Pesquisador. Mais uma vez muito obrigado por participar da entrevista. Ela tem como objetivo compreender os processos de criação e manutenção de código de teste de unidade e também verificar como ocorre a verificação da 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 pensa. 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. Ok.

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

E. Faz uns quatro anos.

P. Na empresa anterior?

E. Foi. Em duas empresas anteriores para traz.

P. E se tratando de código de teste de unidade, quais foram suas fontes de aprendizado?

E. Inicialmente com colegas de trabalho, depois eu fiz cursos em uma plataforma paga chamada Plural site.

P. Em quais linguagens você cria testes?

E. Já criei testes em Phyton, em C#, Go, Node e em Java.

P. Qual plataforma do seu projeto atual?

E. Atual Go.

P. Como é o processo de criação do código de teste?

E. Quando eu não tenho a menor ideia da arquitetura da solução, eu costumo a começar a fazer um esboço pelo caminho feliz, então eu tento fazer alguns mocks dentro do código principal mesmo e chegar até o final, e depois eu vou refatorando e fazendo. E nesse processo eu começo a escrever os testes para os componentes. A partir do momento que eu vou “componentizando” eu vou escrevendo o teste. Mas isso depende da linguagem, então no C# por exemplo, eu conseguia fazer uma abordagem TDD que eu escrevia primeiro o teste, escrevia o código até o teste passar, escrevia de novo um teste e ia fazendo esse ciclo que é bem interessante, mas quando estou diante de uma linguagem nova, aprendendo conceito e ainda não tenho ideia da solução eu sinto que é mais simples eu fazer primeiro o código e depois fazer os testes para garantir que o código esteja funcionando conforme esperado. Então dependendo do nível de amadurecimento da linguagem e do projeto e da arquitetura, eu dou preferência para uma abordagem ou para outra. Nesse último projeto, como eu estava aprendendo a linguagem GO, primeiro eu fiz o caminho feliz e quando eu fui fazer os testes eu comecei a estudar melhor, a pesquisar melhor os conceitos da linguagem, de organização de código, aí eu descobri que o código estava bem ruim e mal formatado, o que é natural quando a gente faz o primeiro código em uma linguagem e isso me permitiu estruturar melhor o código, criar algumas interfaces para fazer a comunicação entre pacotes, isolar componentes e aí eu escrevi os testes e durante esse processo eu consegui capturar alguns bugs e fazer algumas melhorias. Então foi dessa forma. E agora com novas funcionalidades, estou querendo implementar uma nova funcionalidade, primeiro eu vou tentar escrever o teste para depois colocar a funcionalidade porque a gora eu já tenho os pacotes organizados. Mas foi o contrário dessa. Primeiro eu fiz um draft, depois fui tentar escrever teste, não consegui, refatorei o código, escrevi os testes e fui melhorando.

P. Existia algum fluxograma, template... que padronize esse processo de criação de teste unitário?

E. Não porque nessa empresa específica o time é muito novo. A empresa é muito nova, nasceu ano passado, eu entrei faz 3 meses, então a gente está escrevendo os processos como vai ser daqui para frente. É uma startup, então é um momento bem de definição. Não tem nada escrito em pedra e a gente está montando para ver o que é melhor. Então tem muito escolha de ferramenta, tem muito projeto novo que precisa desenvolver e escrever arquitetura. Estamos montando o time de desenvolvimento, então tem muita coisa sendo definida. O processo de desenvolvimento ainda não está amadurecido. A gente começou a fazer à medida que foi tendo necessidades e focando nos problemas mais imediatos. Então a gente não tem um processo pré-estabelecido de como fazer o desenvolvimento ainda.

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

E. Depende da linguagem. No caso do Go ele já tem uma suíte nativa para testes, inclusive para medição de cobertura. Ele já tem tudo pronto. Está dentro da biblioteca Gotests. No caso do Java eu usava JUnit. No caso do C# eu usei o NUnit e o XUnit para o final. O XUnit eu achei um pouco melhor, é menos verboso para utilizar tem alguns conceitos de sempre iniciar classe para passar de um teste para o outro e isso é interessante. De ferramenta para rodar o teste são essas. No Phyton tem o um nativo, não lembro. Faz um tempo.

P. Dentro da empresa, como é o processo de verificação de qualidade do código de teste criado? Por exemplo, verificação de escrita, se algum desenvolvedor revisa o código feito por outro... como a qualidade de teste é feita?

E. A gente tem o conceito de utilizar o code review, para ter uma verificação em par. Então sempre que é feito uma feature ou um bug fiexed, é feito um merge request para revisão de pares. Então 2 ou 3 pessoas revisam o código, esse código volta com os comentários e o desenvolvedor resolve os comentários, é feito um double check em tudo e... perdão. Depois que o desenvolvedor termina, quando ele "commita" é disparado de maneira automatizada no git lab, é rodado os testes anteriores, então é verificado se não quebrou nenhum teste anterior e é rodado também o SonarQube em cima do código escrito. Se há algum problema desde fazer o build ou se dá alguma redução na cobertura ou se há algum problema o build já não passa e o processo não vai para frente. Se o processo da espera funcionou, aí sim ele vai para um merge request e é feita uma revisão por pares. Então quando o par revisa ele já sabe que os testes rodarão que não houve problema de regressão nos testes já escritos e ele faz uma revisão tendo essa tranquilidade. Depois que é feita essa revisão ele vai para o branch principal e é rodado novamente o build, o SonarQube, rodado todos os testes unitários e de integração e é feito o empacotamento do produto.

P. Passando para o processo de manutenção do código de teste, me conte como é este processo na empresa?

E. Temos uma separação do código principal e do código de teste. Se a gente faz uma mudança no código principal, o código de teste vai quebrar com certeza e ele vai ter que ser alterado. Então assim que vc “commita” é disparado todos os testes que estão escritos. Se tiver alguma regressão, o commit não passa e a cobertura também não pode cair. Naturalmente o próprio processo impede que você não mantenha os códigos de teste.

P. O que você conhece sobre test smell?

E. Smell é cheiro ruim em inglês não é?

P. Isso.

E. Eu conheço o conceito de code smell. Tem esse conceito de test smell?

P. Isso

E. Test smell eu não conheço o conceito. Code smell é problema que a ferramenta de análise de teste estática encontra no programa. Test smell seria essa mesma análise em cima do código de teste?

P. Exato. Te passando o conceito de uma forma formal “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.

P. Então basicamente a analogia que você fez. É o mesmo princípio do code smell, só que voltado a parte de código de teste. Isso na verdade foi o processo que ocorreu. Os test smells eles descenderam dos code smells pela história.

E. Que legal. Como funciona o processo de amadurecer meus próprios testes eu comecei ver alguns pontos que poderia melhorar. Eu olhava para alguns testes que tinha escrito no passado, quando eu ia revisar que eu achava alguns problemas que me incomodavam um pouco. Eu comecei a descobrir depois de fazer muito isso. Que eu vi que não agregava tanto. Por exemplo ficar checando se certas mensagens de log foram escritas. Porque às vezes você tinha uma função que não retornava nada mas executava algumas coisas e parte das coisas que ela executava era chamar outros componentes. Então que você escreve o teste unitário você mockava os componentes que ela chamava e ficava sem ter o que fazer o assert na hora que chamar essa função, porque você chama ela para rodar ela não retorna nada. Aí vc mocka, você sabe que ela chamou tais e tais componentes e na hora de fazer o assert às vezes eu ia lá e fazia um assert para ver se o log foi chamado ou tal mensagem de log foi escrita. Depois a gente ia melhorar a mensagem, depois tinha que alterar o teste só porque acertou um comentário no log e depois isso ficava me incomodando durante a manutenção. Eu comecei a remover todos esses asserts de mensagens tipadas do log e quando era uma mensagem, alguma coisa que tinha que acontecer, logar um erro, alguma coisa. Em vez de procurar um erro específico na mensagem logada eu simplesmente falava que a mensagem logada tinha que conter a palavra “error” ou alguma palavra chave que indicava alguma situação ou que simplesmente que o método de log foi chamado e executado alguma vez pelo menos para falar que registrou aquele problema, então eu acabei evoluindo algumas coisas nesse sentido de como escrever testes, mas você forçar com que o teste ser alterado somente quando o comportamento efetivo do seu sistema é alterado e não quando uma “boberinha”, um comentário na escrita de um output faça você ter esse trabalho de alterar o código. sabe?

P. Sim sim. Inclusive esse comentário que você fez tem a ver com um tipo de test smell, chamado redundant print. Você faz algum tipo de impressão sem ter a devida necessidade. Existem outros tipos de smell relacionados a estrutura, mas isso foi um aprendizado que você teve individual que você teve, foi aprimorando o seu processo de criação, manutenção dos testes, não é isso?

E. É.

P. Esse cuidado é um processo seu na sua atividade de criação. No contexto geral da equipe, você consegue dizer se é um cuidado, o tratamento de test smell na criaçao do código de teste?

E. Eu não conheço muito bem o trabalho da equipe porque a gente está numa equipe nova, mas do meu trabalho anterior era uma preocupação minha e de mais uma pessoa. O resto não tinha esse feeling.

P. No momento que você faz a revisão, numa percepção sua, você acaba identificando alguma questão de estrutura que seria considerada como test smell?

E. Agora que você me deu o conceito, nesse caso sim. Do exemplo que eu te falei de pegar uma repetição, dos logs, a gente acaba pedindo para eles tirarem e explica: "isso que você está criando um problema. Depois você faz uma alteração ali vai ter que alterar um teste e que não foi uma alteração de verdade, foi só uma mudança no que está sendo escrito, então dá uma melhorada aqui para você deixar ele mais genérico". É só um exemplo. Mas é bem pouco. Deve ter muito mais conceitos interessantes no test smell que a gente deixa de abordar por não conhecer o conceito.

P. Nesse caso, fica a critério de cada revisor?

E. Isso, a gente pensa muito nesse sentido de ir crescendo com os comentários que a gente vai revisando. Então uma pessoa tem um comentário no seu código sobre um “refactore” que pode otimizar, aí a gente espera que como ele sofreu esse comentário e ele otimizou, quando ele for encontrar esse problema em outro código que ele tenha a mesma criticidade e aponte isso e que todo mundo vá aprendendo com os outros.

P. Só para ratificar, isso funciona tanto para a criação quanto para a manutenção do código de teste?

E. Sim.

P. Para você, como seria possível prevenir/evitar a inserção de test smell no código de teste no momento da criação desse código? Ou seja, evitar que o desenvolvedor no momento que ele esteja criando esse código de teste ele acabe inserindo um smell?

E. Eu uso muito ferramentas de suporte ao desenvolvimento que ajudam a gente a escrever melhor. E eu acho que elas têm resultados muito bons. Um exemplo é o Resharper para o C#. Ele ajuda muito a escrever código melhor porque ele dá essas sugestões. Não especificamente no test smell, mas no code smell, mas como eu te falei a gente não tem essa segregação do que é código de teste do que é código normal, em termos de manutenção. Porque a gente sempre trata tudo junto. Então eu acredito que essas ferramentas, não sei a palavra em português, mas de lint, que elas grifam potenciais problemas, você coloca o mouse em cima e ela explica porque ela está grifando aquilo e qual que é o problema da estrutura. Eu acredito que esse tipo de ferramenta possa ajudar a reduzir isso.

P. Por curiosidade. No Go, a linguagem que você está utilizando agora tem alguma ferramenta que faz isso, que facilita isso?

E. tem, o GoLench deve ter coisas melhores porque ele é feito pela Jet Brains que é a empresa que fez o Resharper para o C# mas estou usando o Visual Studio code e ele tem um conjunto de plugins que você instala para trabalhar para o Go e um desses plugins tem essa questão do Lint ele ajuda a remover alguns problemas. E o próprio Go não deixa você escrever código com problemas mais óbivios. Ele não deixa você declarar uma variável e não usar, por exemplo. Então eu acho que como ele é uma linguagem mais nova ele já nasceu pensando em alguns problemas com mais atenção. Então por padrão ele já tenta ajudar nisso e eu acho que a tendência é evoluir mesmo.

P. Você tem alguma pergunta para mim ou quer acrescentar alguma informação ou ainda tem alguma sugestão para melhorar nossa entrevista?

E. Eu acho que eu estar nessa empresa nova eu deixei de apontar algumas coisas interessantes que eu tinha da empresa anterior. Por conta de focar na empresa nova. Então talvez considerar experiência mais significativa do entrevistado, você pegaria mais informações. A minha maior experiência com desenvolvimento, com teste é com C#, mas não é o que estou fazendo agora. Estou trabalhando com Node e com Go e estou aprendendo esses dois então eu tenho muito menos a contribuir com essa abordagem atual do que com os últimos 3 anos que foi em cima de C#. Mas também não é uma crítica nem nada. Você tem que definir a metodologia e seguir para todo mundo, senão como você vai fazer a separação das entrevistas, né?

P. A sua sugestão é boa. O nosso foco dentro da entrevista é dentro do que está sendo feito no momento, porque por exemplo. Se eu precisar no futuro realizar algum tipo de experimento ou alguma outra prática, o ideal é fazer com base no que está sendo feito agora. Então por isso que o foco tem sido na experiência atual do profissional.. Mas considerando que você me relatou e no que você tem mais experiência, esses processos mudam bastante, por exemplo... você comentou que na sua criação você ao depender do que precisa ser desenvolvido, você usa ou o TDD ou você desenvolve o código do software e depois cria o teste. O processo de verificação da qualidade, manutenção do código de teste é feito sempre através da necessidade da alteração do código de software e é feito através do code review. Isso varia muito de acordo com o que você tem mais experiência?

E. Só varia de acordo com o TDD. Se eu tivesse hoje trabalhando em C# eu com certeza usaria o TDD porque eu já teria na cabeça desde o início do projeto toda a estrutura as classes que eu ia montar então não teria medo de começar com o TDD de ter que apagar um monte de classes depois jogar fora. Agora como eu estou trabalhando com Go que eu não conheço e estou aprendendo, eu pensei em não começar com o TDD para justamente não ter problema de jogar muito código fora e perder muito tempo, só que faz muito mais sentido escrever teste antes do que depois porque até não tem nem graça. É muito mais legal você fazer o teste depois escrever o código. Fazer o código passar no teste. Mas tem essa questão de por estar num ambiente novo, você aquele V0 primeiro sem teste para você entender, desmembrar, desbravar a situação. Agora que está um pouco mais claro eu posso voltar para abordagem mais natural que seria fazer com TDD.

P. Uma outra pergunta por curiosidade. Usando sua maior experiência, no caso em C#. Na empresa que você trabalhava antes tinha algum processo definido, documentado em relação a isso?

E. Eu escrevi esse processo. Pedi que o pessoal seguisse mas muita gente não seguia. Tem sempre a galera de produção, que está aparando os problemas de produção e eu estava em projeto, em arquitetura, então era um mundo diferente, então o pessoal nem tinha tempo para escrever os testes e tinha que passar o dia resolvendo problema e a gente tentava correr atrás disso daí.

P. Era um processo para o time ou para empresa toda?

E. Para o time. Porque era um banco. Você tinha um setorzão dentro do banco então era aquele time lá. A empresa toda não era só uma empresa de desenvolvimento, então era para todo o time de desenvolvimento dentro do banco.

P. Muito obrigado. Todos os seus comentários foram bastante relevantes.