ES4A4 – Exercícios

Aula 5 – Verificação, Validação e Qualidade de Software
Revisões, Inspeções e Técnicas de Análise Estática

Instruções para entrega das listas de atividades:

Meio de Entrega: Os arquivos das resoluções das listas de exercícios devem ser entregues exclusivamente no ambiente Moodle (http://eadcampus.spo.ifsp.edu.br). **Não serão aceitos links para repositórios externos.**

Forma de Entrega: As respostas das questões devem ser entregues em um único arquivo PDF, cujo nome deverá conter a informação da aula, o nome e o sobrenome do aluno. Por exemplo: Aula2_MariaPereira.pdf.

Prazo de Entrega: O prazo de entrega está definido na página da atividade no Moodle, lembrando que o sistema bloqueia o envio de arquivos após a data e horário indicados.

1. **(1,0 ponto)** Explique com suas palavras o que são técnicas estáticas e dinâmicas de verificação e validação de qualidade de software. Dê exemplos de cada uma destas técnicas.
Técnicas estáticas são aquelas que podem ser feitas sem executar o programa, exemplo: Análises de documentação e requisitos da aplicação para inspeção.
Enquanto técnicas dinâmicas são aquelas que são realizadas com a execução do programa, por exemplo: Testes unitários para detecção de problemas por meio da análise da execução e comportamento do programa.

2. **(1,0 ponto)** Que tipos de erros são improváveis de serem descobertos por meio de inspeções?
Erros decorrentes de falhas e exceções em interações entre componentes de um programa são improváveis de serem descobertos por inspeções.

3. **(1,0 ponto)** Em que situações as inspeções técnicas são melhores que os testes de software para detectar problemas nos artefatos produzidos durante o processo de desenvolvimento?
As inspeções são melhores para detectar falhas no desenvolvimento do software em etapas ao qual a funcionalidade não foi totalmente implementada no sistema, problemas arquiteturais, de qualidade e projeção, além de analisar separadamente os erros ocorridos no código, diferentemente dos testes unitários que podem os mascarar sob um grande erro em destaque.

4. **(1,0 ponto)** Supondo uma empresa de desenvolvimento de software que possui um processo formal de gerenciamento de qualidade estabelecido e consolidado, quais as implicações caso esta empresa passe a adotar metodologias ágeis de desenvolvimento em seus projetos?
As metodologias ágeis não tem processos de inspeção e revisão voltados a qualidade que sejam fortemente consolidados e definidos. Uma mudança de metodologia na empresa implicará em mais mudanças drásticas que precisariam ser feitas no modelo de trabalho e gerenciamento dos projetos, pois toda sua lógica seria alterada, incluindo a questão do gerenciamento de qualidade que deverá ser reformulado ou terá de abrir mão do modelo.

5. **(1,0 ponto)** Uma das características das inspeções formais é que elas tornam públicos os erros dos membros da equipe. Quais as implicações disso e quais ações podem ser tomadas para que isso não prejudique o ambiente de trabalho?
A exposição dos erros pessoais dos programadores implicará em discórdia no ambiente de trabalho ou tensões entre os membros por haver problemas no código por suposta culpa de uma pessoa específica. Para que isso não prejudique o ambiente de trabalho, é importante que a empresa não procure culpados para os problemas ocorridos no projeto, e sim apenas comunique os pontos de

melhoria como uma responsabilidade geral da equipe; Não é um indivíduo culpado, todos são responsáveis pelo projeto.

6. **(3,0 pontos)** Elabore um checklist de inspeção para verificar possíveis erros no programa desenvolvido na Aula 3. Em seguida, use esse checklist para inspecionar o código do programa. Obs.: Além do checklist, entregue as observações feitas após a verificação de cada item deste checklist. Para itens, que estiverem corretos, basta informar “ok” na observação.

Classe de defeito	**Verificação de inspeção**
Defeitos de dados	Todas variáveis foram inicializadas antes do programa ser executado? - **ok** Todas variáveis estão sendo utilizadas? - **ok** Todos os métodos estão sendo chamados adequadamente e com a quantidade de parâmetros necessários? - **ok** Os arrays foram inicializados antes de começar a inserção de informação? - **ok** Todos os objetos estão sendo instanciados e inicializados adequadamente? - **ok**
Defeitos de controle	Todas as opções possíveis de estão sendo tratadas adequadamente no switch/case? - **O switch/Case está tratando adequadamente as opções numéricas fornecidas pelo usuário, mas está caindo em InputMismatchException caso seja digitado um valor não esperado no input.** O programa está tendo break no momento esperado? - **ok** O programa está exibindo as informações de saída no momento correto? - **ok** O loop do programa está executando e terminando de forma esperada? - **ok**
Defeitos de entrada/saída	Os campos de input do programa estão com prevenção para caso o usuário insira dados inesperados? - **Não há filtro que evite exceções em caso do usuário digitar um input inesperado.** Todos os campos de input estão sendo utilizados? - **ok**

Defeitos de interface	Todas as instâncias dos objetos estão sendo iniciadas com todos seus parâmetros? - **ok** Os parâmetros estão sendo pegos na ordem correta?- **ok**
Defeitos de gerenciamento de armazenamento	O espaço alocado para as variáveis é adequado? - **ok** As variáveis alocadas dinamicamente estão sendo “limpas” ao término da execução? - **ok** O método de armazenamento das listas é adequado para o objetivo do sistema? - **Poderia ser usado um array comum [] de objetos ao invés de um ArrayList para poupar memória na execução da aplicação e executar as mesmas funções exigidas.**
Defeitos de gerenciamento de exceção	Levou-se em consideração a possibilidade de erro ou exceção na manipulação das listas? - **ok** Levou-se em consideração a possibilidade de exceção na manipulação das variáveis? - **Não há validação para os argumentos inseridos nas variáveis, por exemplo: o usuário poderia digitar uma única letra e ela poderia ser atribuída no campo de nome do autor, algo que ão deveria ocorrer.** Levou-se em consideração a possibilidade de exceção nos inputs e instância de objetos na aplicação? - **Não há verificação para tratar exceções de input ou instância de objetos no programa.**

7. **(1,0 ponto)** Por que pode ser vantajoso usar especificações e verificações formais no desenvolvimento de sistemas de software críticos? Quais os pontos negativos dessa abordagem?
As verificações formais utilizam conceitos matemáticos para testar a lógica do programa, e assim auxilia a descobrir erros, problemas como inconsistência e ausência de requisitos do software. Essa abordagem tem como pontos negativos a representação dos requisitos dos clientes pode não ser totalmente fiel; As verificações de programas extensos pode conter diversos erros devido à complexidade lógica do projeto e por ser um processo muito técnico e exigente de análise matemática, o custo para se realizar essas verificações em um projeto é alto.

8. **(1,0 ponto)** Sugira cinco erros que poderiam ser definidos pelo desenvolvedor para serem detectados por um analisador estático de programas escritos em Java. Não considere erros triviais comumente destacados pelo compilador.

1- Input mismatch Exception → No caso do usuário digitar uma entrada inesperada pelo programa.
2- Throws Exception → Método perigoso do Java que caso não tratado poderá resultar em uma exceção de execução.
3- StackOverflow Error → Erro que pode ocorrer em caso de muitas requisições que façam um flood na pilha de execução do programa.
4- Erro de threads → Falha na lógica de threads do programa que ocasione a execução em ordenação errada das threads do programa.
5- IndexOutOfBounds Exception – verificar se a lógica de call dos arrays no código não corre o risco de lançar uma exception na possibilidade do array não ter itens ou ter menos itens que o esperado pelo programa.