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

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. Ok

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

E. Na realidade é meio que a partir de empresas, eu morava em Manaus e as empresas daqui de São Paulo principalmente, é cultura, pelo menos das empresas que participei, que estou e passei, é cultura nas empresas como parte do desenvolvimento você fazer teste. O mínimo de teste que tem que ter é unitário.

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

E. O próprio código já existente na empresa, artigos, livros, informações da internet, posts e etc e também a passagem de conhecimento na época de outros profissionais plenos e seniores na época que eu era júnior, a passagem de conhecimento deles também.

P. Em quais linguagens você cria testes de unidade?

E. Minha linguagem padrão é Java, mas eu crio testes nas linguagens que eu trabalho, porque eu desenvolvo em Java e eu crio testes em Java, se eu estou desenvolvendo em Python eu crio testes em Python, se eu estou desenvolvendo em JNode eu crio testes em JNode.

P. Qual a linguagem do seu projeto atual?

E. Todas essas que eu te citei.

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

E. Quando eu trabalhava só com Java eu posso dizer que, se eu souber bem o contexto, se eu tiver conhecimento profundo do contexto que eu vou desenvolver, eu gosto de fazer um pouco de TDD, porque eu acredito sim que o TDD ajuda muito na evolução da arquitetura, então, você deve saber, né? Desenvolvimento orientado a testes, então gosto de fazer TDD, mas infelizmente isso não é algo que pode ser realidade 100% no mercado, porque você tem N situações, N circunstancias. Então eu não consigo fazer o TDD, o mínimo que eu faço é desenvolvo a feature específica, ou seja, você tem um app que pode ter uma coisa muito grande, então eu desenvolvo as features, os métodos, etc, e aí eu vou testando por exemplo por cada método que eu sei que tem uma lógica dentro daquele método eu faço os casos de testes para N possibilidades. Por exemplo, se o método tem um IF então eu faço no mínimo um caso de teste para o falso e para o verdadeiro. Eu dependo muito do teste então a priori eu posso dizer que é bem comum fazer o teste depois de desenvolver a feature e o teste, no caso de unidade, eu faço, eu particularmente, baseado nos métodos ou funções e aí eu faço os casos de uso para cada método ou função.

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

E. Não. Porque normalmente o que a gente procura seguir são boas práticas de codificação e isso o teste meio que está vendo essas boas práticas onde diz que uma boa documentação é o próprio código e que os testes podem ajudar ser parte dessa documentação, ou seja, o nome dos testes, não está escrito dentro de um documento da empresa, mas a gente tem referências de livros, de autores, como por exemplo o Uncle Bob, com as boas práticas dele, o clean code, então uma das coisas que se fala tanto nesse livro quanto em outro é boas práticas de documentação, onde não é uma documentação morta, é uma documentação orgânica e viva do código e o teste faz parte. Escrever bons testes no que diz respeito a dar nome certo, dizer exatamente aquele teste faz, o que espera, o que deveria esperar, isso é digamos o padrão que a gente segue.

P. Quais ferramentas são utilizadas para criar e executar testes unitários?

E. Eu acho que dentro de Java, o que é meio global é usarmos o JUnit. Então se usa o JUnit, aí dentro dele você usa algumas ferramentas que ajudam, os mocks, isso dentro do Java especificamente. e tipo, se você quer cobertura de teste, a própria IDE te ajuda a ver cobertura de testes. Ela roda ou você pode dar um comando específico, depende muito de que projeto você está trabalhando e a abordagem. Eu te falo mais de Java, JUnit, porque como falei a princípio ela é a linguagem que eu mais trabalho. O mesmo vale para o python, eu uso uma lib específica lá do python que me ajuda a fazer testes. Do Node é a mesma coisa.

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

E. Uma premissa padrão é que nós mesmos devemos testar o nosso próprio código, está claro dentro das empresas que eu trabalhei até a que estou hoje. Ali além de você fazer o código, testar o código, sempre passa para no mínimo para uma outra pessoa fazer o code review que vai verificar sua escrita e seu teste para ver se seu teste realmente está testando como deveria ser. Porque um teste mal feito pode te dar falsos positivos ou falsos negativos, então um code review bem feito ajuda a você ter uma outra visão, de repente que você não teve enquanto estava desenvolvendo e esse bem escrito parte muito daquilo que te falei, o nome do teste... Uma coisa importante, o teste bem escrito ele não deve testar muitas coisas, ele deve testar uma coisa específica, então tipo, nos asserts, verificar as asserções dele não deve verificar várias coisas. O ideal seria que cada teste fizesse uma única asserção.

P. Como é o processo de manutenção do código de teste de unidade, no caso?

E. Como eu falei no começo, o código de teste ele não é sozinho, ele sempre vem junto com o código em si, então toda vez que você mexe numa parte de um código, tipo, você sempre roda todos os testes existentes e vê se quebrou algum, aí a parte de manutenção, porque se quebrou algum você precisa fazer adaptação, significa que sua regra mudou. Ou você corrige o teste para atender a nova regra ou você entende que sua regra não está sendo feita de forma adequada e por isso seu teste quebrou. Então esse processo de manutenção do teste ele deve caminhar junto com a manutenção do código. Se você deixa eles de forma separada, você vai ter a documentação morta. Tipo, seus testes nunca vão evoluir porque nunca vai ser prioridade.

P. O que você conhece sobre test smell?

E. Test smell? Essa expressão, tipo eu posso dizer que eu já ouvi falar, mas a vivência... você vê tantos termos que as vezes eu posso não saber pelo nome, mas se você me disser "é isso" eu posso dizer, mas só pelo nome não lembro o que é.

P. Como você conheceu este assunto?

E. Quando a gente fala de smell, a gente não fala só de teste, a gente fala de código em si, coisa que não está parecendo adequada. Não é literal, mas você tem um cheiro estranho para aquilo que parece não está fazendo o que deveria fazer.

P. Eu perguntei pois é uma pergunta da pesquisa e porque as perguntas a partir de agora vão estar relacionadas a isso. Eu vou te passar o conceito, porque caso você tenha uma analogia a alguma outra expressão você conseguir fazer essa linkagem, caso você tenha alguma pergunta você me faz, a gente alinha o conceito e depois a gente segue. Tudo bem?

E. Eu posso não estar certa, é como eu te falei, mas test smell para mim parece com o negócio do code smell, pode não estar certo.

P. Isso. Test smell ele descende dos code smells, na verdade. Te passando o conceito certinho, eles 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. Existe uma quantidade de smells específicas que elas diferem dos code smells, mas tem a ver com coisas por exemplo que você já comentou, como o teste ser para um objetivo específico, e nesse caso, um método tenha uma complexidade maior, seria criado testes diferentes, ou então a questão de um assert. Você ter um teste específico para cada um. Isso tem a ver sim com smells, até tem smells específicos justamente para cada um desses tipos que você citou. Eu só te passei o conceito para que a gente consiga alinhar a informação e seguir com o mesmo pensamento.

E. Beleza. Até antes, de você citar, eu me prendi a testes, mas quando esses testes têm essas características onde ele testa muitas coisas, não diz respeito ao teste, mas diz respeito ao código, porque o código ele está tendo mais de uma responsabilidade fazendo mais coisa que deveria.

P. Eu vou passar para a próxima pergunta, ok?

E. Uhum

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

E. Essa é uma pergunta estranha porque como eu falei, a gente procura seguir boas práticas, e aí vamos dizer assim, os tests smells, quando eu percebo alguma coisa estranha, então depende muito de que coisa estranha é essa. Tipo, exatamente... o nome do teste fala que ele faz uma coisa e quando você executa ou debugga ele não faz aquilo. Então isso pode ser um test smell. A forma adotada não é a correta. Ele faz uma asserção de muitas coisas então você não sabe exatamente o que está testando, então se esse teste vier a quebrar você não consegue identificar de cara qual é o problema real. Então tratar isso seria dividir esse teste para cada um testar especificamente. Da mesma forma que o código por todo, cada método ou função deve ter uma responsabilidade única, o teste também deveria ter uma responsabilidade única de testar um único case.

P. Em relação a manutenção, por exemplo, caso na criação de um código de teste não houve o devido cuidado e foi enviado um teste com smell, no momento que está sendo feita a manutenção do código e consequentemente, da forma que vocês fazem, a manutenção do código de teste, se ele é identificado ele é tratado? Como vocês lidam com isso?

E. É uma boa prática. Como o pessoal fala é a prática do bom escoteiro, se você passa por um lugar você deve procurar deixar mais organizado e mais limpo do que você encontrou. Então assim, uma boa prática é se você identificou que um teste não está fazendo o que ele deveria fazer e isso pode induzir ao erro, ele deve ser corrigido.

P. A gente comentou um pouco a respeito, mas você pode extrapolar o pensamento, não focando especificamente no seu dia a dia. Para você, como seria 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. Como prevenir. Você saber o contexto, o que deve ser testado, o que você espera de resultado, o que você quer testar, porque você está fazendo esse teste. Você fazer essas perguntas podem te ajudar a prevenir que seja um teste falso. Por exemplo. A gente vê por aí muitos testes unitários que não fazem sentido porque algumas vezes, eles mockam toda a resposta. Então você não consegue testar. Ele pode ter um test smell porque você não vai testar adequadamente. Então você fazer uma lógica adequada para o teste, para ele testar coisas específicas, como falei anteriormente pensando "porque eu estou fazendo, qual é o meu objetivo, para que" eu acho que são boas perguntas para evitar. E tipo, generalizando, estudar mais sobre códigos, boas práticas de código que estão ligados a teste ou não, entender alguns princípios de engenharia de software mesmo, como a questão do S.O.L.I.D., tudo isso não se aplica só ao código da máquina, mas ao código de software também, procurar entender, pode ajudar a formar um profissional para fazer isso, porque isso infelizmente não é uma receita de bolo que aprende, pega e faz de um dia para o outro. Você vai pegando expertise nisso.

P. Você tem alguma pergunta para mim, quer acrescentar alguma informação ou tem alguma sugestão para melhorar a entrevista? Essa é a última pergunta.

E. Eu acho que não. Você trabalha com teste especificamente?

P. Eu trabalhei como QA durante pouco mais que 7 anos então minha vivência é muito mais a parte de sistemas que na parte de unidade, mas a partir do momento que eu voltei para a academia eu passei a ter mais contato com isso, e identifiquei dentro da área de testes essa questão de test smell, e voltei para a indústria para identificar em que contexto ele está sendo trabalhado e de que forma a gente pode melhorar esse processo.

E. Eu acho que uma coisa que eu tenho para acrescentar é que cobertura não diz tudo. E depende muito da linguagem, mas se vamos usar Java, que é o que muitas empresas usam, uma cobertura de 100% não quer dizer que o código está sendo bem testado. Que os testes estão sendo bem feitos, então acho que isso eu tenho que acrescentar. O teste de fato testar uma lógica e procurar, como eu já falei várias vezes, em cada case, caso de uso, você ter um teste específico. Testar só para ter cobertura, só para citar um exemplo, em Java ter gets and sets só para ter uma cobertura não faz sentido, então você deve verificar o que deve ser testado. Eu sei que está no final, mas é que eu preciso ir...

P. Se você quiser a gente pode encerrar. Todas as perguntas foram feitas. Eu só ia te agradecer mesmo por ter participado e qualquer dúvida que você tiver, estou a disposição.

E. Obrigada por me contatar e desculpa por ter furado ontem.

P. Acontece. A gente está se adaptando no que a gente pode, então é tranquilo, é bem compreensível.

E. Está bom então. Obrigada.

P. Boa noite para você. Tchau.

E. Tchau.