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

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. Perfeito

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

E. Eu fiz tanto curso técnico quanto faculdade, então desde o mundo acadêmico eu fui instruído para ter testes. Eu me formei na faculdade em 2007, imagino ter começado 5 anos disso, durante todo o intervalo a gente já tinha as práticas de TDD, de escrever o teste, pensar o software antes de desenvolver, mesmo na atividade profissional, durante, logo após a formação, já usava teste.

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

E. Eu acho que a graduação foi o primeiro passo, como ensinavam tanto Java quanto .NEt então tinha tanto o JUnit quanto o NUnit, depois a partir daí foram descobertas em fóruns, eventos. Junit ainda é muito usando no ecossistema Java de várias experiências que tive depois, mas depois eu fui aprendendo outras variações e ainda tem a questão de outros ecossistemas, tanto em node, Javascript tem o ecossistema de testes com o Mocha, com Chai, com outros frameworks, e mesmo a linguagem que eu gosto mais, que eu dediquei mais tempo que é o Groovy, ela também expande usando o Spock, também são coisas que aprendi em eventos ou seguindo páginas de produtos, linguagens.

P. Em quais linguagens você cria testes?

E. Eu diria em todas. Na minha carreira eu trabalhei mais em cima do JVM e variações, tanto Java, Scala, Groove, Kotlin atualmente, Node e variações como Typescript principalmente, mas mesmo quando faço estudo em outras linguagens, por exemplo, no meu Github tem rust e golang, ainda assim eu também faço testes.

P. Qual linguagem do seu projeto atual?

E. Kotlin é a linguagem principal

P. Me conte como é seu processo de criação de teste?

E. Usa a prática do TDD, então usando as palavras do Kent Back é muito de perguntar para o código em formato de código, como o código vai funcionar, por exemplo. Imagina que eu vou fazer um exemplo bem simples que a gente faz em dojo de pedra papel tesoura, então eu começo sim escrevendo o teste, tipo "olha eu vou pegar o primeiro caso, eu vou dizer que a pedra ganha da tesoura", então eu vou codificar esse cenário, eu vou dizer, quando o jogador 1 colocar pedra, quando o jogador 2 coloca tesoura, eu vou chamar a função qualquer, jogo, definir um nome, vou dizer que o resultado é pedra vence, e a partir daí sim eu vou desenvolver o código para resolver esse problema.

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

E. Não. Explicitamente não. Um é padrão da indústria. Maior parte das pessoas, empresas, nos cargos anteriores pregam a prática do TDD, ela é bem vista e outra é a dosagem de se é mais unitário ou é mais funcional é "pessoal" de cada time, de cada problema.

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

E. Na stack Java a gente usa ide, IntelliJ e o framework mais usado é o JUnit, sendo que eu expandiria para usar o Spock que é do Groovy. Vão ter outras libs auxiliares, como no Kotlin tem o MockCart, vai facilitar para mocks e stubs, como tem fluent, que ele facilita para deixar os testes mais verbosos.

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

E. Particularidade de cada time, tem time que usa a linguagem Cucumber, Gherkin digamos assim. A gente tem a garantia que o requisito foi entregue, mas tem times que colocam direto no Junit com talvez alguns comentários guiando o Gherkin, dizendo dado esse cenário de teste quando eu rodar isso aqui o que é esperado, “Give, When, Then, mas de novo, isso particular de cada time e de repente estendendo na sua pergunta, temos analisadores de código como o SonarQube que vão dar complexidade do código, isso é uma forma de medir e daí dependendo do time, se tem regras de negócios mais pesada, a gente usa testes de mutação também.

P. Esse teste de mutação é feito manualmente ou utiliza alguma ferramenta?

E. Usamos ferramenta. PhyTest para Java tem variações dele para node, Stryker, que a gente usa para node.

P. Quando é feito o teste de mutação, os testes gerados por essa ferramenta eles também são mantidos ou depois que os testes são validados, os testes gerados pela ferramenta são descartados?

E. Como essas ferramentas, elas instrumentam o teste, elas vão mutar o teste que está escrito a gente não guarda os artefatos que são gerados, a gente guarda, o que a gente olha são os resultados dessa execução

P. Passando para o processo de manutenção do código de teste, como ele ocorre dentro da empresa ou equipe que você trabalha?

E. Ele ocorre junto com as alterações de requisitos ou bugs, toda e qualquer mudança no código é esperado que o teste seja alterado. Obviamente, um adendo, se temos teste que toda hora a gente tem que mudar, temos testes problemáticos, mas se o requisito muda, se uma nova funcionalidade é incluída, ou se um bug é encontrado, isso também é alterado.

P. O que você conhece sobre test smell?

E. Por esse nome, pouca coisa, mas se você disser a definição, cheiro de teste, smell mesmo, eu entendo que assim como a gente tem os code smells, quando a gente olha algumas práticas de código lá, exemplo, eu tenho um código que tem 200 linhas. Não necessariamente esse código é ruim, mas é um smell é um guia, um cheiro que tem oportunidade de melhoria. Obviamente esse código não é simples, tem 200 linhas e tal, e pode ser refatorado seguindo as práticas de clean code, reduzir. Imagino assim que em testes temos smells, por exemplo, se tem um teste que não tem nenhuma validação final, não tem nenhum assertion, tem um cheiro de que aquele teste não está bacana, que o resultado dele não está mostrando a realidade, que seja ou funcionando ou que seja dando erro.

P. O princípio é esse mesmo, os tests smells eles descendem dos code smells. Eu vou te passar o conceito para formalizar, mas é o que você comentou mesmo. 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. Então basicamente o conceito tem a ver com o que você já tinha comentado. Tem smells relacionados a asserts, construtores, a própria estrutura dele, por exemplo, caso ele não tenha nenhum script executável, ele é considerado um smell. Eu vou compartilhar a tela com você e mostrar 2 exemplos que na literatura são considerados smells. Por exemplo, se você tem um teste que não tem nenhum script executável, ele é chamado de Empty test smell. Ele vai passar ou falhar sem que nada tenha sido feito dentro dele. Isso vai prejudicando a atividade de teste em si. O outro exemplo é o chamado Conditional Test Logic smell. Ele tem estruturas de condição e de repetição que, além ao depender do profissional possa ter uma dificuldade maior de compreender esse teste, uma vez que esse teste falhe é mais difícil identificar do porque, em que ponto específico dessa condicional, dessa repetição esse teste falhou.

E. Perfeito. Esse código específico entra em caso específico de clean code. Esse teste pode estar fazendo mais coisas que deveria.

P. Você tem alguma dúvida em relação a esse assunto? Porque as perguntas a seguir têm a ver com esse tópico.

E. Por enquanto não. Por enquanto está de boa.

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

E. Cai naquele quesito de testes como Sonar pegaria os dois exemplos que você mostrou. Um de código vazio, e outro de complexidade. Durante o pull request, quando a gente fosse revisar o código, a gente teria no relatório que esse código introduz alguns bugs, novos smells.

P. Pelo que você comentou o processo de manutenção do teste tem a ver com a manutenção do código de produção. Nesse momento existe algum tipo de cuidado para identificar e tratar smells que eventualmente existam?

E. Sim, a gente usa o dilema do escoteiro que cada vez que alguém faz uma manutenção que tem que entregar o código num melhor estado do que pegou, mas de novo, a gente não tem um checklist para isso. O sonar ele entrega métricas de complexidade, de defeitos, entrega um checklist e o ideal é que a cada modificação, cada atualização que a gente entregue menos defeitos e não mais.

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

E. Usando ferramentas de continuos integration, imagino que a cada commit, a cada fim de alteração que o “dev” faça que ele empurre essa change para o repositório master, seja ele qual for, que a partir daí o CI rode os testes, rode as validações e verificações e retorne isso, talvez essa forma seja um pouco mais custosa. Uma outra seja plugin ou algo que rodasse direto da máquina do desenvolvedor para ter um feedback mais rápido, seja em radle, mavin, no gerenciador de build digamos assim.

P. Estou chegando no final da entrevista. Você tem alguma pergunta para mim, quer acrescentar alguma informação ou ainda tem alguma sugestão para melhorar a entrevista?

E. Eu achei muito legal, gostei acho que conseguimos caber dentro do tempo. Deixo meus contatos, se precisar mais perguntas, discutir mais sobre o assunto. E além disso, eu até palestrei sobre o tema testes. Eu vou te mandar o link da palestra se tiver interesse. Até estava esperando que nessa entrevista a gente entrasse em temas mais polêmicos tipo "o que é melhor, se mais unitário, mais funcional, integrado, end to end" a gente não caiu nada nesse cenário. Eu particularmente tenho uma visão diferente da maioria dos “devs” digamos assim, eu prefiro muito mais os testes funcionais os testes de requisito, tipo Cucumber do que de repente um teste unitário, geralmente o pessoal confunde a unidade, tipo para mim uma unidade, eu penso uma unidade de valor e o pessoal pensa na unidade como um método e daí entra aquela discussão de testar um método privado ou coisas desse sentido, mas mais isso, é bem um comentário específico sobre essa parte de teste. Você foi super querido, agradeço pelo seu tempo. Me deixo a disposição para mais testes nesse sentido

P. Muito obrigado. Depois eu vou acessar sua palestra. Minha pesquisa ela é focada no test smell, então por isso eu acabo não entrando nesses outros pontos de teste de integração, end to end e assim por diante por conta do foco da minha pesquisa, mas é bacana saber que existem esses tipos de discussão, pois pode ser, por exemplo que outro pesquisador do meu grupo de pesquisa entre nesse tópico, ou eu enxergue isso como uma necessidade, uma vez que nem todo mundo trabalhe com teste de unidade, trabalha com outros tipos de teste eu passe a pesquisar esse tipo de coisa no doutorado, por exemplo. É bacana ter esse tipo de feedback para que outras pesquisas possam ser realizadas. Muito obrigado.

E. Eu que agradeço por ter participado, te desejo boa sorte no teu trabalho, que dê tudo certo.

P. Obrigado, boa tarde para você.