Pesquisador. 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 ou desejáveis ​​ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que realmente pensa. Tudo bem?

Entrevistado. Perfeito

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. Tudo bem?

E. Perfeito.

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

E. eu comecei a trabalhar com teste de software em meados de 2009, a partir de uma “aprofundação” minha com a linguagem Ruby na ocasião.

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

E. Eu acho que principalmente muitos posts em blogs de tecnologias relacionados a desenvolvimento de software. Naquela ocasião onde eu comecei eu usava bastante um blog do Aquito on rails, famoso palestrante e organizador de diversas conferências no Brasil, com suporte de livros relacionados a linguagem, que eu pude perceber ao longo do tempo que os próprios livros de linguagens de programação começaram a ter capítulos específicos sobre testes unitários. De uma forma ou outra eu sempre casei tanto a parte de consulta em blogs quanto em livros, um 50/50.

P. Em quais linguagens você cria testes?

E. Normalmente eu escrevo testes em Ruby, que é a linguagem que me sinto mais confortável, em Javascript, Python, já cheguei a fazer uma coisa em Java, mas não muito, e uma coisa ou outra em Go lang que é a linguagem do Google

P. Qual a linguagem do seu projeto atual?

E. atualmente eu tenho trabalhado em projetos que envolvem python, então acaba sendo esse universo phytonista por assim dizer.

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

E. Depende muito da natureza do projeto, existem projetos os quais que eu prefiro trabalhar com o BDD, que também deixa de ser teste unitário, porém é um teste comportamental. Porque que eu prefiro o BDD ao invés de TDD. O TDD te garante cobertura de código. Tem uma maneira bem 1 para 1. Entretanto, o teste comportamental eu consigo fazer simulações de um workflow inteiro. E isso muitas vezes, faço com que eu acabe envolvendo testes de, vamos dizer assim, funções. Porque a gente está falando de classes aqui, estamos falando um pouquinho aqui de orientação a objeto. Eu gosto muito do BDD porque eu consigo testar um overflow inteiro de uma ou mais classes envolvidas na obtenção de um resultado final, e porque que isso é bacana porque só o BDD em si não é o suficiente na maior parte das vezes. Apesar de você ter uma cobertura de código, 95% de teste unitário, falando sobre funcionalidades, requisitos de negócio principalmente, é necessário fazer um teste envolvendo mais componentes, mais classes no caso. E aí, dependendo da complexidade do projeto eu acabo optando naturalmente pelo BDD. E o BDD consegue fazer cobertura um pra um. O ideal é sempre ter na faixa de 90 - 95% de código coberto. independente que seja tdd ou bdd. necessariamente com o norte acaba pendendo para o bdd.que eu consigo fazer teste em overflow completo, no caso.

P. Existe algum fluxograma, template documentado, que padronize esse processo de criação na equipe que você trabalha ou não empresa?

E. Não pelo seguinte. Eu trabalho com infraestrutura com o código. Os meus testes acabam sendo muito pontuais e variam de acordo com a pureza do projeto porque infra estrutura depende de um cliente para o outro. Então não existe um processo padrão, ou formas padrão para abordagem. Isso eu falo do meu trabalho em si enquanto em engenheiro devops agora para quem trabalha num time de backend, precisa sim ter algo mais consolidado ao qual eu não me envolvo, porque vamos dizer assim, é outro time dando conta.

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

E. eu normalmente utilizo behave em python, para ruby, eu normalmente utilizo o elt spec. Javascript acaba sendo o jest no caso.

P. Como é o processo de verificação de qualidade do código de teste criado, por exemplo: verificação de escrita, algum desenvolvedor revisa o teste depois de criado, verificação de compatibilidade com o requisito?

E. Normalmente o que se trabalha são com code reviews, onde o pessoal... estou falando de um workflow baseado em git, normalmente o pessoal abre um pull request onde é solicitada a aprovação de um ou mais pessoas dentro do time, onde normalmente esse código ele passa por uma avaliação de qualidade para ver se aquele código é minimamente suficiente para cobrir aquele teste, e também não tem algumas implementações, coisas que levem por exemplo a complexidade exponencial de algoritmo, esse tipo de coisa. Existe sempre essa avaliação por parte do time onde esse código relacionado aos testes é validado.

P. Quantas pessoas fazem essa verificação de um código submetido?

E. Normalmente tem uma pessoa ao qual valida o código, muito mais para que ela entenda como funciona esse workflow e obviamente ela acaba validando, mas há de se considerar que meu time é pequeno, tem atualmente uma pessoa no meu time.

P. Passando para o processo de manutenção do código de teste, como ele ocorre dentro da equipe ou da empresa, se for um padrão?

E. No caso de infraestrutura é um pouquinho diferente do habitual, falando de manutenção de software porque infraestrutura ela tende a ser estável. Consequentemente o teste ele é escrito muito na criação, consequente na manutenção, nesse tempo todo na empresa não houve necessidade de fazer manutenção nesse código devido a natureza de infraestrutura que tende a não mudar.

P. O que você conhece sobre test smell?

E. Test smell já significa algum código que não está muito de acordo com boas práticas de programação, enfim, pode ser por conceito de anti pattern ou simplesmente um design do código que não está apropriado para execução daquele teste.

P. Como você conheceu este assunto?

E. Entrou meio que naturalmente quando você faz um aprofundamento em testes unitários, porque da mesma maneira, digamos que ele também é código e código ele também é sujeito ao que é bom e o que é ruim. Então eu acho que no momento que você se aprofunda sobre teste de código você acaba encontrando anti-pattern, afinal de contas, até mesmo pelos próprios testes você acaba encontrando anti-pattern no seu código, e mas nada te garante também que você não vá colocar um anti-pattern no seu código de teste. Eu acho que uma coisa puxa a outra de certa forma.

P. Possui alguma dúvida sobre esse tema?

E. Não. No momento presente não tenho dúvidas.

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

E. Sempre que possível durante o PR esses smells são detectados em código antes mesmo daquela subestação de pull request ser mergeada no branch master. Então é responsabilidade do time que está desenvolvendo aquele código ali evitar isso via review sempre que possível e o quanto antes melhor.

P. A outra pergunta tinha a ver com manutenção, mas como você mencionou atualmente você não tem essa atividade no dia a dia.

E. É até porque como eu havia citado, desenvolvimento de infraestrutura acaba tendo um universo muito estável comparado com desenvolvimento de sistemas porque normalmente quando a gente monta a infraestrutura é uma vez em definitivo. Ela não tende a mudar. Simplesmente por isso esse tema de manutenção não é tão abrangente na minha situação agora. Quando eu era programador, líder técnico em outra oportunidade sim, nós tínhamos todo um ferramental ali envolvendo Jenkins etc e tal para que a gente pudesse fazer o tracking de manutenção especialmente de testes porque o codebase sempre evoluía.

P. Para você, como é possível prevenir ou evitar a inserção de test smell no código de teste no momento da sua codificação, no momento que esse código é criado?

E. Olha, no momento de criação eu acredito que depende muito da senioridade de quem está escrevendo. A tendência é muito mais juniores colocarem test smell, é por isso que baseado no workflow especialmente de pull request é muito importante esse processo de feedback antes mesmo de fazer qualquer merge em código, no que seria o branch principal, porque pela estrutura de pull request, pelo workflow natural de um pull request por exemplo do github você submete, pede aprovação para as pessoas e uma ou mais pessoas podem revisar aquele código, identificar possíveis problemas naquele teste unitário. Nesse sentido que eu digo que o pull request ele é muito válido porque você vai, submete seu patch dependendo de como for uma ou mais pessoas aprovam mas sempre pode haver o caso de uma ou outra colocar indicações ou simplesmente pedir para corrigir determinados trechos de código. Se caso detecte, no caso, um test smell.

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

E. Eu acho que por hora não tenho perguntas não, está bem concisa.

P. Muito obrigado por ter aceito participar da entrevista. Ela com certeza vai ajudar bastante na minha pesquisa do mestrado.

E. Perfeito

P. Assim que a gente tiver resultados da pesquisa a gente vai compartilhar com quem participou dela, então em breve, assim que a gente tiver os resultados mais consolidados a gente vai compartilhar com vocês.

E. Tranquilo.

P. Eu vou parar aqui a gravação.