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. Ok?

Entrevistado. Está bom.

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. Ok?

E. beleza, beleza

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

E. Eu comecei a trabalhar em 2008 com testes, com teste manual e fui aprofundando o conhecimento até agora.

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

E. Eu estudo bastante ecleticamente, eu sou muito eclético nos estudos. Então eu busco em várias fontes. O que eu mais posso dizer que aprendi sobre testes foi o aprendizado empírico. Tem eventos, meetups de qualidade, a convivência com pessoas que são bastante destacadas nessa área que eu trabalhei com grandes especialistas de qualidade. Tratamento estratégico de qualidade das empresas, eu já vi muitas coisas acontecerem em qualidade. Tive a oportunidade de estudar case de grandes empresas que são destaques para a gente, como Netflix, Facebook, Google e tudo mais pelas empresas que eu participei e por empresas mais antigas que faziam diferentes e tinham problema, porque não basta ver o que funciona, você tem que ver também o que não funciona para você conseguir comparar, então empresas como o próprio empregador que eu tenho hoje que é a empresa B, eu tive contato com empresas concorrentes, então eu trabalhei com contas como a empresa P do Peru, empresas de telefonia como a empresa Q, a empresa Q do Peru também. A empresa R da Colômbia, quando eu atual por outra empresa, então o conhecimento ele aparece de todo lado e eu sempre procurei estar atento para conseguir ter uma visão empírica da qualidade. Empírica e holística.

P. Em quais linguagens você cria testes?

E. Eu já trabalhei com muitas linguagens, desde VB script no começo. Até agora tenho trabalhado com Java, Ruby, C# e as linguagens de dispositivos móveis que é Android, IOS, que tem o Kotlin, Swift e que eu trabalhei da Microsoft que é para ambas as plataformas, não lembro o nome agora, mas é uma linguagem para mobile também

P. Qual linguagem do seu projeto atual?

E. Eu não estou trabalhando com desenvolvimento, estou numa parte mais estratégica, mas o último projeto que eu trabalhei foi em Java.

P. Como é o processo de criação de teste?

E. Nós temos 2 tipos de teste unitários que a gente chama, para facilitar nossa criação. O teste unitário objetivo e o não objetivo. O teste unitário objetivo é aquele que a gente está intrinsecamente ligado aos requisitos de negócio. Então imagina que o P.O. escreve os requisitos de negócio e ele escreve todos os critérios de aceite. A gente vai escrever todos os testes que a gente imagina para aquela demanda. Então a gente escreve livremente, faz uma escrita de todos aqueles testes e a gente distribui eles usando uma modelagem que chama pirâmide de testes, a gente distribui eles dentro dos níveis dessa pirâmide. Geralmente essa pirâmide é dividida em 3 partes: unit, services e guee, que é tela, serviços e unidade. A gente procura sempre pela camada mais baixa que é mais rápida e objetiva. Que consegue rodar mais rápido e consegue encontrar defeitos antes no ciclo de vida de desenvolvimento, então a gente sempre procura focar em fazer esses testes na unidade. A gente faz uma listagem e coloca esses caras nos testes unitários e o desenvolvedor usa esses testes que a gente modelou como TDD. Primeiro ele faz os testes e depois ele começa a programar o código da aplicação. Feito isso a gente tem os testes unitários não objetivos. A gente terminando o projeto a gente roda uma ferramenta para medir a cobertura como o Jacoco por exemplo e ele vai retornar quantos por cento está coberto de linhas de código desenvolvidas e a gente vai alcançar o nível de cobertura de 80% das linhas de código e posteriormente a gente roda uma outra ferramenta que chama teste mutante que é o pi teste. Esse pi teste ele faz uma análise qualitativa desses testes. Ele vai e altera o operador do if, por exemplo se era X > 0 ele coloca X < 0 e aí ele roda os testes unitários e vê se algum teste falha. Se o teste falhar ele declara que o mutante morreu, e se o teste não falhar ele declara que o teste mutante não está morto. E aí ele faz diversas mutações nesse código, ele tira partes do código, tira chamadas de métodos, muda a declaração de variáveis, entre ouras coisas e vai validando aquela suíte para verificar se ela está testando adequadamente o software.

P. Esses testes quando você roda o pi teste, eles são mantidos ou são descartados depois?

E. a gente cria, aqueles que a gente entende que são factíveis. Tem alguns testes que o pi testes cria que não vê tanto valor porque aquilo não é factível quando a gente cria o código, mas a ideia é que a gente tenha uma cobertura maior de teste mutante dentro da esteira. Então a esteira que roda, ela roda na máquina do desenvolvedor, mas roda na esteira também. Então a ideia é que cada vez que a gente rode na esteira essa análise de mutação a gente tenha um aumento da cobertura de mutação. A gente está procurando aumentar. Esses testes de mutação geralmente são uma boa parte deles incluída na nossa suíte, mas não é uma obrigação. Se a gente vê que não faz sentido a gente tira. Não vamos colocar cobertura por colocar.

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

E. Não.. Essa é uma estratégia que a gente passa para o pessoal como boa prática e tudo mais, mas a gente acredita muito na evolução de cada squad. A gente faz o treinamento e mostra o valor do teste unitário, como isso pode fazer a diferença dentro do ciclo de vida do desenvolvimento, que a gente encontra defeito antes, e tudo mais. A gente faz um treinamento de mindset, mas a gente acredita muito em PDCA. A gente deixa os times muito livres para eles fazerem a evolução dessa prática de teste unitário e colocarem outros passos que eles entendem que vá ajudar ou facilitar a vida deles e tudo mais. A gente tem uma sugestão, um template, mas a gente entende que os times vão trabalhar numa evolução desse template numa versão que seja melhor para eles.

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

E. aqui, como é uma empresa grande podem ter todas as ferramentas que você possa imaginar. As que eu usei, para teste unitário é o próprio Jacoco no Java, o JUnit, o Phytest e a gente usava o sonar para análise estática, mas a gente colocava os dados que a gente conseguia coletar da central de qualidade.

P. Como é o processo de verificação de qualidade do código de teste criado, por exemplo: verificação de escrita, existe um code review, alguém revisar o teste depois de criado, a compatibilidade com o requisito e assim por diante?

E. A gente tem como eu já falei a gente tem os testes mutantes e a medida cobertura de linhas de código de teste, a gente tem um code review pré merge, que um outro desenvolvedor vai analisar esse código desenvolvido e vai dar um parecer para a gente aceitar ou não o merge ou fazer sugestão para ser alterada e a gente tem a análise estática pela ferramenta.

P. quantas pessoas fazem a verificação de um código submetido.

E. geralmente a gente faz um peer. Duas pessoas para validar esse código e trocar opiniões. é o desenvolvedor e 2 pessoas que vão fazer essa validação. Aí eles dão o veredito em conjunto, mas isso depende muito do tamanho da demanda. Tem vezes que a gente precisa colocar mais pessoas fazer a validação, mas duas é o mínimo.

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

E. A gente costuma usar esses métodos de validação de cobertura tanto da qualidade mutante quanto do código, e também como nossa suíte ela é executada sempre. A gente tem domínio de teste falhando e tudo mais. Isso é gate na nossa esteira, então a gente sempre está sabendo das coisas e no nível dos nossos testes unitários e tudo mais. E a manutenção é feita durante as demandas, quando você faz a alteração no código, você tem que fazer uma alteração no código de teste também e a gente tem uma equipe que faz esse merge e essa equipe submete as solicitações do código de teste que ela ache necessários. Então imagina que ela começa a encontrar muito defeito naquela parte do código, ela vai solicitar para que sejam incluídos os testes ou ela vai solicitar para que sejam alterados esses testes, dependendo de como a qualidade daquela parte está sendo percebida durante o dia a dia daqueles caras que fazem o merge e coloca os aplicativos na loja.

P. O que você conhece sobre test smell?

E. eu não conheço.

P. Eu vou te passar o conceito sobre o assunto, vou te passar também 2 exemplos, porque as perguntas a seguir têm a ver com esse tópico, ok?

E. Está bom.

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 legal, entendido

P, eu vou te passar também 2 exemplos e a gente conversa um pouco sobre eles. O primeiro que deve ter chegado para você é o chamado Empty test smell. Ele é considerado um smell porque não tem nenhum script executável. Então fica mais difícil de identificar o porque ele está comentado ao invés de ter sido excluído, então ele simplesmente não vai executar o que ele foi proposto ao ser escrito. O segundo é o chamado Conditional Test Logic smell. Ele é considerado um smell porque ele tem uma estrutura de repetição, condicional dentro dele. Então uma vez que esse teste falhe e mais difícil de identificar o porque que esse teste falhou. São testes que de acordo com a forma que ele for escrito ele pode ter um impacto negativo na atividade de testes.

E. Legal. Sobre isso, eu acho que a gente está muito relacionado a arquitetura que a gente desenvolve o nosso código, porque muitas vezes a gente tem que fazer testes complexos porque o código é complexo. Então a gente procura sempre estar modularizando esses testes e quando a gente nota que esses testes vão ficar muito complexos a gente tira de dentro do código e coloca como testes de serviço, porque eles vão ser mais rápidos e mais baratos de fazer. Agora, a nossa ideia para que isso não aconteça é usar das nossas práticas que a gente tem. A gente entende que um mutante ele vai ajudar nisso porque se você tem teste que não testa nada a gente provavelmente vai ter um mutante vivo e a gente também acredita muito no code review e nas práticas que a gente vem trabalhando de evolução de teste unitário que é para a gente o tipo de teste mais importante, que nós temos hoje. Então se você mandar um teste que seja muito complexo para sala que vai fazer o merge dessas features, provavelmente isso vai de alguma forma ser rejeitado e corrigido, porque uma vez que você vai desenvolver, mais pessoas vão ter que achar uma decisão junto de como fazer. Eu não posso dizer que a gente não tem esse tipo de problema. A gente só iniciou essa nossa estratégia de qualidade porque a gente tem muitos problemas desse tipo, então é uma coisa que a gente nota muito que o nosso teste unitário ele praticamente não existia antes da gente começar a fazer isso e acarretava num problema gigantesco de você ter ciclos de teste que demoravam um mês o ágil isso não é cabível. Então por isso a gente começou a tentar encontrar essas soluções. Para minimizar esses testes unitários que não testam nada ou que são inviáveis. Então a gente está contando com o pessoal de arquitetura para ajudar também.

P. Você acabou respondendo praticamente as próximas duas perguntas, mas só ratificando, esses dois processos que você acenou me passando de prevenção, ocorrem tanto na criação quanto na manutenção do teste de unidade, confere?

E. Isso.

P. Em algum dos dois processos tem alguma etapa adicional, tem algum processo que é feito diferente ou é igual mesmo?

E. É igual, a gente atua. Os nossos desenvolvimentos eles geralmente são sempre manutenções, porque de alguma forma você vai ter que alterar alguma coisa, seja para o código novo você espera que não tenha alguma alteração de cobertura, mutar célula e tudo mais. Ele cria o teste novo e a gente vai fazer uma validação da mesma forma.

P. Para você, como seria possível prevenir ou evitar a inserção de test smell no código de teste no momento da criação, ou seja, quando o desenvolvedor está codificando, como seria possível prevenir test smells nesse momento?

E. Eu entendo que isso aí passa muito por esteira e pós commit. A gente vem falando muito sobre isso. O que acontece com o teste é que parece que eles ficaram para trás dessa virada para o ágil. Muita gente deixou teste para lá e tudo é ágil menos a qualidade. A gente entende que se a gente não tem o feedback para o desenvolvedor que seja no mínimo diário, a gente quer que seja a cada commit, mas se a gente não tem condições por questões de performance, a gente colocar isso em cada commit, a gente roda a cada noite, a gente roda as ferramentas não como gates, mas como feedback para os desenvolvedores. Então a gente rodaria essa esteira a noite que a gente chama de nightly build, a gente roda essas ferramentas, essas validações e a gente manda para um chat para o desenvolvedor na hora que ele chegar de manhã ele notar que está cometendo esses erros, e ele já vai saber que quando ele chegar para fazer o merge desse código que ele está desenvolvendo, que ele vai ter que prestar contas quanto a esse problema. Aí a única coisa que ficaria faltando é você ter o pair programing, que é dois desenvolvedores trabalhando na mesma demanda, que é um processo que a gente tem bastante no banco, que nossas demandas são grandes, mas que não dá para ser em todo lugar. porque tem squad que só tem um desenvolvedor, mas na maioria dos lugares dá. Os dois recebem essas críticas e eles podem trocar ideias e evoluir isso. O engenheiro também que a gente tem. Cada squad tem um engenheiro e dois desenvolvedores pelo menos, geralmente. Tem uma ou outra que não tem dois desenvolvedores. Geralmente tem dois ou mais. Então é assim que a gente vê que eles poderiam encontrar durante a programação em si.

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. Não. Achei muito legal o tema. É um tema em que estou muito, faz 2, 3 anos que estou atuando muito fortemente nisso, que eu tenho muito interesse. Por isso que eu de pronto que eu li eu já aceitei porque é uma coisa que a gente precisa falar mais na área de qualidade. Parece que o teste dele fica focado em teste funcional e eu atuo com teste funcional a 12 anos e sei que o teste funcional não é a melhor opção, é só a última porta. Se você puder quando terminar esse trabalho mandar para a gente para que a gente também possa aprender com essas entrevistas que você fez e conseguir aplicar alguma coisa, ajudar na nossa estratégia aqui. Que eu não falei a fundo mas estou trabalhando aqui no banco como uma área estratégica de qualidade. A gente estuda as coisas, faz pilotos, ou seja testa em alguns lugares e depois a gente aplica no banco inteiro, então a gente está sempre fazendo essas coisas. Então se a gente puder aprender mais e mais vai ser melhor para a gente.

P. A pesquisa está terminando essa fase agora de entrevistas, a gente vai depois analisar os dados, verificar o que seria de oportunidade de desenvolver uma aplicação, um guia para melhorar a prática, porque minha pesquisa em si é focada em test smell, então por isso que da metade para o final as perguntas são focadas nisso. Então com certeza assim que a gente tiver os resultados em relação a isso a gente vai compartilhar com vocês. Muito obrigado pela sua participação.