Pesquisador. Muito obrigado por aceitar participar da entrevista. Ela tem como objetivo compreender os processos de criação e de manutenção de código de testes e como funciona a verificação da qualidade do código de teste nesses processos. Não há respostas certas ou erradas ou desejáveis ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que realmente pensa.

Entrevistado. Uhum.

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.

E. OK

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

E. Eu comecei a trabalhar logo no início da carreira, mas não foi por iniciativa própria. Eu acredito que tem uma questão cultural que quando, por exemplo, eu era um programador júnior e quando você não consegue se desenvolver bem, as pessoas já te colocam para testar. Então foi isso que aconteceu comigo. Eu não tinha uma habilidade de programação grande, mas eles enxergaram em mim uma habilidade analítica. Então me colocaram para fazer testes de software. Então eu comecei a testar e até hoje sigo na carreira. Mas não foi por opção própria. Algumas pessoas que já vem da academia com essa mentalidade de querer testar o software, mas eu não tinha essa mentalidade.

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

E. As minhas fontes principalmente foram cursos online e fóruns. A base de tudo. Eu acho que ajuda bastante, mas principalmente o network com os colegas. Eu acredito que 90% das habilidades que consegui desenvolver foi por conta do network com as interações com os colegas, no dia a dia de trabalho. Isso ajuda a trazer teoria para prática. Então sempre que eu via algum curso, buscava algum conteúdo, aprendia alguma coisa em fórum, eu sempre contava com a ajuda dos colegas para conseguir implementar isso no dia a dia.

P. Em quais linguagens você cria testes?

E. Atualmente em Ruby, mas já criei alguns testes em Java e em Phynton. Sendo que minha proficiência é em Java. Estou agora iniciando os testes em Ruby e em Capybara.

P. Qual plataforma do seu projeto atual?

E. A gente tem uma plataforma em Ruby, totalmente em Ruby, só que a gente tem um projeto novo que talvez seja em Typescript. Atualmente a suíte que eu trabalho é tudo em Ruby.

P. Você trabalha com teste de unidade em Ruby?

E. Por diretiva da empresa os desenvolvedores fazem os testes de unidade. Nós trabalhamos com teste funcional.

P. Então sua experiência de teste de unidade foi em Java.

E. Sim.

P. Minha pesquisa o foco é em teste de unidade, então o que a gente conversar a partir de agora, tenta responder com sua experiência com teste de unidade, ok?

E. Beleza.

P. Como é seu processo de criação de teste?

E. Geralmente vinha uma documentação e pela primeira vez que fiz teste unitário era bem aberto. Você lia a documentação que vinha dos analistas, entendia o processo e aí você levantava alguns testes básicos antes de você liberar o código que você acabou de desenvolver. Então não era nada muito formal, mas você tinha que ter uma compreensão do que você estava fazendo e entender o processo que estava fazendo para que quando chegasse na mão de um QA ele não pegasse erros básicos.

P. Existia algum fluxograma, template... que padronize esse processo?

E. Sim. Havia um template. Você lia a documentação e baseado no que você desenvolveu você elencava seus testes unitários dentro desse template. E ao final dessa implementação, depois que o código era validado por ferramentas como o Sonar e outras, você submetia.

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

E. Geralmente nós utilizávamos o JUnit e esqueci de mencionar que, o primeiro teste unitário eu fiz em Java, mas tinha teste unitário em alta plataforma também, em Cobol. E em Cobol a gente usava uma ferramenta chamada RDZ mais ele era mais para motivos de controle, você não fazia nenhum tipo de upload, era parecido com checklist. Era um assertion com todos os seus testes unitários e o analista principal ele validava os assertions desses testes unitários antes de liberar o seu código de alta plataforma.

P. Como era o processo de verificação de qualidade do código de teste criado?

E. Na célula, a primeira que eu trabalhei havia os analistas sêniores, eles verificavam seu bloco de código de teste e eles pontuavam se precisava de alguns testes a mais, alguma cobertura a mais. Já em outros processos eu me lembro que haviam testes de caixa branca. Ou seja, uma analista de QA validava os seus testes no JUnit para ver se eles eram concisos. Mas o meu caso, era um analista sênior que revisava.

P. Passando para o processo de manutenção do código de teste, como ele era realizado na empresa?

E. Basicamente, antes de a gente publicar uma nova versão a gente usava o Jenkins e ele disparava os testes unitários. Então havia uma suíte de teste unitário que ele chamava de sanity que ele testava os princípios básicos do sistema antes de publicar. Então toda vez que a gente subia um novo deploy usando o Jenkins, essa bateria de testes unitários era executada antes de fazer a publicação.

P. Nesse caso se algum teste falhasse, era realizada a manutenção do teste, era isso?

E. Isso. Primeiro fazia uma avaliação, para saber porque o teste quebrou, se foi alguma funcionalidade nova ou o que aconteceu. Primeiro era avaliado se era alguma alteração do software que quebrou o teste unitário ou foi algum problema com o código do teste unitário. Então ocorria essa triagem antes de tomar alguma decisão.

P. O que você conhece sobre test smell?

E. Não tenho conhecimento.

P. As perguntas a seguir estão relacionadas a esse tema. Então vou te passar o conceito, te mando um exemplo, a gente conversa um pouco sobre e depois a gente segue, ok?

E. Ok

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. Uhum.

P. Eu vou te mandar dois exemplos de test smells, vou explicar conceitualmente porque eles são considerados smells e caso você tenha alguma dúvida a gente conversa, ok?

E. OK.

P. A primeira imagem que eu te mandei é do chamado Empty test. Ele é um teste que não tem nenhum script executável. Caso tenha uma quantidade grande de teste dessa forma vai mais complicado de se entender porque está comentado, porque ainda permanece dentro da suíte, então dificuldade na manutenção da atividade de teste, uma vez que ele existe. Já o segundo exemplo é o chamado Conditional Test Logic. Ele é considerado um smell porque uma vez que o teste com essa estrutura ele falhe, vai ser mais difícil de identificar porque ele falou considerando que existem estruturas de condição e de repetição dentro dele. Então os smells de teste ou test smells são falhas, pontos de melhoria dentro da estrutura do código de teste e que ele dificulta a atividade do teste em si. Não é necessariamente que vá estar diretamente ligada a uma falha no código do software, mas que pode impactar na manutenção da atividade de teste.

E. Entendi

P. Você tem alguma dúvida?

E. Não até o momento está claro. Qualquer coisa vou recapitulando com você.

P. Como os tests smells eram tratados no processo de criação do código de teste?

E. Praticamente não havia tratativa para os smells. Quando você tinha alguma suíte assim, o trecho comentado mesmo era muito comum. Por exemplo, essa funcionalidade foi tirada do sistema ou a regra dela mudou, simplesmente o teste era comentado e era deixado lá. Agora não sabia o porquê ele continuava lá se ele não tinha funcionalidade. Geralmente smells se fazia tratativas pontuais e deixava ele lá. Dificilmente alguém separava uma equipe, separava um squad, separava um tempo para tratar esses testes. Geralmente eles ficavam lá e quando saía alguma métrica final dos testes que passaram e que falhavam e praticamente ali já tinha algo meio cultural que eles já sabiam implicitamente que alguns desses testes que falhavam eram porque realmente eram erro. Então os smells acabam entrando na cultura do desenvolvimento.

P. Isso no processo de criação. Eles eram tratados no processo de manutenção?

E. Não. Eles só eram tratados em manutenção quando acontecia algo muito crítico, ou quando esse smells chegavam a uma criticidade muito alta. Por exemplo. Um teste que dentro dele tinham diversas condições de teste, e aquilo começou a ficar inconcebível. Então parava naquele momento e fazia praticamente apagar incêndio. "Concerta aquele smell ali, que o cliente sabe que não está testando direito e vamos parar para concertar". Então era tudo a nível de urgência. Não havia nenhum planejamento para fazer manutenção não.

P. Para você, como seria possível prevenir/evitar a inserção de test smell no código de teste no momento da criação dele?

E. Primeiro eu vejo uma dificuldade muito grande as vezes dos desenvolvedores de estar mais preocupado na funcionalidade em si. Eles querem entregar a demanda. As vezes a demanda vem atrelada com teste unitário para ele desenvolver, mas esse não é o foco dele. Então as vezes por não ser o foco dele, justamente por não ter uma tratativa muitas vezes de um desenvolvimento orientado a teste, ou seja, ele cria os testes depois e não antes, então ele acaba criando o teste meio com rapidez. Tem um ponto também da questão da modularidade da coesão. Uma questão de superclasse, de superclasse de testes que ao invés da gente quebrar as funcionalidades, a gente cria uma superclasse e sai jogando todo o tipo de condicional dentro dela. Isso acontece principalmente para funcionalidades mais complexas. Eu vejo isso bastante. Então você tem um problema aí de modularização. Então para mim no início seria você abordar a metodologia do desenvolvimento orientado a testes e para as funcionalidades mais complexas você trabalhar na modularização delas. Eu creio que isso evitaria muitos desses problemas.

P. Estou chegando no final já. Você tem alguma pergunta para mim, quer acrescentar alguma informação ou ainda tem alguma sugestão para melhorar nossa entrevista?

E. Como você está falando de testes unitários, seria bom colocar uma pergunta justamente sobre a maturidade, qual a percepção que a pessoa tinha ou que tem da maturidade do desenvolvimento orientado a testes. Porque eu vejo que essa é uma cultura que ajudaria muito no desenvolvimento dos testes unitários e na melhora do código, só que muitas vezes a gente não tem essa técnica implementada ou ela é feita de um jeito que não segue os padrões. Então acho que seria uma boa pergunta.

P. Muito obrigado. Eu vou finalizar a gravação. Muito obrigado por participar da entrevista.