Pesquisador. Gravação iniciada, mais uma vez muito obrigado por participar da entrevista. Ela tem como objetivo compreender os processos de criação e manutenção do código de teste e a qualidade do código de teste. Não há respostas certas ou erradas ou desejáveis ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que realmente pensa.

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 e no relatório final conterá os comentários dos participantes sem nenhuma referência aos indivíduos.

E. OK

P. Bom, para começar como você começou a trabalhar com teste de software?

E. Olha, para mim foi como cair de paraquedas, não é? Estava na faculdade e recebi um e-mail de uma empresa, a empresa A, informando sobre o cargo de analista de qualidade de testes. Eu não conhecia nada a respeito, submeti ao processo seletivo, fiz a entrevista, fiz uma prova de seleção foi aprovado. E aí a gente começou a trabalhar. Isso foi em janeiro de 2011.

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

E. Bem, primeiramente foram os colegas de trabalho, colegas de faculdade e de profissão, segundo estudos de extensão, cursos, contatos com outros programadores, com outros analistas de testes.

P. Em quais linguagens você cria testes?

E. Atualmente, não tenho mexido com codificação, né? Com programação para criar casos de testes. A última linguagem que trabalhei, as últimas linguagens que trabalhei para criar casos de testes foram Java, Ruby e C#.

P. OK

E. As últimas e as únicas.

P. Certo. Na sua última experiência, como era o processo de criação de teste, criação do código de teste?

E. Primeiramente a gente recebia os casos de uso que descreviam a função de determinado parte do código, era um sistema web. A partir dos casos de uso, a gente modelava testes manuais, para validação dos fluxos básicos e fluxos alternativos. A partir da criação desses casos de teste básicos e alternativos, a gente modelava os testes automatizados. Como "prioridade um" na criação dos casos de teste, primeiro ficavam os testes de fluxo básico para a gente montar as features de regressão, mas era esse fluxo: primeiramente casos de uso, para modelagens de testes manuais, a partir desses testes manuais, a gente traduzia para os testes automatizados.

P. Ok. Dentro da empresa, onde você teve essa experiência, como era o processo de verificação da qualidade do código de teste criado? Por exemplo, verificação da escrita, se alguém revisava o teste depois que ele era criado, ele em algum momento era verificada a compatibilidade com o requisito, ou classe de produção?

E. Bom, o fluxo era o seguinte: o analista de testes modelava o teste automatizado e o líder revisava o fluxo para ver se tinha erro de lógica, se estava atendendo o requisito, se estava tudo correto. Se o líder modelasse o teste automatizado, outro analista faria a validação e, de uma maneira aleatória, as vezes o gerente também fazia validação de caso de teste, para ver se tinha algum erro, algum desvio e para propor melhorias também.

P. O processo de criação de código de teste, ele seguia algum fluxograma, algum template, ou seja, existe algo que padronizasse esse processo?

E. A gente utilizava o framework do Page Object. Antes de eu sair já estava começando a modelar alguma coisa utilizando o BDD também.

P. Passando para processo de manutenção do código de teste, me conte como ele era feito na empresa?

E. Geralmente a gente enxergava os testes que falhavam durante a execução dos testes de regressão, e quando havia falha, código desatualizado, imediatamente após os resultados para o cliente e aí a gente já colocava esses testes sob demanda para deixar a suíte redonda para a próxima execução. Às vezes também quando havia baixa demanda de testes, a gente aproveitava para rodar aleatoriamente a suíte para tentar fazer uma manutenção preventiva do código.

P. Me conte o que você conhece sobre test smell.

E. Não, não conheço test smell.

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

E. Bom, para evitar esse tipo de problema, a gente estruturou em TI, em Page Objects para deixar a manutenção do código mais simples de fazer e a gente também implementou a rastreabilidade. Então a gente sabia quem fez a manutenção, quando fez a manutenção, quem escreveu, quem validou a última execução, onde e qual o método que estaria quebrando. Então a manutenção se tornava mais prática. E quando havia necessidade, em alguns métodos haviam comentários, para especificar o que estava se testando ali. Fora que a gente também adequava o nome do método geralmente ao que ele executava no sistema que estava sob teste. Incluindo entidade, pesquisar entidade... isso daí.

P. Certo. Você tem alguma dúvida, você quer mais alguma informação sobre esse tema? Eu posso te passar alguns exemplos e depois a gente segue, porque as perguntas seguintes são relacionadas a test smell.

E. Pode passar mais um exemplo.

P. Ok, eu vou te mandar agora duas imagens de exemplos de test smell e a gente conversa sobre essas duas imagens para você ter uma ideia, tá?

E. OK, Pode mandar.

P. Eu te mandei duas imagens, uma é relacionada ao tipo de test smell chamado Conditional Test Logic. Esse test smell é considerado um smell, considerado como um ponto de melhoria, pois o teste tem que ser objetivo e voltado a um único componente, então a partir do momento que você inclui condicional, ou repetição nessa estrutura, uma vez que o teste falhe você vai ter uma dificuldade de identificar em que momento esse teste falha, em qual condição ou em qual repetição ele falha. O outro é o Empty Test, ele é bem mais simples de se entender que é basicamente um caso de teste sem nenhuma estrutura lógica de execução executável dentro do código de teste, então ele fica mais difícil de se manter uma vez que você vai ter um conjunto de testes e alguns testes sem nada a executar, se você tiver uma quantidade grande disso, isso vai começar a ter dificuldade de realizar a manutenção desses testes, ok?

E. Beleza.

P. Alguma dúvida?

E. No momento não, se surgir, eu te pergunto.

P. Perfeito. Você comentou que em se tratando desse tipo de test smell, isso era tratado de alguma forma no momento que o teste será criado era identificado, em que classe de produção ele corresponda e algumas informações a respeito para identificar quem criou, conseguia rastrear esse teste no momento ele era criado, não é isso?

E. Correto.

P. Ok. isso Ele era identificado, só para eu conseguir rastrear melhor, através de alguma ferramenta ou dentro do código mesmo, ele era colocado?

E. Não, não exatamente, quer dizer sim, dentro da ferramenta, dentro da nossa IDE, porque a partir da IDE a gente estruturava as classes, cada classe seria um teste então, o nome de cada classe era um respectivo teste criado manualmente e dentro do teste tinha toda rastreabilidade da sua criação e da sua modelagem, dos métodos também.

P. Qual a IDE?

E. A gente trabalhava inicialmente com o Eclipse quando a gente fazia a codificação Java mais Selenium e a partir do momento que a gente tentou montar alguma coisa em Ruby, mas a gente matou o projeto em Ruby e fomos para C# a gente passou a usar o VSCode.

P. OK. Para esse tipo de estrutura isso também era tratado na manutenção do código de teste?

E. Sim. Principalmente.

P. Principalmente na manutenção mais do que na criação?

E. Nas duas coisas. A gente tinha um montado uma estrutura, um esqueleto na verdade então a para gente escrever um novo caso de teste a gente já tinha aqueles métodos mapeados já. Então eram métodos de rastreabilidade. Então era mais um Control C + Control V. No último momento quando a gente passou para o C#, a rastreabilidade ela estava se dando através dos gits.

P. Para você como seria possível prevenir ou evitar a inserção de test smell no código de teste no momento que o código de teste é criado?

E. Eu acredito que principalmente a gente tem que ter uma estrutura de rastreabilidade algo similar com o que a gente trabalhou na última empresa. Saber que pode ser um comentário, pode ser um método, mas alguma coisa que descreva quem criou, quando criou, quando foi feita a manutenção, quando foi feita a validação, em que parte do escopo do objeto de teste aquele teste pertence.

P. Certo, isso em termos de rastreabilidade. Em se tratando da qualidade do código, você identifica alguma técnica de programação ou processos organizacionais que podem ser adotados?

E. O que eu trabalhei que eu achei bem bacana foi a parte de você utilizar o Page Object, porque você cria um método que ele pode ser replicado em várias partes do código, em várias classes diferentes, em vários pacotes diferentes também, basta você instanciar o cara lá. Você pode criar um método de criação de entidade, clicar num botão e esse método ser instanciado em diversos pacotes, diversas classes o que facilita a modelagem dos testes automatizados, porque você tem a reutilização de código.

P. Certo.

E. Além da praticidade, você ainda tem um padrão.

P. Você tem alguma dúvida para mim ou você quer acrescentar alguma informação a respeito, ou você tem alguma sugestão para melhorar essa entrevista?

E. Não. Acredito que está bem alinhada, bem objetiva também.