma partida de xadrez e jogá-la sem a necessidade de ler uma documentação de utilização do software.
- **Abordagem**: Avaliar o uso feito por usuários leigos nos momentos de apresentação do software.
- **Critério de aceitação**: O usuário leigo sentiu facilidade no uso do software.

3.1.2 Objetivo 2 - Jogador computador inteligente

- **Condição do teste**: Um jogador iniciante deve apresentar dificuldade em vencer do computador.
- **Abordagem**: Avaliar o uso feito por jogadores iniciantes nos momentos de apresentação do software.
- **Critério de aceitação**: O jogador iniciante sentiu dificuldade em vencer o computador.

3.1.3 Objetivo 3 - Uso do software em plataformas Windows e Mac

- **Condição do teste**: O jogo deve ser utilizável por usuários que utilizam plataformas Windows e Mac.
- **Abordagem**: Tentar executar o jogo em plataformas Windows e Mac.
- **Critério de aceitação**: O jogo deve executar corretamente em plataformas Windows e Mac.

3.2 Dependências dos casos de teste

Para os casos de teste unitários e de integração, é necessário que o sistema tenha a ferramenta Pytest instalada, além do Python versão 3.6. Para os casos de teste de aceitação, basta que a aplicação esteja devidamente instalada e pronta para ser executada.

Além disso, é necessária a presença de um sistema de janelas para os testes de aceitação, já que fazem uso da parte gráfica, e também a presença de um *mouse* ou dispositivo de entrada semelhante para fornecer a entrada necessária aos testes.

3.3 Preparação para os casos de teste

Antes da execucação dos testes é necessário estar no diretório raiz do projeto tanto para execução dos testes automatizados quanto para execução dos testes de aceitação.

3.4 Casos de teste

3.4.1 CT 1 - Testes automatizados

- **Descrição**: Verificar as funcionalidades dos sistemas de forma isolada e integrada através de testes que usam o modelo de caixa-branca. Os testes são executados através do comando "python -m pytest tests/" em plataformas Windows e Mac.

- **Tempo estimado**: 3 segundos.
- **Critério de saída**: A saída esperada é uma mensagem de que todos os testes passaram.
- **Requisitos do ambiente**: O pacote "pytest" deve estar instalado.
- **Objetivos mapeados**: Objetivo 3.

3.4.2 CT 2 - Teste de movimentação válida

- **Descrição**: Após inicializar o jogo, verificar que a movimentação das peças funciona conforme as regras do jogo de xadrez.
- **Tempo estimado**: 10 segundos.
- **Critério de saída**: A tela do jogo deve mostrar o movimento da peça conforme a entrada do usuário.
- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivos 1 e 3.

3.4.3 CT 3 - Teste de movimentação inválida

- **Descrição**: Verificar que a movimentação das peças não funciona quando o movimento pedido é inválido.
- **Tempo estimado**: 10 segundos.
- **Critério de saída**: A tela do jogo deve mostrar que a peça não foi movida como desejado.
- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivos 1 e 3.

3.4.4 CT 4 - Teste de indicação de movimentação

- **Descrição**: Verificar que ao selecionar uma peça para movimentação, as casas para as quais a peça pode ser movida são coloridas de verde caso estejam livres e de vermelho caso sejam ocupadas por uma peça adversária.
- **Tempo estimado**: 20 segundos.
- **Critério de saída**: A tela do jogo deve mostrar a coloração do tabuleiro conforme especificada.
- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivos 1 e 3.

3.4.5 CT 5 - Teste de notificações

- **Descrição**: Verificar que notificações são dadas ao usuário quando há xeque e xeque-mate.
- **Tempo estimado**: 30 segundos.
- **Critério de saída**: Uma notificação deve aparecer informando ao usuário o estado de xeque ou xeque-mate.
- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivos 1 e 3.

3.4.6 CT 6 - Teste de movimentação inválida quando em xeque

- **Descrição**: Verificar que ao movimentos que não eliminam a condição de xeque são considerados inválidos.
- **Tempo estimado**: 30 segundos.
- **Critério de saída**: O movimento não deve ser aceito e uma notificação deve relembrar que o jogador está em xeque.
- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivos 1 e 3.

3.4.7 CT 7 - Teste de promoção de peão

- **Descrição**: Verificar que ao chegar com o peão na última linha possível, são dadas opções de seleção de peças conforme as regras.
- **Tempo estimado**: 30 segundos.
- **Critério de saída**: O peão deve ser substituído com a peça escolhida.
- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivos 1 e 3.

3.4.8 CT 8 - Teste de movimento especial: Roque

- **Descrição**: Verificar que o movimento de Roque é considerado válido e permitido.
- **Tempo estimado**: 20 segundos.
- **Critério de saída**: O movimento de Roque deve provocar uma coloração diferente no tabuleiro para indicar a sua possibilidade quando o Rei é selecionado e o estado do tabuleiro permite tal movimento. Além disso, as peças envolvidas (rei e torre) devem ficar em suas posições esperadas após o movimento ser realizado.

- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivos 1 e 3.

3.4.9 CT 9 - Teste de movimento especial: Ao passar

- **Descrição**: Verificar que o movimento de Ao Passar é considerado válido e permitido.
- **Tempo estimado**: 20 segundos.
- **Critério de saída**: O movimento de Ao Passar deve provocar uma coloração diferente no tabuleiro para indicar a sua possibilidade quando o peão é selecionado e o estado do tabuleiro permite tal movimento, conforme as regras.
- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivos 1 e 3.

3.4.10 CT 10 - Teste de inteligência do computador

- **Descrição**: Verificar que o computador apresenta movimentos inteligentes, promovendo, assim, uma partida desafiante para jogadores iniciantes.
- **Tempo estimado**: 5 minutos.
- **Critério de saída**: Xeques e, possivelmente, xeque-mate do computador contra jogadores iniciantes.
- **Requisitos do ambiente**: O software deve estar instalado.
- **Objetivos mapeados**: Objetivo 2.

4 Execução dos Testes

4.1 Cobertura

O teste cobrirá todas as funcionalidades do software, além da usabilidade do mesmo e qualidade da inteligência artificial.

4.2 Resultados

O único caso de teste executado com sucesso até momento da última edição deste documento é o Caso de Teste 1.

5 Lições aprendidas

É importante que se tenha, desde o início e por toda a equipe, o entendimento da importância de se fazer os testes automatizados junto à tarefa de desenvolvimento de cada funcionalidade. Na mesma linha, é importante que o processo de testes já esteja bem definido desde o início e que ferramentas de integração contínua auxiliem na tarefa de garantir que nenhum teste está sendo quebrado após alguma modificação na base de código.

Bibliografia

Foundations of Software Testing: ISTQB Certification – Chapter 4: Test Design Techniques, Dorothy Graham et al.

Test Plan Outline (IEEE 829 Format) – Foundation Course in Software Testing - Prepared by Systeme Evolutif Limited, accessed in August, 2012 at http://www.computing.dcu.ie/~davids/courses/CA267/ieee829mtp.pdf

Boris Beizer - Black-Box Testing - Techniques for functional testing of software and systems .

William E. Perry - Effective Methods for Software Testing

https://www.ibm.com/developerworks/br/local/rational/criacao_geracao_planos_testes_software/