Pesquisador. Mais uma vez muito obrigado por participar dessa entrevista. Ela tem como objetivo compreender os processos de criação e manutenção de código de teste de unidade e como a qualidade do código de teste é verificada nesse processo. Não há respostas certas ou erradas ou desejáveis ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que pensa.

Entrevistado. Ok

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. Foi em 2015. Basicamente eu era da área de suporte, redes. Eu venho da área de redes basicamente e aí eu decidi migrar para área de sistemas. Na época eu fui atrás de cursos, eu estava ainda cursando a faculdade. Eu sou formado em segurança da informação então é mais voltado para a área de redes mesmo. Eu até tentei entrar em vagas de estágio para programação, mas como meus conhecimentos de programação era do técnico que eu tinha feito e da faculdade não tinha o foco em programação acabou não dando certo, e aí eu me interessei pela área de qualidade, principalmente por ser uma área relativamente nova. Apesar de já ter essa função faz um bom tempo, nas faculdades não eram muito falada da área de testes de software e ai eu comecei a me interessar. Outro ponto que me ajudou bastante por eu trabalhar com suporte, trabalhar com troubleshooting, coisas do tipo, um pouco de Linux, esse tipo de coisas, eu mesmo sem experiência consegui entrar no trabalho que estou.

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

E. Basicamente dia-a-dia de trabalho, entrei em 2015 então eu tinha basicamente cursos teóricos online sobre testes no TI Exames, principalmente manuais, voltado para certificação de software CTFL, e tinha conhecimento de programação que vinha da academia. Então no dia a dia a empresa apresentou os frameworks que o cliente usava e aí eu fui estudando, olhando o que tinha de código e aprendendo. Foi basicamente e-learning que eu tive contato com Java, e também lá na empresa eles deram cursos de Java na época e aí com Java eu parti para outros frameworks como Selenium, a gente usava o robot framework que era o principal que a gente usava lá. Então foi bem na prática o aprendizado que eu tive sobre codificação de testes.

P. Em quais linguagens você cria teste de unidade?

E. Atualmente não trabalho mais com teste de unidade, mas na época a gente criava testes em Java basicamente. Um pouquinho de Javascript com Jasmine, mas bem pouco. Eu tive mais contato com Java. Mas a experiência mesmo é mais em teste UI que teste de unidade.

P. Qual plataforma do seu projeto atual?

E. Java, todos os testes eram feitos em Java para web.

P. Como era seu processo de criação do código de teste de unidade.

E. Tinha uma regrinha básica que o ideal era que quem escrevesse o método ou alterasse a classe em questão ou a pessoa ia fazer alguma manutenção, criava o teste de unidade. Você está mexendo, dando manutenção em alguma classe, está vendo que está faltando algum teste de unidade, já criava. Eles tinham um aplicativo de code coverage, que é o Sonar, que mostrava a cobertura do código. Tinha uma porcentagem, não lembro de cabeça qual era, mas tínhamos que atingir aquela porcentagem mínima para o Jenkins, o software de integração contínua dizer que estava ok o código para ser “mergeado”. Se fosse criada uma classe nova, se eu tentasse mandar um código para revisão, já receberia um negativo do Jenkins que não atingiu a cobertura mínima, “você precisa criar testes unitários”. No caso dessa empresa, que foi o meu primeiro cliente, o teste era mandatório. Você não entregava o projeto se não tivesse testes de unidade feito, teste de UI também. Então fazia parte do escopo. Isso não é a realidade de todos os projetos. Na verdade, é meio que ao contrário, mas são poucos os projetos, agora as coisas estão mudando recentemente mas a maioria dos projetos não tem essa exigência de ter o teste, principalmente teste de unidade, que desenvolvedor não gosta muito de testar. Junta isso com a não exigência do cliente, pressa na entrega, tudo isso acaba “despriorizando” os testes.

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

E. Não tinha. Só tinha algumas boas práticas, quanto a variável, nome de métodos, eles tinham um documento de algumas boas práticas. Tanto de coisas simples como nome de variáveis, como fazer um assert, até padrões de commit, como tinha quer feito para ter a devida aprovação. Como era feito principalmente por pessoal de fora, se não seguisse àquele padrão, eles acabavam rejeitando o código e teria que fazer de novo.

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

E. Para Java, utilizávamos o JUnit.

P. Dentro da empresa, como é o processo de verificação de qualidade do código de teste criado? Você comentou que o Sonar fazia verificação de cobertura, mas além disso, existia alguma verificação de qualidade desse código?

E. Sim. Através de revisão. Tanto código de feature, quanto código de teste era revisado por pessoas, primeiro do time, segundo iria para um segundo nível de pessoas que eram aprovadoras. Geralmente eram pessoas muito mais experientes, com bastante tempo de empresa que avaliam o código, era uma revisão estática mesmo. Ler o código, olhar tudo. Além disso, a gente tinha a revisão dinâmica, que era via software de integração contínua. Ele rodava seu código, pegava seu código, puxava seu código da master, rodava os testes que tinham lá para garantir que seu código também não quebrou nada. Então tinha esses dois tipos de validações. Validação humana, estática, que olhava a sintaxe, coisas do tipo e a dinâmica que era feita de forma automatizada.

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

E. Basicamente qualquer desenvolvedor, se fosse implementar algo ele tinha obrigatoriamente que fazer o build da aplicação. Quando ele fazia o build, ele rodava os testes, então, se quebrou algum teste, ele já tinha que fazer o fix, corrigir, porque se ele mandasse o código sem o teste corrigido, ele iria para o Jenkins, o software de integração contínua. Ele iria fazer a mesma coisa que o dev (developer) não fez, que é pegar o código e rodar os testes e iria identificar os testes que falhou e não autorizar o código para subir. Basicamente a manutenção era muito tranquilo, porque não era algo que ficava abandonado e de vez em quando era mexido então, a cada vez que subia o código se tivesse algum teste falhando, então obviamente qualquer um que fosse mexer no código e quebrasse algum teste teria a responsabilidade de consertar. Se mudasse algum requisito, qualquer um que fosse mexer naquele requisito, vamos supor, ele recebeu algum tipo de parâmetro e agora vai receber outro tipo, ele vai ter que alterar no test case também; ou ele criou um novo método então ele tem que criar um teste case para aquele método novo.

P. O que você conhece sobre test smell?

E. Não conheço. Já ouvi falar o termo mas não sei o significado, para falar a verdade.

P. Eu vou te falar o conceito de test smell, pois você pode ter lidado no dia a dia, mas não pelo termo. Eu preciso te apresentar esse conceito porque as perguntas a seguir estão relacionadas a esse tema. “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.” Eu vou te mandar dois exemplos, conversamos sobre e depois seguimos, ok?

E. Ok.

P. Esse que eu te mandei é chamado Conditional Test Logic, ele é considerado um smell porque possui estrutura de repetição ou uma estrutura condicional. Nesse caso ele é considerado um smell, pois caso esse teste falhe, vai ser mais difícil identificar em qual momento da execução do sistema, esse teste falhou, devido a essa estrutura. Então o ideal seria que essas condições ou repetições fossem incluídos em testes separados para que uma vez o teste falhe fosse muito mais fácil de identificar. O outro exemplo que estou te mandando, ele é o chamado Empty Test. Ele é considerado um smell porque ele não tem nenhum script executável dentro desse teste. Caso você tenha uma quantidade alta desses testes vai ter dificuldade de identificar de porque eles não estão sendo executados e fazer a correção necessária. Então a existência de testes sem nenhum script executável é algo que vai acabar dificultando a atividade de teste em si.

E. Apesar de não ter trabalhado exatamente com esse tipo de conceito, o próprio Sonar ele avisava a gente sobre esses dois problemas, tanto quando a lógica era muito complexa, com muito "if", ele mandava a gente quebrar isso em métodos diferentes, coisas do tipo. E eu lembro também, que ele identificava comentários, código comentado, e manda um warning. Então quando você está usando a extensão dele no Eclipse, por exemplo, você consegue identificar se tem comentários também. São ferramentas que ajudam, ele tem regras nele que você pode customizar, e uma delas eram essas duas, relacionado com o que você mostrou. Interessante, eu não conhecia o termo.

P. O Sonar fazia a verificação tanto no código de produção quanto no código de teste?

E. Sim. Ele varria todo o código, tudo que você publicava. Você tem a extensão dele para IDE, então ele analisa o código e baseado nas regras que ele tem, ele vai te dando os feedbacks. Ou se você publicasse o código de produção, o seu teste para ser revisado, ele também tinha um Job do Jenkins que analisava, fazia o mesmo processo, checava se o código tem comentários ou se a lógica era complexa e recomendava quebrar em mais métodos.

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

E. O Sonar fazia essa verificação. Tinha também o próprio revisor que podia comentar se ele encontrasse alguma coisa do tipo, mas normalmente o próprio Sonar acusava para nem precisar chegar no revisor humano para ver esse tipo de coisa.

P. No processo de manutenção, os test smells eram tratados com esse mesmo processo?

E. Sim. Normalmente tinha código legado, código que foi desenvolvido antes de ser implementado o Sonar, quando você mexesse nele, por algum motivo, caso tenha alguma lógica complexa em qualquer parte do código, o Sonar vai acusar porque ele vai analisar a classe inteira que você mexeu. Então as vezes acontecia de você ter mexido numa linha abaixo e ter algum código acima ou mais abaixo que não tenha mexido, não foi o autor, mas precisar arrumar por conta do Sonar acusar.

P. Para você, como seria possível prevenir/evitar a inserção de test smell no código de teste no momento da criação desse código?

E. Eu recomendaria, o único que eu conheço, o próprio Sonar. A extensão do Sonar para o Eclipse, por exemplo. Você instala e ele vai te dar os warnings. Isso ajuda bastante você não ter que mandar seu código para revisão e uma pessoa pegar. Ele já acusa na hora. Por exemplo, uma lógica complexa, como essa que você mostrou, uma estrutura de repetição e de condição no mesmo teste, ele acusaria. Obviamente se o sonar for configurado para isso. Ele tem várias regras default, mas precisa configurar várias coisas também.

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

E. Eu fiquei curioso em relação ao assunto, eu achei bem legal. Eu queria saber se na sua universidade tem matérias relacionadas a teste de software, teste unitário, coisas do tipo.

P. Eu estou fazendo mestrado, então quando eu fiz a graduação, eu sou formado desde 2009. Quando eu fiz a graduação, eu tive a parte de qualidade de sistema muito mais voltada para auditoria do que para teste em si, teste de unidade ou outras possibilidades de teste. Recentemente, já vem sendo abordado, pelo menos na universidade a qual faço mestrado, já vem sendo abordado a parte de testes dentro do curso de graduação, por exemplo, mas não sei te informar especificamente se tem uma disciplina que aborde exclusivamente isso, lá existe disciplinas de Engenharia de Software e dentro dessa disciplina são comentados a respeito de testes, depende muito da ementa do professor e da disciplina em si para ele poder abordar, mas já tem sim, vem sendo abordado no curso de graduação, tanto que, até no final do ano passado ou início desse ano, já teve trabalhos de conclusão de curso já voltado para ferramentas de teste, por exemplo. Então, já vem sendo abordado na graduação mas não saberia te dizer qual a profundidade disso.

E. Bacana, interessante.

P. Meu mestrado, eu estou abordando a parte de qualidade de teste de unidade focado em test smell. Como eu vim da indústria, eu trabalhava como Analista de qualidade e esse assunto nunca chegou para mim, por exemplo. Eu precisei voltar para a academia para ter acesso a esse tema e a minha curiosidade era saber até que ponto a indústria conhece esse tema, seja com o nome específico ou trabalhando sem saber se esse termo existe, e se existe alguma forma de melhorar os processos industriais para reduzir a inserção de test smells nos testes. E minha pesquisa tem sido através disso, conversando bastante com os profissionais e tentando conhecer mais a fundo o cenário da indústria e vendo onde é possível abordar.

E. Interessante, bem bacana. A realidade, é que algumas empresas têm investido bastante em automação de testes desde teste unitário até de UI, passando por integração e assim por diante, mas ainda é um paradigma a ser quebrado, por que não são todos os clientes que aprovam esse tipo de coisa, para eles é difícil mostrar para o cliente o ganho que ele vai ter com automação de teste porque não é nada que ele possa visualizar, como ele vai investir X reais em automação se ele pode investir em uma tela nova para o sistema? Então é difícil colocar isso na cabeça dele. Eu tive a sorte de entrar num cliente que já tinha toda essa cultura enraizada. Até porque o projeto do cliente é bem grande, então se ele não tivesse esses testes para regressão seria muito difícil de manter o produto dele, porque é um produto muito grande da área de telefonia então seria muito complicado. Mas é interessante ter pesquisa nessa área. Eu acho bem bacana, então foi por isso que eu perguntei, porque eu pretendo fazer uma pós, alguma coisa desse tipo, especificamente da área de qualidade de software.