Pesquisador. Mais uma vez muito obrigado por ter aceito participar da 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, desejáveis ​​ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que realmente pensa. Ok?

Entrevistado. Está certo

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

E. Está certo.

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

E. Eu comecei como estagiário na empresa em Porto Alegre, eu sou de lá, chamada empresa T, eu trabalho atualmente com desenvolvimento, mas o meu primeiro trabalho foi exclusivamente testar software. Eu comecei como estagiário, mas fui contratado na empresa e eu fazia tanto testes manuais quanto automatizados em Ruby. Era um e-commerce de um cliente nosso e eu basicamente fiquei um ano só testando a plataforma, um ano e meio e foi assim que eu comecei.

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

E. Minhas fontes foram mais eu já estudava isso durante a faculdade, eu acho que na cadeira de, não sei se em Engenharia de Software se eu não me engano, e lendo mesmo na comunidade, eu sempre li sobre desenvolvimento. Desenvolvo desde os 14 anos. Quando eu me envolvi com a comunidade Ruby, linguagem de programação durante a faculdade, eu tenho uma ênfase forte em teste unitário e ali eu comecei a aprender a escrever teste unitário como parte do desenvolvimento e daí quando eu comecei a trabalhar profissionalmente continuei aplicando isso, desde o meu trabalho inicial na empresa.

P. Em quais linguagens você cria testes?

E. Eu crio em Ruby principalmente, teste unitário. Teste de integração em Ruby e a gente tem utilizado muito Javascript no NodeJS, que tem uma biblioteca que integra com o Selenium e com o Chrome headless que a gente tem usado para teste de tela, teste de integração.

P. Hoje com o teste de unidade você trabalha basicamente com o Ruby, no caso?

E. Ruby e Javascript também. Tenho que adicionar isso. Porque eu tenho trabalhado bastante com o node e a gente escreve teste unitário em Node também. Node e Ruby tanto para integração quanto para unitário.

P. Como é seu processo de criação de teste de unidade?

E. Atualmente é um pouco um mix as vezes eu escrevo antes de escrever o código, o TDD mesmo, mas eu acho que a maior parte das vezes eu escrevo um código, alguma funcionalidade e escrevo o teste depois. Não necessariamente eu faço toda a funcionalidade depois do teste mas em partes, em pequenas partes. Escrevo o código e depois o teste. As vezes eu faço o teste antes também. Depende muito da situação. Quando eu quero explorar a solução eu primeiro escrevo o código. Quando eu já sei o que eu vou escrever eu crio o teste primeiro, vejo falhar e escrevo o código depois.

P. Existe algum fluxograma, template documentado que padronize esse processo de criação?

E. Não. Na empresa que eu estou atualmente. Eu trabalhei em outras, como a empresa T que era muito mais TDD, era incentivado pelo menos, não era uma coisa forçada. Normalmente era TDD, teste primeiro e depois escrever o código. Atualmente não tem nada assim. A gente não costuma commitar código sem teste. Embora não esteja escrito é um consenso já entre os desenvolvedores da empresa.

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

E. De unidade no Ruby a gente está utilizando a AirSpec e para o Javascript acho que é o Mocha, eu só escrevo não lembro qual a biblioteca. Mocha, Jasmine e Jest também. São esses.

P. Dentro do processo de criação do código de teste, como ocorre o processo de verificação de qualidade do código de teste criado, por exemplo: verificação de escrita, alguém revisa o teste depois de criado, compatibilidade com o requisito, cobertura e assim por diante?

E. Depende muito da aplicação que a gente trabalha aqui, tem mais de 60 eu acho. Mas basicamente a gente está usando um verificador de sintaxe, que por exemplo para o Ruby é o Robocop, para Javascript é o ISLint. Então eles fazem um check básico de sintaxe, variáveis declaradas mas não utilizadas, então isso roda local e no CI. Toda vez que a gente commita o código no repositório a gente tem o circle CI ou o Travis que vão rodar os testes automatizados, vão garantir que todos estão passando antes desenvolvedor poder dar um merge, ou fazer o chip do código. Tem isso não é, tem o verificador de sintaxe, os testes unitários e teste de integração e dependendo do que é alterado é feito um teste manual também antes de ir para produção.

P. Passando para o processo de manutenção do código de teste, como ele é realizado equipe? Edição, exclusão de teste.

E. Cada vez que a gente cria. A gente trabalha aqui com o git, com pull request. Então tu cobre com o teste. Então se tu tirou uma funcionalidade, tu pode tirar o teste. Os testes que cobriam aquela funcionalidade. Agora não costuma tirar teste só porque ele está falhando. Isso não é uma prática aqui. Então cada um é responsável por adicionar os testes da funcionalidade ou por remover alguma coisa que foi removido de funcionalidade também. E nunca remover sem ser por outro motivo ou tu perde cobertura.

P. O que você conhece sobre test smell?

E. Test smell? Não sei se eu conheço. Tem alguma definição? Só para eu ter uma ideia. Talvez eu conheça mas não por esse termo.

P. Tudo bem. Eu vou te passar o conceito sobre o assunto...

E. Code smell?

P. Na verdade é test smell

E. É a mesma coisa que code smell, não?

P. Ele descende de code smells.

E. Pode explicar então porque talvez seja bom uma definição.

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. Entendi. Test smell que a gente note tem dois tipos principais. Um são os testes que estão flaky, que estão falhando randomicamente, de vez em quando falham, então a gente tenta consertar esse tipo de teste. Pode falhar por exemplo porque tu está testando uma coisa , mas tu não tem o isolamento necessário no banco de dados para fazer com que aquele teste passe sempre. Então a gente sempre tenta consertar. As vezes tu tem que refatorar os testes para que eles não fiquem flaky, falha só de vez em quando. Dá um feedback errado da sua alteração. Então primeiro approach é tentar refatorar o código do teste. Se a gente não consegue resolver a ver que o valor é baixo, nesse caso a gente tira o teste. É algo bem raro. Geralmente fazer refatoração do código resolve. Mas o teste que falha demais, que dá falso feedback é ruim. Até Martin Fowler fala sobre isso, que tu pode tentar isolar o teste primeiro e tirar ele da stack até tu conseguir consertar. E a parte de complexidade do código de teste, a gente encara o código de teste como código de produção. Então de tempos em tempos a gente refatora para diminuir a complexidade do code base de teste.

P. Você acabou respondendo até uma das perguntas que eu ia te perguntar a seguir que é em relação a manutenção, que vocês acabam refatorando o teste e tentando deixar ele mais limpo, mas na questão da criação, no momento que está codificando o teste, existe algum tipo de tratamento em relação a test smell?

E. Na hora que está criando? Não sei se a gente tem alguma coisa na hora de criar. Na verdade, eu esqueci de mencionar que tem o pull review, pull request review, então durante a criação, todo o time pode dar o feedback antes das alterações serem para a master. Então se tem algum code smell ali no teste ou o teste está muito complexo geralmente é dado o feedback por alguma terceira pessoa. Uma segunda pessoa no caso. A gente não costuma a fazer merge sem ter um reviewer pelo menos, seria o lugar onde a gente pegaria um test smell, por exemplo. Claro, isso se quem criou não viu, não é, ou não percebeu.

P. Só um detalhe: No seu caso, o code review é feito por uma pessoa só.

E. No mínimo uma pessoa. Tem o time, ninguém faz merge sem ter pelo menos uma pessoa ter feito o review. Mas assim, vai da razoabilidade. Tem alterações muito grandes que aí tem que esperar mais um ou dois. É que depende muito da alteração, mas no mínimo 1, porque evita que vá alguma coisa que poderia causar dano, criar falha de segurança, alguma coisa assim.

P. Tranquilo. É que como você não tinha complementado antes...

E. É. Eu esqueci de mencionar isso quando estava falando do processo. A gente sempre abre um pull request no repositório do Git e sempre tem uma pessoa revisando também.

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

E. Da criação. Da criação é mais complicado. A gente não tem nada específico, não é! Mas o que poderia ser feito, por exemplo eu estava falando em complexidade de código, é aquele Code Climate tem algumas ferramentas que já verificam complexidade do código no momento da criação, então poderia ser usada uma ferramenta como essa para validação da complexidade em si do código. E daí então teria que refatorar. Nós não temos isso no momento. Eu usaria uma ferramenta de verificação de complexidade mesmo

P. Estou chegando ao final da entrevista. Você tem alguma pergunta para mim, quer acrescentar alguma informação ou tem alguma sugestão para melhorar a entrevista?

E. Não. Gostei da entrevista. Achei bem interessante as perguntas. Tu é mestrando da Universidade Federal da Bahia, não é? Tua área é Engenharia de Software? Qual é tua área de pesquisa?

P. O mestrado em si ele é em Ciência da computação, e lá tem os eixos de pesquisa, e entre eles tem o de Engenharia o de software que é o que eu me interessei mais por eu já ter trabalhado mais de 7 anos como QA, então por eu já ter essa experiência de mercado e aí eu voltei para academia e eu queria continuar ainda nessa área e ai eu fio para Engenharia de software.

E. Ah, que legal. É uma área que muitas empresas não cuidam as vezes da questão de teste de qualidade. Legal estudar isso.

P. E aí dentro da academia eu consegui identificar essa oportunidade de pesquisar a parte de test smell e eu voltei meu olhar novamente para a indústria. Novamente para o mercado para entender como funciona isso dentro das empresas e ver em que ponto a gente pode melhorar esses processos e alguma forma aumentar a qualidade e não manter o conhecimento exclusivamente na academia, fazer com que esse conhecimento de alguma forma chegue na indústria seja pelas entrevistas ou pela solução final que a gente ainda vai desenvolver. Então a ideia é fazer com que esse conhecimento chegue o mais rápido possível para o mercado.

E. Entendi. Que legal. Eu notei que quando eu comecei a uns 10 anos atrás depois que eu saí da faculdade não era muito comum fazer teste unitário, fazer teste de integração. Hoje é um pouco mais comum assim. Desde quando eu mudei de empresa muitas vezes tu encontra já um code base testado, que era uma coisa mais rara lá no início. Acho que devagarinho está mudando a mentalidade dos desenvolvedores. Muito legal tua pesquisa.

P. Eu agradeço muito a sua participação, sua contribuição na pesquisa. Eu vou encerrar aqui a gravação.