Pesquisador. Mais uma vez muito obrigado por ter aceito participar dessa entrevista. Ela tem como objetivo compreender os processos de criação e de manutenção de código de testes e também como é realizada 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 realmente pensa.

Entrevistado. Certo

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. O relatório final conterá os comentários dos participantes sem nenhuma referência aos indivíduos.

E. Ok.

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

E. Na verdade eu sou desenvolvedor no geral, aí a uns 6 anos atrás comecei a desenvolver testes, comecei a parte de testes mesmo. Vi que isso era importante comecei a estudar sobre a aplicar testes unitários, integrados, no meu dia a dia.

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

E. Troca de conhecimento em equipe, os colaboradores e colegas e vi vídeos na internet: YouTube, curso na Udemi, lendo artigos na internet. Livros sobre clean code... é isso. Prática.

P. Em quais linguagens você cria testes?

E. Hoje em C# .NET principalmente

P. Qual plataforma do seu projeto atual?

E. C# .NET

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

E. eu tenho noção do que eu tenho que desenvolver, eu tento começar com TDD, escrever a regra do negócio dirigido ao teste. Escrever primeiro o teste depois o código. Quando tem necessidade. Nem sempre eu faço com o TDD. Se eu vejo que é um código pequeno, nada complexo, que eu vou conseguir escrever o meu código e depois um teste para ele praticamente 100%, eu as vezes inverto. Não uso 100% TDD. Conforme eu vejo a necessidade.

P. Existe algum fluxograma, template na empresa que padronize esse processo de criação de código de teste?

E. Por enquanto não vi. A gente só adota boas práticas de ter 80% de cobertura em código novo e cada coisa nova tem que ter teste e aí padrões assim de código limpo, teste limpo, direto. Não tem nenhuma convenção de como escrever.

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

E. Eu uso o próprio Visual Studio que tem a ferramenta de execução de testes ali do próprio Visual Studio, do compilador e para desenvolver eu uso o MSTest. Tem pessoas que usam o NUnit, mas eu uso o nativo mesmo.

P. Dentro da empresa, como é o processo de verificação de qualidade do código de teste criado?

E. Na empresa que estou atuando eu comecei a pouco tempo. Tem 2 meses. Na minha equipe, na parte de C# eu estou, eles estavam contratando novos em C# que eles não tinham ninguém. Contratou eu e um outro cara e eu que tenho mais experiência ali. E ainda não estou trabalhando com outros times, então eu não posso falar sobre como os outros estão medindo a qualidade do código de teste, pois eu sou o principal que revisa, o que bate bastante em cima de testes. Falando no que eu reviso, eu reviso os testes numa forma que ele seja bem claro no que aquele teste quer testar seja um nome bem indicativo de qual é a regra de negócio está sendo testada. "Soma de um número e outro deve retornar X" bem que a pessoa leia e saiba o que a regra de negócio faz. Que seja bem claro, todas as partes de preparar os dados de teste, executar os testes e validar os retornos, seja bem direto para que o teste seja a própria documentação do que aquele método deve fazer. Às vezes eu tento fazer o teste de forma que eu lendo o teste não precise ver o detalhe da implementação, eu já sei que aquele código se comporta daquela maneira. E evitar muita regra de negócio nos testes, evitar muita coisa desnecessária. Reutilizar, não ficar as mesmas coisas em vários testes, tentar reutilizar, criar um setup, alguma coisa assim nas classes de teste e tentar deixar o mais limpo possível. É o que eu prezo e também estava tendo uma boa cobertura. Um revisa o código do outro. Eu reviso o do meu colega, e ele revisa o meu. Nós temos o mesmo cuidado.

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

E. Como a gente está num projeto relativamente novo, as manutenções são poucas. Conforme está vindo algumas refatorações do código do software, os testes que a gente faz primeiro é para não quebrar, para validar o negócio. Se o negócio mudar, a gente tem que fazer de forma que o teste não quebre, que não quebre a ideia inicial ou refazer o teste de uma forma que ele não vire um código macarrão. Você vai adicionando várias coisas no teste e ele que reflete que o código está mal escrito. O código do negócio. Se o requisito mudar e tiver que refatorar, a gente tenta refatorar os dois em conjunto. Tanto o negócio quanto o teste para manter limpo os dois e ter uma boa cobertura, uma boa legibilidade do código e tudo mais. Recentemente teve um caso que mudou o processo de criar logs. A gente teve que usar uma outra ferramenta para criar logs e os testes tinham injeção de dependência. Os testes usavam esses logs mockados para passar para as classes que iam ser testadas. Aí teve que mudar a forma de logar e teve que mudar os mocks. Foi uma coisa bem pontual que não afetou. Que mudou de um tipo para o outro. A gente manteve a legibilidade do código e ele continuou passando sem problema nenhum. As regras de negócio não foram alteradas. Eu tento sempre dessa forma.

P. O que você conhece sobre test smell?

E. Eu não li, mas pelo que eu entendi é uma coisa parecida com code smell?

P. Sim.

E. que é mas práticas na escrita do código, a ter bugs ou problemas que podem dar problemas de variável que não está sendo utilizada, alguma coisa que pode quebrar que pode atrasar alguma regra de negócio em cima dos testes que pode fazer com que ele tenha um comportamento que não é esperado, que não execute rápido, que tenha dependência com outro teste, com alguma coisa que não deve ter, alguma coisa nesse sentido?

P. Isso, é exatamente esse princípio. “test smells são pontos fracos na estrutura do código de teste, tendo um impacto negativo na sustentabilidade da atividade, pois os tornam complexos, difíceis de entender e modificar, prejudicando a repetibilidade, independência e estabilidade dos testes.”

E. Certo

P. Você comentou que fez uma analogia com code smells. Você já tinha ouvido falar de test smell?

E. Não. Com essa nomenclatura não, mas no dia-a-dia você tenta ter código limpo, ter teste limpo a gente acaba seguindo essas práticas mesmo sem conhecer os nomes.

P. Você possui alguma dúvida sobre esse tema?

E. não sei se tem mais definições além do que eu conheço sobre boas práticas, sobre código limpo de teste limpo e tudo mais. E até porque eu li a versão em português do clean code então provavelmente tinha esse termo lá mas como está traduzido ele aplicou para testes limpos.

P. O conceito é aquilo que você já tinha comentado fazendo uma analogia a code smell. Eu li para você o conceito de test smell. Eu posso te passar dois exemplos de test smells e explicar porque ele é considerado um smell e a gente segue daí porque as perguntas a seguir têm a ver com esse tema. Ok?

E. OK.

P. Vou te mandar dois exemplos e a gente conversa sobre. A primeira é o chamado Empty Test. Esses dois testes estão em Java mas o conceito é o mesmo. O empty test ele é considerado porque não tem nenhum script executável. então uma vez que tenham testes com esse tipo de estrutura fica mais difícil de se rastrear porque ele está comentado, ao invés de ter sido excluído e a existência de vários testes dessa forma acaba impactando no resultado da atividade. O outro ele é o chamado Conditional test logic. ele é considerado um smell porque ele tem estrutura de repetição, condicional dentro dele. Uma vez...

E. Muita lógica né?

P. Exato. Então uma vez que esse teste falhe é mais difícil de identificar o porquê que ele falhou, em qual momento da condicional ou da repetição ele falou. Esses são alguns exemplos de test smells. Têm smells relacionados a asserts, e por aí vai, Mas tem muito a ver com o conceito que você mesmo comentou. Torna os testes mais complexos, difíceis de entender e de modificar e assim por diante.

E. Certo.

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

E. Como eu já disse, a gente faz o code review, bem criterioso, em cima do código do software quanto de testes. A gente tenta não botar, como você mostrou ali muito IF, FOR, estrutura de repetição nos testes, código comentado é abolido. Zero código comentado, não deixa porque tem o versionamento, tem o git ali para precisar deletar código que não é usado, a gente faz tranquilo porque se precisar voltar depois a gente tem histórico, não tem problema. A parte de boas práticas de variáveis que podem ser reutilizadas ou de coisas que podem ser reutilizadas, a gente conversa bastante. As vezes tem num teste uma preparação de um dado para teste que está se repetindo em vários lugares e a gente tenta criar uma “comom”, alguma coisa assim para fazer essa preparação, mocks, alguma coisa o tipo. Sempre prezando a simplicidade, nada complexo. Comentários desnecessários, a gente sempre tenta mostrar as intenções através do código sem ter que ficar comentando o que ele faz por causa de uma deficiência nas nomenclaturas ou ficar colocando código redundante que a função diz "faça isso" e tem um comentário dizendo "faça isso" então não tem necessidade. Isso tudo a gente olha no code review quando está desenvolvendo para ter algum código limpo, legível, sem smells.

P. Como a verificação de smells é feita na manutenção do código de teste?

E. Na manutenção, como eu comentei que é pouca, a gente tenta. Tem um teste que não está mais sendo usado a pessoa já pensa que "isso pode ser usado depois, ou vou colocar aqui para aproveitar depois" isso não passa. Isso não teve nenhum caso agora novo, mas caso aparecesse a intenção seria não comentar nada e também se tiver que mexer muito nos testes, fazer muita manutenção, tentar abstrair, tentar ver se não tem algum teste não está testando, alguma função que faz muita coisa e tem que quebrar em mais funções, e consequentemente quebrar em mais testes. Refatorar tudo. Tudo isso a gente dá uma olhada também e na cobertura. Se for dar manutenção nos testes se a cobertura afetou, se melhorou. Porque ela não pode piorar. A cobertura de código, ela se tem que refatorar, o mínimo que tem que acontecer é ela manter a mesma, mas sempre tem que manter a mesma ou melhorar, nunca piorar a cobertura de código.

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

E. Uma coisa que a gente usa são os Lint, no C# tem o SonarLint que eu uso, que ele evita no código como um todo. Bugs, comentários, ele avisa variáveis declaradas e não utilizadas, um monte de coisa para evitar smells. O que não passa ali a gente também confia que cada desenvolvedor vai atrás de conhecimento, vai atrás de estudar, a gente conversa sobre boas práticas em teste, clean code. Sempre tem talks, a gente compartilha artigos nos grupos internos para o pessoal estar se atualizando sobre boas práticas e também o principal no final, se ainda assim subir algum smell, é pego no code review. A gente tem que ser bem criterioso no code review para passar sem smells. Futuramente a gente vai jogar isso na integração contínua, ainda não está, mas vai subir para um merge e vai rodar no SonarQube, e caso se ainda além de tudo isso subir algum smell, o SonarQube vai acusar e tem já todos os outros times já fazem que é ter uma refatoração continua para minimizar o máximo os smells.

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

E. Uma sugestão é essa parte que eu comentei mesmo. Se a empresa algum pipeline de execução de testes contínuos, testes integrados que eu senti falta. Que mesmo a pessoa rodando teste local tem que ter teste integrado. O seu foco é só teste unitário, não é? Se incluir teste de integração eu incluiria. Mas se é só teste unitário não tem necessidade. Era só essa parte. Integração contínua de testes unitários e quais as ações que a empresa toma se caso tenha problema nessas execuções. Como ela acompanha, onde ela acompanha a execução do teste, se caso ocorrer o erro em testes ou lá no como exemplo, o sonar cube, for acusado problema nos testes, pouca cobertura, bugs, smells. Isso já faz parte do que tu comentou, mas a partir depois da integração contínua se tem algum acompanhamento da equipe nisso, porque depois que sobe para integração contínua é um ambiente que já está em ambiente de homologação e logo já vai para ambiente de cliente e tem que ter muito cuidado com o que se mexe, com o que se refatora e tudo mas para não impactar o cliente final. Só isso. Eu inclusive comentei sobre isso, mas só falta essa parte de DevOps de testes unitários.

P. Muito obrigado. Eu vou encerrar aqui a gravação.