Pesquisador. Muito obrigado por ter aceito participar dessa entrevista. Ela tem como objetivo compreender os processos de criação, manutenção e verificação da qualidade do código de teste de unidade. Não há respostas certas ou erradas ou desejáveis ​​ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que realmente pensa. Ok?

Entrevistado. Ok

P. Reforçando um item do termo de consentimento, nossa conversa está sendo gravada. O objetivo disso é para que eu possa obter todos os detalhes, mas ao mesmo tempo ser capaz de manter uma conversa atenta com você. Garanto que todos os seus comentários permanecerão confidenciais. O relatório final conterá os comentários dos participantes sem nenhuma referência aos indivíduos. Ok?

E. ok

P. Como você começou a trabalhar com teste de software?

E. Foi de forma bem natural, na verdade desde o meu primeiro emprego, primeiro estágio, a vida inteira eu trabalhei com teste de software. Os meus empregadores e gestores, o pessoal que estava nos projetos quando entrei já tinha essa consciência e eu aprendi na faculdade, já levei isso comigo, então foi de forma bem natural.

P. Quais foram suas fontes de aprendizado sobre código de teste de unidade?

E. Foi uma disciplina na faculdade sobre testes, eu li referências sobre o livro do Kent back, sobre o TDD. Eu não cheguei a ler o livro, mas conversei com colegas que leram o livro, basicamente o TDD é a forma que eu mais usei e que eu emprego hoje.

P. Em quais linguagens você cria testes?

E. Java, Javascript e Typescript

P. Qual linguagem do seu projeto atual?

E. essas 3 Java, Javascript e Typescript

P. Como é seu processo de criação de teste?

E. quando é possível eu faço o TDD, eu tento fazer a variação do BDD, que é orientada a comportamento, é só uma forma de escrever o teste de forma mais fluida, mas não deixa de ser o teste antes do código. Quando é um código novo, um projeto novo, ou que ha está bem estruturado, bem desacoplado mais fácil de trabalhar, eu procuro fazer o TDD, agora tem alguns projetos que já são legados, que o código é um pouco mais complexo que eu faço primeiro as vezes o código, testo manual e depois faço o teste automatizado. De todas essas duas formas, mas de todo o jeito eu entrego o teste junto com a tarefa. É um pré-requisito para entregar a tarefa para a revisão de código é ela estar com os testes.

P. Existe algum fluxograma, template na empresa ou equipe que padronize esse processo?

E. Não. Depende do projeto. Tem alguns projetos que de forma orgânica já tem um padrão. Isso é muito feito de forma orgânica. O pessoal começa um projeto novo e os mantenedores do código se organizam dessa forma e começam a exigir isso no code review. O code review que geralmente padroniza essas coisas. Agora ter uma cartilha escrita, um lugar que esteja escrito para ser seguido, não.

P. Quais ferramentas são utilizadas para criar/ executar teste?

E. Em Java, a gente usa o Mockito, o Mocha, que é de Javascript, o SinonJS que é da parte de Javascript também e o Chai. Wire mock para Java, JUnit e AssertJ.

P. Como é realizada a verificação de qualidade do código de teste criado, por exemplo: verificação de escrita, você falou sobre o code review, então você pode falar um pouco mais sobre esse quesito, verificar compatibilidade com o requisito... como a verificação da qualidade do código é feita?

E. Basicamente a verificação é feita no code review. Tanto no nível de design pattern, legibilidade e tudo mais e isso se aplica tanto ao código que resolve o negócio quanto o código de teste, e quanto ao atendimento dos requisitos também.

P. Nesse processo de code review, quantas pessoas revisam um código submetido?

E. Varia muito. No mínimo uma, vai muito do feeling nosso, então a gente tem um acordo se uma tarefa foi feita com peer programing, não precisa de review, então se já foi feita com duas pessoas, já subentende que o review já foi feito ali ao vivo, automaticamente. Aí dependendo da outra tarefa, se for de um maior impacto, que a gente avalia nas reuniões de time que é uma tarefa mais crítica, de maior impacto, aí a gente procura mais uma pessoa, ou se for uma tarefa mais cotidiana, que o problema já está mais ou menos dominado, uma pessoa pode ser suficiente.

P. Como é realizada a manutenção do código de teste de unidade?

E. Ele é embutido no ciclo de vida do software como um todo. A gente não tem ações específicas para manter o teste. Geralmente o teste vai ser mantido ou quando muda o requisito ou quando muda a estrutura do código mesmo, sem alterar o requisito, alguma refatoração ou uma melhoria que pode quebrar o teste, aí ele precisa ser mantido, mas a gente não tem uma atividade com foco em manutenção de teste desligada dos requisitos não.

P. O que você conhece sobre test smell?

E. Não conheço

P. Eu vou te passar o conceito sobre o assunto, vou te passar também 2 exemplos e se precisar a gente conversa sobre, porque as perguntas a seguir têm a ver com esse tópico, tá?

E. Tá.

P. “Test smells são pontos fracos na estrutura do código de teste, tendo um impacto negativo na sustentabilidade da atividade de teste, pois os tornam complexos, difíceis de entender e modificar, prejudicando a repetibilidade, independência e estabilidade dos testes.”

E. Ah tá, entendi. É tipo code smell só que aplicado ao teste.

P. Isso. Os test smells eles descendem dos code smells, foi pesquisado e identificado que não necessariamente aplicar os mesmos smells de código para teste. Existem várias pesquisas que identificam diferentes smells voltados para teste. Eu vou te mandar 2 exemplos de smells, para fechar essa questão conceitual. Caso você tenha alguma dúvida, alguma pergunta, é só me perguntar. O primeiro dele é o chamado Conditional Test Logic smell. Ele é considerado um smell porque ele tem estrutura condicional e de repetição dentro dele. Então ele torna-se difícil de compreender, ao depender da experiência do desenvolvedor, e uma vez que ele falhe fica mais difícil identificar em qual momento dessa estrutura esse teste falhou. O outro exemplo é o chamado Empty test smell. ele é considerado um smell porque não tem nenhum script executável. Em relação a manutenção do teste acaba sendo prejudicial pois não tem informação de porque ele está comentado, ao invés estar sendo executado, ou porque ele não foi excluído e permanece no conjunto de testes da aplicação. Então, não é necessariamente uma dificuldade de se ler o teste, mas de compreender de porque esse teste está aí, então ele acaba prejudicando a manutenção da atividade de teste em si. Existem outros tipos de smells relacionados a asserts, a valor de parâmetro, a construtores, mas todos eles vao acabar impactando na sustentabilidade da atividade de teste, seja pela complexidade, por dependência, instabilidade desses testes e assim por diante. Ok?

E. Ok

P. Alguma dúvida?

E. Não, tranquilo

P. Como test smells são tratados no processo de criação do código de teste?

E. Volta no mesmo ponto. A gente foca no code review. Geralmente isso vem de quem tem mais experiência no sistema. É meio que dever de um sênior e dos especialistas de orientar a galera enquanto a isso, mas isso é mais natural, que se siga essa hierarquia, mas não obrigatoriamente, pode muito bem um estagiário dar palpite no código de um especialista. mas é sempre no code review, porque a gente sempre trata no quesito legibilidade de teste e não só legibilidade, mas manutenibilidade. Então o que a gente faz, aqui na empresa quando a gente sente que alguma coisa está mais enrascada independente de ser teste ou padrão de projeto, ou arquitetura, a gente as vezes faz algumas reuniões técnicas para discutir o melhor caminho, mas basicamente é com a revisão também.

P. Então só para formalizar, esse processo vale tanto na criação quanto na manutenção do código de teste? Vocês fazem algum cuidado específico para manutenção em relação a test smell?

E. A gente segue os mesmos princípios em alto nível de legibilidade e manutenibilidade, só que aí a gente já conhece um pouco das características do teste, o teste que tem muitos asserts, testes que ficam confuso o que ele tem que fazer, as vezes tem um escopo muito grande sendo testado. O que a gente tenta forçar aqui seria o teste ficar bem legível no sentido BDD, a gente tenta dado tal cenário, tal configuração, executa uma ação apenas e depois dessa ação a gente faz asserts para verificar o estado ou o resultado obtido, então a gente sempre tenta visualizar o teste nessas 3 etapas, são 3 passos simples: dado um contexto, executa uma ação, verifica o resultado. A gente tenta focar nessa parte do TDD, dessa forma de pensar, tenta verificar se não tem muitos asserts se um teste pode ser quebrado o máximo que ele pode ser quebrado, talvez aplicando o conceito de single responsability do solid, de alguma forma aplicar isso aos testes, mas tudo orgânico no processo de code review como um todo.

P. Para você, como seria possível prevenir ou evitar a inserção de test smell no código de teste no momento que esse é criado? Ou seja, antes do code review até, como seria possível alguma prática de programação, processo, ferramenta, que pudesse evitar que o desenvolvedor inserisse esse smell no código de teste desde o momento da criação?

E. Capacitação. Eu acho que capacitação e modelos para serem seguidos. Eu não gosto muito da parte de documentação no sentido mais antigo da coisa, de ter uma cartilha, um livro, para mim a documentação seria um processo do github de modelo. Aí naturalmente ele teria um readme que é uma parte textual, só que uma parte textual simplificada que explicasse as coisas em alto nível e que o detalhe estivesse no próprio código e que fica mais fácil de manter aquilo. Para mim o ideal seria a gente ter um projeto de modelo para testes que documente e coloque essas boas práticas de testes todas com exemplos e que isso faça parte de onboard do funcionário novo, então se a pessoa entrou na empresa, a gente passa por esse treinamento, talvez com uma vídeo aula associada, mas que ele saiba que a qualquer momento que ele tiver uma dúvida que ele for criar ele vai ter esse modelo e esse guia para seguir

P. Estou chegando no final da entrevista, você tem alguma pergunta para mim, quer acrescentar alguma informação ou ainda tem alguma sugestão para melhorar a entrevista?

E. Não. Acho que foi bem tranquilo, foi bom.

P. Muito obrigado.