Hangman Testing

Departamento de Produtos - Time de Tecnologia e Desenvolvimento - Política(s) de Teste e Prática(s) de Teste Organizacional.

Autores: Pedro Vinícius (201907494), Sávio Matheus de Sousa (201907505)

Política(s) de Teste

Objetivos de testagem: a testagem tem como objetivo garantir a robustez e confiabilidade das aplicações, garantindo principalmente uma boa experiência de uso aos seus usuários.

Processo de teste: os testes são conduzidos de acordo com a ISO/IEC/IEEE 29119-2, seguindo variações que atendam às regras de negócio e alinhados às práticas ágeis. O processo de teste consiste no entendimento do caso de uso, análise de riscos, elaboração de um plano de teste, execução e registro dos resultados. Ele deve ser realizado de forma iterativa, conforme novas partes do software são testadas.

Organização dos testes: os testadores são alocados a projetos. Os testadores relatam individualmente seus resultados a outros membros da equipe e a seu gerente, a fim de avaliar consistências e inconsistências frente a diferentes técnicas. Os testadores possuem equipamento e documentação necessária para realização destes.

Treinamento: a rigor, para todos os testadores é necessário possuir diploma de conclusão do ensino superior na área de tecnologia da informação, entre eles os cursos de Sistemas de Informação, Engenharia de Software, Ciências da Computação ou correlatos, e certificação no setor de Teste de Software. Ademais, faz-se necessário que o testador obtenha pela empresa a certificação de teste de software que inclui treinamento de nível básico, intermediário e avançado baseado em seu nível de conhecimento.

Código de ética: todos os testadores devem aderir ao código de ética da companhia.

Padrões: a companhia segue os padrões estabelecidos pela ISO/IEC 20246 (quando se trata de testes estáticos), ISO/IEC/IEEE 29119-2, ISO/IEC/IEEE 29119-3 e ISO/IEC/IEEE 29119-4.

Políticas relevantes relacionadas: o ciclo de testagem de software é primariamente pautado nos padrões acima definidos com o adendo de seguirem as boas práticas de teste de software disponíveis no repositório de documentos, acessível pelos colaboradores através da rede privada da companhia.

Aprimoramento do processo de teste e determinação de valor: obrigatoriamente, em todos os projetos, deve-se prover relatórios com dados coletados em diversos pontos do ciclo de testes a fim de aprimorar o processo, boas práticas e normas seguidas pelos testadores.

Assim, ao ter embasamento necessário, são estudados e estabelecidos pontos de melhoria nos pontos identificados.

Armazenamento e reuso de artefatos de teste: obrigatoriamente, deve-se manter registro de todos os processos, boas práticas, normas e artefatos de teste à disposição, com a companhia provendo repositórios em sua rede privada com os devidos cuidados em relação à segurança da informação.

Publicado por: Pedro Vinícius (Chefe de Testes), Sávio Matheus de Sousa (Líder de Testes)

Aprovado por: Oscar Nogueira (Chief Technology Officer - CTO)

Prática(s) de Teste Organizacional

Escopo: As Práticas de Teste Organizacionais são aplicáveis a todos os testes realizados na Hangman Testing.

Gerenciamento de risco: as equipes ágeis devem avaliar o risco do produto utilizando o Processo de Gerenciamento de Risco Relacionado a Testes (TRM56), as quais serão conduzidas durante a iteração zero e revisitadas durante as reuniões de planejamento de iteração subsequentes.

Seleção e priorização de teste: No planejamento de cada iteração, será realizado uma seleção e priorização dos testes a serem feitos baseando-se na avaliação de risco da iteração.

Documentação e relatórios de teste: antes de cada iteração, cada equipe deverá produzir um plano de teste, o qual contemplará quais funcionalidades serão testadas e as técnicas e os critérios a serem utilizados. Cada testador deverá documentar os casos de teste a serem realizados. Ao final da iteração, a equipe deverá produzir um relatório que irá conter os casos de testes aprovados, reprovados e bloqueados e o defeitos encontrados, classificados conforme sua gravidade.

Automação e ferramentas de teste: sempre que for possível, os testes serão automatizados por meio de scripts e de ferramentas, dentre as quais pode-se listar ferramentas de gerenciamento de teste, ferramentas de execução de teste, ferramentas de teste de desempenho e ferramentas de teste de usabilidade. A nível de sistema, o teste deverá ser feito utilizando a ferramenta Cucumber.

Gerenciamento de configuração de ativos de teste: as ferramentas de gerenciamento de requisitos, gerenciamento de teste e automação de teste da organização possuem gerenciamento de configuração integrado para todos os ativos armazenados em cada

ferramenta. Todo ativo baseado em documento será armazenado na pasta da equipe da ferramenta de armazenamento em nuvem utilizada pela organização.

Gerenciamento de incidentes: incidentes devem ser registrados no sistema de gerenciamento de tarefas/requisitos vinculados a histórias de usuário e classificados conforme a gravidade (alta, média ou baixa).

Tipos de teste: geralmente, se aplica testes funcionais, de desempenho, de acessibilidade, de penetração e de desastre/recuperação.

Níveis de teste: serão feitos testes no nível de unidade, de integração, de sistema, de aceitação, de verificação de produção e de regressão.

Regras/diretrizes para desvio das práticas de teste organizacionais: toda equipe deverá utilizar os mesmos modelos de gerenciamento de requisitos/histórias, baseados em Gherkin, gerenciamento de defeitos e gerenciamento de testes pertencentes ao conjunto de ferramentas da organização. Os modelos de documentação de teste são adaptáveis, conforme o que for exigido por cada equipe de entrega.

Critérios de entrada: Os critérios de entrada para execução do processo de teste em cada iteração podem incluir: plano de teste foi criado e aprovado e casos de teste foram criados e aprovados pelas partes interessadas.

Critérios de saída: Os critérios de saída para o teste do sistema em cada iteração podem incluir: todas as funcionalidades determinadas pelo plano de teste foram testadas e aprovadas, todos os casos de teste de alta prioridade foram executados e aprovados no teste.

Critérios de conclusão do teste: em cada iteração, para se ter como concluído o processo de teste, o código deve ter sido estruturalmente testado utilizando o critério todos os branches, todas as funcionalidades selecionadas no plano de teste devem ter sido testadas utilizando o critério de análise de valor limite e os testes de mutação devem apresentar um resultado de pelo menos 85% dos mutantes eliminados e o relatório dos testes realizados deve ser gerado.

Grau de independência: cada equipe receberá um testador, o qual se reportará ao líder da sua equipe. No entanto, os testadores também podem reportar qualquer risco ou problemas ao Chefe de Testes da Hangman Testing.

Técnicas de projeto de teste: no nível unitário, deverá ser aplicado as técnicas de teste com o critério de análise de valor limite, de teste estrutural com o critério todos os branches e de teste de mutação. No nível de sistema, os testes deverão ser realizados utilizando o Cucumber e as histórias deverão seguir a sintaxe Gherkin. Nos demais níveis, a escolha das técnicas ficará a critério do líder da equipe.

Ambiente de teste: Testes de unidade de integração deverão ser realizados no ambiente de desenvolvimento pelos testadores. Testes de sistema serão realizados no ambiente de teste do sistema. Testes de aceitação serão realizados no ambiente de pré-produção pelos usuários. Testes de desempenho, acessibilidade, penetração e desastre/recuperação também são realizados no ambiente de pré-produção. A verificação da produção e os testes de penetração são realizados no ambiente de produção.

Métricas:

Ao final de cada iteração:

- É coletado o número de testes aprovados, reprovados e bloqueados.
- É medida a cobertura de automação por meio da coleta da porcentagem de histórias de usuários cobertos por testes de automação.
- É coletado e avaliado o número de defeitos encontrados por nível de gravidade, a fim de identificar oportunidades de melhorias para as próximas iterações.

Reteste e teste de regressão: os testes deverão ser automatizados sempre que possível, a fim de reduzir os custos relacionados aos testes de regressão. Um teste de regressão automatizado ou manual deverá ser criado para todo defeito detectado.

Hangman Testing

Departamento de Produtos - Time de Tecnologia e Desenvolvimento

Autores: Pedro Vinícius (201907494), Sávio Matheus de Sousa (201907505)

Descrição do Software em Teste

A aplicação em teste, *Hangman*, trata-se de uma implementação do jogo da forca para dois jogadores por meio de uma interface de texto. Serão descritos a seguir os principais fluxos de interação do programa e a sua estrutura interna, sem muito detalhamento.

O primeiro jogador, ao iniciar a aplicação, deverá fornecer uma palavra para ser adivinhada pelo segundo jogador. Este deve ser chamado em seguida. Na tela haverá a quantidade de erros permitida e uma sequência de traços representando as letras da palavra. Após inserir uma letra, o jogador pode se deparar com as seguintes respostas:

- a quantidade de erros restantes mais a sequência de traços contendo a (nova) letra inserida, caso ela esteja na palavra;
- uma quantidade menor de erros restantes e a mesma sequência de traços (e letras), em caso de erro do jogador;
- um aviso de letra repetida e mesma quantidade de erros restantes.
- uma mensagem como “a palavra certa era…” caso acabem as tentativas e o jogador não tenha conseguido adivinhar;
- uma mensagem “você ganhou com N tentativas restantes…”, caso o jogador adivinhe a palavra.

Internamente, o software testado está dividido em três classes: Hangman, Game e Prompter. A primeira contém o ponto de entrada e é responsável por instanciar um objeto Game e passá-lo para uma instância do Prompter. A classe Prompter constrói a interação com o jogador, exibindo mensagens e recebendo a letra seguinte. A classe Game contém a lógica do jogo e é manipulada através da classe Prompter.

Hangman Testing

Departamento de Produtos - Time de Tecnologia e Desenvolvimento

Autores: Pedro Vinícius (201907494), Sávio Matheus de Sousa (201907505)

Plano de Teste

1. Introdução

Este documento apresenta as atividades de teste do sistema *Hangman* e servirá de utilizado como guia para as atividades de acompanhamento, revisão, verificação e validação durante todo o processo. Ao final, da fase de teste, os resultados e processo realizado será analisado, comparado com o que foi planejado, com a finalidade de encontrar desvios e aplicar ações corretivas em testes futuros.

2. Funcionalidades a serem testadas

A fim de tornar o processo mais ágil, a única classe que será testada é a *Game.java*, a qual agrupa a lógica de funcionamento do jogo *Hangman*. Serão utilizadas as seguintes técnicas de teste:

1. Teste Funcional com o critério de análise de valor limite.
2. Teste Estrutural com o critério todos os branches.
3. Teste de Mutação.

A seguinte tabela contém os métodos a serem testados, com sua descrição e técnica ser utilizada:

Método	Descrição	Técnica(s)/Critério
boolean guess(String s)	Verifica se uma string é a na resposta	2 e 3
boolean guess(char)	Verifica se uma letra se encontra na resposta	2 e 3
String getCurrentProgress()	Obtém progresso atual.	2 e 3
getRemainingTries()	Obtém quantidade de tentativas restante	3
boolean isSolved()	Verifica se a resposta foi adivinhada.	3
String getAnswer()	Obtém a resposta	3

Além disso, ao nível de sistema, as funcionalidades deverão ser testadas utilizando o Cucumber.

Hangman Testing

Departamento de Produtos - Time de Tecnologia e Desenvolvimento

Autores: Pedro Vinícius (201907494), Sávio Matheus de Sousa (201907505)

Documentação de Casos de Teste

Caso 1

Teste do método `boolean guess(String s)`.

palavra escolhida	entrada	saída esperada
resposta	“r”	true
resposta	“l“	false
世界	“七”	false
(qualquer palavra)	“”	IllegalArgumentException
(qualquer palavra)	null	IllegalArgumentException

Caso 2

Teste do método `boolean guess(char s)`.

palavra escolhida	entrada	saída esperada
resposta	‘r’	true
resposta	‘l’	false
世界	‘世’	true
世界	‘セ’	false
(qualquer palavra)	‘r’,’r’	IllegalArgumentException
(qualquer palavra)	‘1’,’1’	
(qualquer palavra)	null	IllegalArgumentException
(qualquer palavra)	‘1’	IllegalArgumentException
(qualquer palavra)	‘🤨’	IllegalArgumentException

Caso 3

Teste do método `String getCurrentProgress()`.

palavra escolhida	tentativas feitas	saída esperada
“resposta”	‘r’,’s’	r-s--s--
“世界”	‘世’	世-

Caso 4

Teste do método `boolean isSolved()`.

palavra escolhida	tentativas feitas	saída esperada
resposta	‘r’, ‘e’, ‘s’, ‘p’, ‘o’, ‘t’, ‘a’	true
resposta	‘r’, ‘e’, ‘s’, ‘p’, ‘o’, ‘t’	false
世界	‘世’	false
世界	‘世’, ‘界’	true

Caso 5

Teste do método `String getAnswer()`.

palavra escolhida	saída esperada
resposta	resposta
世界	世界

Caso 6

Teste do método `String getRemainingTries()`.

palavra	tentativas	tentativas restantes (saída)
“resposta”	‘l’	5

Hangman Testing

Departamento de Produtos - Time de Tecnologia e Desenvolvimento

Autores: Pedro Vinícius (201907494), Sávio Matheus de Sousa (201907505)

Relatório de Teste

Relatório de testes para: Hangman Game, versão 1.0.0

Data: 09 de setembro, 2022

Cobertura: Primeira versão estável (1.0.0), planejada para o próximo ciclo de lançamento.

Riscos: Todos os riscos restantes foram comunicados ao Gerente de Produto e aceitos por ele. São relacionados à localização de caracteres, podendo afetar o funcionamento da aplicação com palavras em sistemas de escrita incomuns. Dado o público alvo e considerando as ferramentas da linguagem de programação utilizada, os impactos serão quase inexistentes.

Resultados dos testes: O Gerente de Produto aceitou a conclusão desta versão com base nos seguintes critérios:

- Os testes unitários atingiram cobertura de 100% no componente especificado (Game.java);
- Os testes de mutação indicaram 100% de eficácia dos testes unitários desenvolvidos;
- A nível de sistema, os principais casos de uso do Hangman Game foram cobertos e todos passaram nos testes elaborados.

Observações para as iterações futuras:

- É necessário adaptar o código-fonte para facilitar a testagem dos demais componentes da aplicação;
- Foram observadas incompatibilidades entre o Cucumber e a IDE Eclipse mais recente, sendo necessário utilizar uma versão mais antiga desta (2021-09);
- Novos tipos de teste podem ser usados para aumentar a robustez do processo, por exemplo, teste de estresse com grandes quantidades de dados e sistemas de escrita menos utilizados.