Pesquisador. Muito obrigado por ter aceito participar da entrevista. Ela tem como objetivo compreender os processos de criação, de 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.

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

E. Beleza

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

E. Eu comecei a trabalhar com teste de software em mais ou menos 2015 quando eu fui trabalhar na prefeitura e aí, assim, eu já tinha um entendimento prévio sobre teste de software a parte do processo da faculdade, mas eu não tinha aplicado ainda no mercado, então, sei lá, digamos que 2013. 2014 eu conheci teste de software, li, entendi um pouco do projeto e fui aplicar em 2015 para 2016 nesse trabalho que eu tive na prefeitura da cidade. Mas eu considero que a maior parte do conhecimento de teste de software venha desses dois anos que eu estou aqui como consultor na empresa N porque embora eu tive contato lá atrás, agora eu acho que esse conhecimento é extremamente mais consolidado. Agora eu conheço, acredito eu, que conheço de fato teste de software. Lá era só aplicação ad hoc.

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

E. Minhas fontes de aprendizado, você diz anteriormente ou agora?

P. Os dois.

E. desde o início eu sempre busquei pedir para que outra pessoa me passasse o conhecimento em relação a teste de software, porque seria meu source of thruth. Tipo, a pessoa aplicando ali, no dia a dia acaba sendo mais fácil do ponto de vista de evitar um caminho grande que a gente tenha que fazer e aplicar o teste de software de uma forma que seja bem voltado para o negócio, bem voltado para o problema que estou buscando resolver, mas paralelamente a isso geralmente eu gosto de consultar muita documentação da linguagem, especificamente e eu gosto muito de ler, então o próprio livro desenvolvimento baseado em teste, em TDD, TDD na prática, o TDD do Kent Back são os 2 livros que foram para mim um norte. De outro lado, às vezes tem um livro especifico de uma linguagem, por exemplo Phyton, hoje eu programo muito em Phyton, então vou entender muito como funciona teste no contexto de Phyton, mas eu diria que predominantemente minhas duas fontes de aprendizado é alguém me passar, fazer o processo de pareamento e consequentemente consultar o código, ler o código e a documentação. Depois eu parto para os conceitos, ou em paralelo, vou estar ali observando os conceitos e tentar extrair dos conceitos coisas que posso fazer na prática no dia a dia.

P. Em quais linguagens você cria testes?

E. Eu praticamente crio teste em todas as linguagens que eu estiver trabalhando, obviamente, se a linguagem permitir algum tipo de suporte a isso. Então os cenários são um pouco complicado, se estou escrevendo um besc script que fica um pouco mais difícil de pensar uma maneira de você testar porque talvez você precise escrever outro besc, vai testar esse besc que você escreveu, mas voltando para as linguagens no geral, é mais como a gente fala, utilizar Java. Lá no início trabalhei com teste em PHP, hoje predominantemente eu trabalho com teste em Javascript, na real Phyton primeiramente e depois Javascript. Mas eu já escrevi teste em Groove, já escrevi teste em C++, C#, então eu diria que o conceito eu percebo que é possível de se aplicar em diferentes linguagens mais predominantemente são essas linguagens que eu citei.

P. Qual plataforma do seu projeto atual?

E. Estou trabalhando com Javascript, estou trabalhando com Phyton, tenho feito muito pouco, mas algumas coisas em Groove. Predominantemente essas 3 linguagens.

P. A ferramenta que você trabalha é Web então?

E. A ferramenta que eu trabalho hoje, a aplicação especificamente que eu trabalho hoje é domínio de web sim. Mas tem outras coisas que envolve a web, mas que necessariamente não é restrito a web, tipo processamento de dados, que não é web em si, mas envolve um trabalho básico no backend ou em um cluster de dados por exemplo, e isso é servido depois para web.

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

E. primeiro eu analiso o cenário, aí depende, tem 2 formas: às vezes a história, o projeto que estou ou estive, segue um caminho baseado em BDD, que ajuda você a escrever o teste principalmente de front, mas as vezes a gente tem que fazer pela nossa própria forma, então quando vou iniciar o teste eu analiso os cenários possíveis e vou derivando os testes na mão ou na minha cabeça a partir da prática, se estiver bem confortável, aí depois eu vou criando os métodos de setup, de finalização do teste e depois vou derivando os casos bases. Se eu colocar um caso bem básico, se eu tiver testando uma classe que tenha "carro" por exemplo, que tenha alguns atributos, eu crio o teste, depois faço meu código quebrar, ou passar nesse teste, depois refatoro, executo o teste de novo para ver se o teste está funcionando, aí adiciono mais uma feature, testo de novo, vai quebrar, certamente a ideia é que quebre mesmo, que estou pensando nos “et cases”, depois vou faço passar e aí refatoro, faço improvement daquilo ali. Então basicamente é um ciclo de TDD mesmo, mas eu faço isso para cada método, em cada método aumentando a granularidade para cada atributo e vou derivando todos os casos que sejam possíveis.

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

E. Não, aqui geralmente a gente usa muito o TDD porque, inclusive na aplicação do processo de pareamento, quando as pessoas estão programando juntas ou o TDD sozinho também, mas a gente não tem um processo fechado, a gente tem alguns guidelines que a gente segue, como por exemplo, aplicar TDD, aplicar BDD quando possível, pensar nos “et cases” dos testes. Aí tem uma coisa que rola que é muito legal, que é tipo a gente coloca geralmente, depende do time também, depende do projeto, mas a maioria dos times que passei, a gente coloca os critérios de aceitação de uma tarefa dentro da definição do ticket, do card. Então quando a gente pega o card, aqueles critérios de aceitação ajuda a gente a pensar nos testes. Meio que um processo que não necessariamente é um processo, mas ajuda num contexto de escrever teste.

P. Esse guideline é documentado?

E. Não, como está embutido na nossa cultura, quando a gente inicia um novo projeto a gente tem que ter o teste como uma das coisas primárias, então a gente não tem uma documentação específica. A documentação que existe, que a gente segue é a documentação da linguagem, a documentação de livros, de material para ser consumido em relação ao tema, mas em alguns momentos a gente precisa criar algum tipo de documentação, não pra gente, mas para o cliente, então nesse processo a gente acaba criando uma documentação para ajudar o cliente no entendimento de como ele pode suportar o projeto depois que a gente sair do nosso processo de consultoria e seguindo aquele mesmo padrão de, ou pelo menos quase o mesmo padrão de qualidade que foi embutida a partir da nossa atuação naquele código, mas não tem nenhum, a gente não segue nenhum, segue os guidelines gerais. Não tem nenhuma documentação interna, que seria uma silver bullet para você seguir.

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

E. depende muito da linguagem, então no caso de Java, a gente utilizava e utiliza o JUnit, no caso de Javascript a gente utiliza o Jest, mas aí vai ter alguma coisa específica de teste de end to end a gente vai colocar um Cucumber, a gente vai colocar um net watch, aí vai derivar da linguagem. No caso de Phyton eu utilizo muito o unit test no projeto, mas nos meus projetos pessoais eu, a título de aprendizado também, eu utilizo o Phytest, as vezes a gente pode utilizar algum tipo de ide ou command line. No caso de Java eu utilizava o InteliJ, no caso de Phyton eu utilizo o Phycharm, no caso de Javascritp eu utilizo o VScode, então as IDEs que possam dar algum tipo de suporte, de setup nos testes. Seriam essas ferramentas e engines por assim dizer que estão no meu dia a dia.

P. Como ocorre a verificação de qualidade do código de teste criado, por exemplo: verificação de escrita, alguém revisar o teste depois que ele é criado, como é feita a verificação da qualidade do código?

E. Tem processo de review. Como a gente antecipa e a gente aplica tanto o pareamento quanto o TDD, o processo de revisão já vai acontecendo durante o processo de escrita, mas tem uma outra fase de check up que é levar o teste para outro ambiente, para verificar se ele vai passar também. Por exemplo. A gente aplicando continuous development, deployment, delivery, enfim os Ds da vida, a gente vai codar o teste na nossa máquina, mas depois esse teste vai passar pelo processo de verificação nos outros ambientes. Se tiver alguma dessincronização em relação a esses ambientes, vai haver algum tipo de detecção de anomalia nesse teste por assim dizer que pegue essa verificação que faça essa verificação do teste. Além disso tem um outro tópico que eu acho importante falar, que acho que poucas pessoas tiveram oportunidade, de experiência em trabalhar que acho que importante falar, mas enfim para continuar a abordagem, por a gente aplicar esse processo muito intensamente de pareamento e TDD, evita que tenha um processo de qualidade seja muito necessário para fazer essa verificação, mas ao depender do time a gente tem outro processo que ajuda a garantir, ou aumentar, não necessariamente garantir porque a gente nunca vai garantir, mas aumentar essa observação sobre a qualidade que é review, code review. Por exemplo, eu estou com uma dupla pareando e tem uma outra dupla que está pareando em outra coisa, então a nossa dupla faz o code review da outra dupla e a outra dupla faz o code review do nosso código, então isso é um processo de qualidade que ajuda essa verificação. Alguns, aí retornando ao tema que gostaria de falar, alguns projetos específicos, bem específicos, porque nem todo mundo usa porque tem um custo envolvido, a gente utiliza um processo de teste de mutação. Basicamente, não sei se você tem familiaridade com o tema, mas teste de mutação seria a gente adicionar anomalias no teste para que verificar se a qualidade do nosso teste. Então o teste de mutação testa o teste. A gente fica discutindo, a gente constrói um monitoramento, mas quem testa o monitoramento? A gente escreve o teste, mas quem testa o teste. Então existe esse caminho do teste de mutação. Então por exemplo, sob alguma condicional, a mutação por exemplo, variável é igual a 3, o nosso teste está testando isso, então a mutação em cima dessa condição é adicionar qualquer outra condição, ou várias condições derivadas dessa diferente, tem que ser diferente dessa que verifique se o nosso teste está pegando essa condição, então tipo, para gente aplicar esse tipo de teste é um teste muito custoso, é um teste que leva muito tempo, dependendo do número de mutação que ele crie e o tempo que o teste leve pra rodar, então ficaria impossível de se fazer aplicando TDD gratuitamente, então geralmente o que a gente faz, a gente cria os testes, até porque o teste de mutação é para ajudar o desenvolvedor a aprender a escrever teste melhor inclusive, não é especificamente para sempre ter ali rodando e o desenvolvedor pegar as condições que estão sendo colocadas lá e traduzir no seu teste não. É para que o desenvolvedor pratique o teste, pelo que a gente acredita e depois rode o teste de mutação para tentar detectar alguma anomalia que ele não tenha pensado a cerca. Então o que a gente faz: a gente coloca em uma fase, em um step de desenvolvimento do nosso código. Por exemplo, se estou desenvolvendo um módulo, eu termino de desenvolver o módulo, a história, e antes da entrega da história eu rodo o teste de mutação para ver se ele está detectando alguma anomalia. Se detectar eu volto, faço alguma melhoria, caso contrário, o card, a história do projeto passa. Eu considero como um dos processos melhores, ou um dos mais diferentes de se testar a qualidade de teste. Do ponto e vista de código, porque do ponto de vista de outros tópicos à cerca de qualidade, aí é só tanto o business quanto a experiência do time, dos desenvolvedores mesmo que vai ajudar nesse processo.

P. Em relação ao teste de mutação, vocês realizam a codificação da mutação manualmente ou utilizam alguma ferramenta de geração automática?

E. A gente utiliza uma ferramenta de geração automática. Na verdade todas as linguagens tem suporte para alguma ferramenta, inclusive sempre esqueço o nome da ferramenta, de geração automática, porque ela roda em cima, a gente faz o setup do nosso projeto, do nosso teste, na real, ela roda em cima dos testes, ela roda o teste e consequentemente ela faz a verificação de anomalia, que seriam as mutations, e tem as mutations killed que são as mutations que foram mortas, ou seja, adicionei a mutação mas o teste está cobrindo a mutação então a mutação foi morta, foi assassinada por assim dizer, e tem as mutações que são vivas e são para essas mutações que tenho que observar e fazer uma verificação, um improvement de qualidade do meu código, mas são ferramentas que você seta no teste ou no projeto e depois consequentemente você roda ela.

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

E. A gente não pensa muito sobre manutenção sobre código de teste especificamente, a gente pensa na manutenção do código como um todo. A gente tem um processo que a gente evita a criação de débito técnico, ou seja, que a gente fique olhando muito para manutenção, quer dizer, é um dos fatores que faz que você tenha que olhar um pouco mais para manutenção, mas as vezes é alguma coisa que a aplicação mudou, ou a versão de uma biblioteca, alguma coisa mudou que também faz você observar a manutenção. Como a gente considera que o teste, a documentação é viva, a gente não tem um processo de manutenção, a gente tem um processo em que, a partir do momento que aparecer alguma detecção de anomalia em relação, estou colocando esse conceito para dizer que alguma coisa que não está no nosso esperado, relacionado ao teste que pode estar também relacionado a alguma atualização de código, a gente vai o quanto antes olhar para aquilo ali e atuar naquilo ali. Porque definir um processo de olhar para a manutenção do teste é considerar na nossa opinião, a maior parte das pessoas pensam aqui também, é considerar que a gente está escrevendo algo que não está com tanta qualidade assim, que a gente está sempre observando para aquilo ali. Se você for falar, quem mantém, do ponto de vista da pessoa que tem que observar aquilo, é o time, a qualquer momento, a qualquer horário, se definir um processo específico, mas isso pode definir um step by step, isso é também de cada time, não é da empresa em si. Eu diria que a gente não tem um processo de manutenção definido, mas a gente olha sempre quando necessário em função de alguma anomalia que possa aparecer.

P. O que você conhece sobre test smell?

E. Esse termo para mim não é muito familiar. Você poderia descrever um pouco em pequenas palavras, porque talvez você pode estar falando test smell e pode ser alguma outra coisa que eu conheço mas não necessariamente é a mesma coisa.

P. Eu vou te passar o conceito, te passo também dois exemplos sobre o assunto, até porque as próximas perguntas tem a ver com esse tema. Você pode conhecer mas não com esse nome. “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. Tá. Eu peguei um pouco o conceito. Seria um pouco parecido com os code smells, mas voltado para o contexto de teste, não é?

P. Exato.

E. Eu acredito que test smell é algo que não deve existir. Na prática o que eu conheço sobre padrão de escrita de teste é que segue uma necessidade de se evitar, como a gente considera um teste a documentação viva, a gente evita descrever, não só o teste, mas o código em si, a gente evita escrever teste complexo. Então o que eu quero dizer por isso, testes que tem vários asserts sobre várias coisas, um teste que não está testando uma única coisa, ele está testando muita coisa. Ou então assert que chama, ou uma função de teste que chama outra função que chama 3 asserts que chama outra função que chama 5 asserts, aí você está adicionando complexidade que não deveria existir no seu código, no ponto de vista de conceitos de acoplamento e coesão para dentro do seu teste. O que eu já vi sobre smells relacionados a testes são coisas que inclusive vem dos smells relacionados ao código especificamente, por exemplo. Não ter um setup no teste. As vezes você tem uma classe que você está instanciando ela em diferentes testes, mas você poderia tranquilamente colocar ela em um setup e ter um método for each pra iniciar essa classe que rodaria antes do teste, então before each, antes de cada teste inicie essa classe e depois no final da execução da suíte de teste você colocar um método para, aí você pode colocar uns métodos mágicos da própria suíte para você matar aquela instância ali. Então geralmente são essas coisas que eu vejo. Já vi um código que uma vez trabalhando num projeto que chamar asserts dentro de asserts, um método chamava outro, que chamava outro. São essas coisas que eu vivenciei mais muito na prática, sem muito estar associada a conceitos.

P. eu vou te mandar 2 exemplos de test smells, vou falar um pouco sobre eles. Pelo que eu entendi você já tem um conhecimento, até porque tem a ver sim com essa questão de asserts de complexidade, mas eventualmente depois dos exemplos se você tiver alguma outra dúvida e quiser me perguntar, eu respondo e a gente segue depois com as perguntas da entrevista, ok?

E. Uhum.

P. O primeiro, ele é o chamado Conditional Test Logic smell. Tem a ver com a questão de complexidade que você comentou, porque tem estrutura condicional e de repetição dentro dele. Então uma vez que esse teste fale é mais difícil identificar em qual momento esse teste falhou, por exemplo O segundo ele é o chamado Empty test smell e ele é considerado um smell porque não tem nenhum script executável. Então fica difícil de identificar porque ele está comentado, ao invés de ter sido excluído por exemplo. Se tiver uma quantidade alta de testes dessa forma é possível que a atividade de teste seja abandonada porque está acusando teoricamente que pode na verdade não existir. Então são essas questões que estão relacionadas a estrutura e que prejudicam a atividade de teste em si.

E. Massa. Não conhecia com esse conceito, eu conhecia na prática, mas não conhecia o conceito. Sim, como no código, você escreve um método que não está fazendo nada, não vale a pena manter aquele método no seu code base. De outro lado, se você coloca condicionais dentro do seu teste, significa dizer que talvez essa condicional deveria ser um assert em outro lugar, inclusive deveria ser um teste que deveria rodar antes do que você quer testar mesmo lá em baixo que é o assert que você está fazendo. Aí para evitar esse tipo de coisa, acredito que também é necessário uma familiaridade com alguns conceitos. Na real, o que acontece, o que estou percebendo aqui, é que você tem um assert sob um conjunto de coisas que você dependia antes de fazer.

P. Sim, mas esse é só um exemplo. Tem a ver também com coisas que você já disse antes, um assert chamar outro assert, ou testes dependentes, então você tem um teste e o outro dependendo do resultado do primeiro vai falhar ou não, concorrência de recurso, então você vai ter um teste que vai estar trabalhando com recursos diferentes ou testes que podem ser executados simultaneamente, mas por causa de um recurso um dos dois testes falha, dependendo do que "começou primeiro". Então tem a ver com estrutura do teste e que acabe de alguma forma impactando na independência, na estabilidade, na sustentabilidade desses testes.

E. Sim, mas um ponto que eu gostaria de adicionar é que isso tem a ver talvez com o nível de conhecimento e experiência da pessoa desenvolvedora na escrita do teste, porque o que que acontece: esse assert aqui em baixo ele basicamente o que ele está fazendo, ele está testando um adapter lá, está pegando um item do adapter e testando com result que foi chamado em algum lugar, o que acontece aí, eu dependo nesse assert, inclusive desse for, eu dependo, isso aí, o meu próprio código já esteja fazendo, estou criando isso aí, estou na verdade fazendo vários asserts, então isso aí seria resolvido, mas as linhas anteriores, principalmente seria resolvido com o conhecimento de mock e com conhecimento de stubs. Com o conhecimento esse for aqui na verdade, com o conhecimento dos asserts existentes na linguagem, porque esse assert equal aqui, o que ele está fazendo é testar um por um, mas se você conhece um assert que ele testa multiplos, ele faz o teste de multiplos objetos, não precisamente você precisaria testar um por um. Você poderia testar o próprio objeto SpinnerAdapter, na existência dos multiplos itens que existe dentro dele, então assim, só entender também que tipo, e não sei como isso ajuda na sua pesquisa, que o processo de ida e de escrita de um smell está relacionado ao fato de um não conhecimento da linguagem ou de outros objetos e de outras bibliotecas que podem ser utilizadas no processo de teste que vá ajudar esse processo. Eu confesso a você que esse teste eu acho que conseguiria reduzir ele para colocar ele para escrever em 2 ou 3 linhas no máximo considerando talvez uma boa prática que é você não ter um teste que ele seja, que aqui é como se fosse um long method, uma má prática de um método gigante que você não consegue ler o teste, quem dirá entender talvez depois o que seu código está fazendo. Você está adicionando complexidade de leitura ao teste, talvez o código é tão simples, que o teste não conseguiu reproduzir a simplicidade que é aquele código. Eu acho são esses elementos que são importante trazer.

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

E. Como eu falei, por exemplo, eu não instanciaria nenhum objeto aqui a não ser que fosse um objeto result mesmo, não o spin adapter que estou fazendo o assert especificamente. Eu não instanciaria nenhum objeto que ele dependesse literalmente. Todos os objetos eu mockaria, com isso eu reduziria possibilidades de existência de condicional porque inclusive quando você vai escrever um mock, você pode dizer para aquele mock o que você espera que ele retorne. Por exemplo, se eu tenho na linha event scrap aí, está instanciando um novo event scrap passando Runtime.aplication e answer e eu pego o answers lá de cima. Então o que eu vou fazer, eu vou fazer a minha pilha ao contrário, o meu SpinnerAdapter ele vai de fato existir no meu teste, eu vou instanciar ele, mas o event scrap eu vou mockar ele dizendo que eu espero dele um valor de RuntimeEnvironment.application e um valor de answer. O aswer eu vou mockar ele também na pilha de dependência ali porque o EventsScraper vai precisar talvez que esse valor de answer que esteja mockado em um retorno de valor específico, porque answer pode ser uma coisa mas pode ser outra totalmente diferente, então eu vou mockar answer, dizendo que eu espero dele que ele esteja mockado com um valor de entre value conforme eu quero e de cook manage conforme eu quero. Todos esses mocks serão uma assinatura do meu método e o próprio JUnit permite isso, quanto o PhyTest e outras bibliotecas permitem isso, eles vão ser assinaturas do meu método que eu só vou precisar me preocupar com a instancia de SpinnerAdapter e depois fazer um assert, não assert equal, talvez possa ter esse primeiro assert equals num teste, mas eu vou ter talvez um outro assert equal, e pode ser num outro teste, não necessariamente o mesmo para ficar reduzido, para o teste fazer uma coisa, inclusive o nome desse teste aí poderia ser melhorado, que testaria esses múltiplos objetos que o for está fazendo. Esse seria caminhos que eu executaria no sentido de melhorar, resolver esse smell aí.

P. Ou seja, de uma forma geral, no momento que você cria teste você utiliza conceitos para tornar o teste mais limpo, um clean code...

E. Sim para evitar tanto esse condicional quanto looping e consequentemente complexidade ciclomática associada ao meu teste. O meu teste não pode fazer isso. Result stance of events é algo que eu faço quando eu vou testar, que eu tenho que extrair para um outro assert para testar essa condição específica, se é que eu preciso disso, que eu estou entendendo que é uma condicional que eu também tenha no meu código, então que eu tenha de alguma forma por algum outro modelo de condicional verificar a cerca do meu código, mas se não for, talvez o ´próprio processo de mock pode ajudar, ou stub, pode ajudar na resolução desse tipo de smell.

P. No processo de manutenção do código de teste, por exemplo, você precisou editar um código de software e se deparou com algum teste. É feito algum tipo de tratamento de test smell nesse momento caso você se depare com ele?

E. Só se for um código legado, porque se for um código novo se a gente começar a fazer isso a gente considera que já está errado. Então se for um código legado, a gente verifica como os testes estão sendo escritos e gradualmente a gente vai removendo esse tipo de complexidade do teste. Teste legado é geralmente teste do cliente e a gente vai tentando resolver. Caso contrário não, porque geralmente a gente não segue essa abordagem de escrever esse tipo de coisa relacionada a teste.

P. Quando você fala gradualmente, qual seria a periodicidade disso? Seria de acordo com a demanda da manutenção do código do software?

E. A gente não tem um processo de manutenção definido, então quando algum desenvolvedor vai escrever alguma história e que ele verifica que ele depende daquele teste, ele já olha para aquele teste em certo sentido, aí no tempo dele e na verificação dessa situação ele vai e opera o improvement desse teste. Obviamente fazendo o teste rodar, passar para garantir inclusive que não vai quebrar nada em relação ao que já existe. Continuos Improvement mesmo, mas a gente não tem um processo de manutenção definido. as vezes, a gente tem uma coisa chamada é olhar para os débitos técnicos, e isso pode ter sido registrado no debito técnico Wall, que basicamente lista todos os débitos técnicos, tanto de código, quanto de teste, quando de infraestrutura e aí a gente tira um dia para olhar para os débitos técnicos, porque a gente sabe que eles vão impactar ou no fluxo do dia a dia do desenvolvedor, do time ou pode impactar em algum tipo de anomalia que possa aparecer relacionada ao nosso código, ao nosso produto de software. Então a gente tenta evitar esse tipo de coisa para coisas novas e coisas legadas a gente tenta seguir esses dois caminhos. São processos adhoc, não são nada definidos e cada time acha o que é melhor. Às vezes não é interessante olhar para esse tipo de coisa porque tudo que vai ser escrito vai ser novo e como a gente está no processo de transformação digital e consequentemente aquilo ali, aquele legado, com o passar do tempo vai ser deixado de lado, vai deixar de existir. São essas considerações.

P. Para você, como seria possível prevenir ou evitar a inserção de test smell no código de teste no momento que ele é criado?

E. Um pouco mais sobre o conhecimento dos métodos existentes a engine de teste que você está utilizando, isso já daria várias helps para o desenvolvedor, para a pessoa desenvolvedora. E uma outra coisa seria olhar para esses conceitos: mocks de teste, stub de teste e consequentemente teste de mutação, e fazer TDD e fazer pareamento. E fazer code review. Acho que todos esses elementos são essenciais e importantes para não necessariamente para garantir que não vai ter smell, porque ainda assim possa ter, mas para evitar, reduzir significativamente a quantidade de smell relacionada ao seu code base.

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. Legal. Desejo a você, eu também faço pesquisa de vez em quando, entendo a importância desse tipo de trabalho, parabéns pelo seu trabalho e que você consiga concluir o seu mestrado com êxito. Conheço Ivan, na verdade eu estudei na federal. Estou conectado com as pessoas. Ivan foi meu professor, excelente professor inclusive. Que depois você compartilhe. É uma das coisas que a gente pede. Eu sei que vocês têm um processo de compartilhar cientificamente isso, talvez seja criar um artigo, publicar, isso a gente considera que é uma pratica essencial da ciência, mas as vezes, e por estar no mercado agora, eu trabalhei muito com projeto de pesquisa, na verdade boa parte da experiência da minha vida foi com projeto de pesquisa, que você tem que divulgar isso por outros meios. Escrever no medium, ou fazer palestras em eventos, não necessariamente científicos, porque eu acho que é importante o compartilhamento desse tipo de conhecimento para evitar que a gente se depare com os legados, ou tentar reduzir que a gente se depare com os legados, ou o código com um todo que faça esse tipo de coisa. Acho que é um papel essencial nesse processo.

P. Minha pesquisa ela é focada na indústria, estou pesquisando diretamente com profissionais, justamente para que de alguma forma o resultado da minha pesquisa consiga chegar na indústria de uma forma mais fácil até. Não só os resultados, mas durante esse processo, por exemplo. Estou tocando no assunto de test smell, ontem mesmo estava entrevistando uma pessoa e eu fiz a pergunta, ela abriu uma aba e começou a pesquisar, então de alguma forma a pesquisa já está surtindo algum tipo de efeito e é claro que com os resultados a minha ideia é fazer com que o que está sendo produzido na academia chegue mais rápido no mercado, chegue mais rápido na indústria. Muito obrigado. Não vou ocupar mais seu tempo. Caso você tenha algum contato para me passar, que eu possa realizar a entrevista também eu vou agradecer bastante. A única coisa que eu te peço é que você não passe nenhum dado que a gente conversou para que eu possa ter o mesmo cenário com todo mundo que eu entreviste.

E. Tranquilo. Eu tenho, depois eu passo para você.