Pesquisador. Muito obrigado por ter aceito participar dessa entrevista. Ela tem como objetivo compreender os processos de criação e manutenção de código de teste de unidade e também a verificação da qualidade desse 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. 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. Há mais ou menos 3 anos numa empresa que trabalhei anteriormente.

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

E. Eu aprendi alguma coisa na faculdade, no mestrado. Pelo menos a parte teórica. Mas na prática a gente não usava muito até eu entrar na empresa anterior e lá a equipe tinha bastante costume de usar testes então acabou virando rotina.

P. Em quais linguagens você cria testes?

E. PHP.

P. Qual plataforma do seu projeto atual?

E. Web. PHP

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

E. Normalmente depois de acabar o desenvolvimento, eu faço a cobertura de testes. Eu entendo que provavelmente deveria ser antes para fazer um TDD, mas normalmente é após o desenvolvimento.

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

E. PHPUnit. A gente também usa o Ghost Spectre para fazer teste end to end.

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

E. Não.

P. Como é o processo de verificação de qualidade do código de teste criado?

E. Não sei se tem algum padrão. Simplesmente tem o code review aprovar os PRs (product releases) e a equipe só aprova se a PR se as novas funções têm cobertura de teste. Não é feita nenhuma verificação de escrita, apenas se o teste está passando ou não. Normalmente são duas pessoas. Um desenvolvedor e um membro da equipe de teste. Eu esqueci de falar, mas a gente tem uma equipe de testes. O desenvolvedor cria os testes em PHPUnit e a equipe de testes fazem testes manuais e escreve os testes end to end.

P. Passando para o processo de manutenção do código de teste, como ele é realizado? Em qual momento os testes de unidade eles são editados, excluídos, como funciona esse processo?

E. Normalmente quando a gente inicia uma tarefa, a gente cria branch nova e roda a suíte de testes para ver se está tudo ok, se ninguém deixou alguma coisa quebrando. Se deixou, aí já aciona o colega e durante o desenvolvimento vão se construindo os testes novos ou adequando testes antes antigos quando tem alguma atualização de código, e aí roda de novo a suíte antes de submeter o PR a revisão.

P. O que você conhece sobre test smell?

E. Test smell... não conheço o termo.

P. Eu vou te passar o conceito, pois as perguntas a seguir têm a ver com esse tema. Então eu vou te passar o conceito, vou te passar uns exemplos de test smell e caso você tenha alguma dúvida você me pergunta e depois a gente segue com a entrevista, ok?

E. Tá bom.

P. “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.” Até aí ok?

E. Ok.

P. Eu vou te mandar dois exemplos, esses exemplos estão em Java mas como se trata de estrutura, eu te explico o porquê eles são considerados smells e seria aplicado para qualquer linguagem. O primeiro ele é chamado de "Empty Test" e ele é considerado um smell porque não tem nenhum script executável. Uma vez tendo testes dessa forma na suíte não vai ter nenhuma identificação de porque ele está comentado ao invés de excluído, prejudicando a manutenção da atividade de teste. O outro é o Conditional Test Logic. Ele é considerado um smell porque existe estruturas de condição e de repetição dentro desse teste, dentro da lógica dele, ele é prejudicial dentro da atividade pois uma vez que esse teste falhe vai ser mais difícil de identificar porque que ele falhou, uma vez que existe condições e repetições dentro dele. Então seria interessante dividir esse teste considerando as condições e repetições que teriam dentro. Existem ainda outros smells relacionados a asserts a valores de parâmetros a configuração, e que entram no mesmo contexto de dificultar a compreensão do teste e qualquer tipo de modificação que precise ser feito. Alguma dúvida?

E. Não. Não tenho dúvida.

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

E. Como eu disse, a gente não tem um padrão ou alguém que revisa ativamente esses testes. Talvez isso seja tratado na hora do code review, a pessoa olharia e perguntaria "o que você quis fazer aqui?". Ele é o que faz que a suíte de teste não tenha testes que quebrem à toa, sempre tem que estar passando tudo, ou se você marcar como “skiped”, que se tenha um motivo do porquê isso está acontecendo. Acredito que seria assim, uma coisa orgânica, que aconteceria de forma espontânea.

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

E. Não é que não é feito, mas seria desse jeito. Se os revisores detectassem algum problema eles solicitariam a revisão para o autor do código e, uma vez revisto, depois que todo mundo estivesse de acordo e passado, aí não voltaria nessa questão a não ser que o código começasse a quebrar no caso.

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. Bom, no momento que ele é criado, eu acredito quer o ideal é que fosse utilizado alguma técnica como o TDD e só se criar testes com propósito claro. Você usa os testes para o próprio desenvolvimento. Você cria o cenário antes de começar a desenvolver e invoca os testes para poder fazer o desenvolvimento. Dessa forma você garante que todos os testes vão estar funcionando e você vai ter todos os testes que você pensou como desenvolvedor ou analista que tenha feito a tarefa pensou em todos os cenários. Você deveria ter todos os cenários codificado em teste para poder invocar os métodos que você está desenvolvendo. Acredito que isso resolveria o problema.

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

E. Talvez falar que no mundo prático, fora da academia, acho que testes é uma questão de bastante disciplina. Se você ou o gerente abaixa a guarda a tendência forte é que a equipe pare de fazer testes. É bem "ocorrente" ver acontecendo em diversas equipes em diversas empresas. Acredito que um processo bem forte de disciplina, de ficar em cima, de sempre exigir a construção de testes durante os code reviews, talvez trabalhar em seminários com a equipe sobre a importância de TDD, mas se é um projeto que corre contra o prazo... a gente já teve caso aqui na empresa de projetos que nasceram extremamente cercados de testes, tinha até pipeline no Cialike que fazia isso antes de fazer o deploy ele rodava os testes de unidade e com o tempo isso foi caindo por terra, priorizando o desenvolvimento. Com o tempo se você não toma cuidado e não tem uma disciplina rígida em cima disso, com o tempo, os testes de unidade acabam morrendo. Mas é uma questão de consciência da equipe da extrema importância disso, porque quando você vai fazer uma manutenção muito grande, uma quebra de base no código mesmo, uma coisa que vá mudar, por exemplo vai mudar uma camada inteira de repositório ou de alguma conexão forte com o banco, tirar alguma dependência forte do projeto, sem testes automatizados é quase impossível fazer isso de forma assertiva, você provavelmente vai deixar alguma coisa passar e impactar no sistema, impactar no clientes, impactar em produção. É uma questão mais de convencer a equipe e principalmente o comercial de que é muito importante gastar uma parte da tarefa na construção de testes automatizados. Acho que essa é uma das lições mais valiosas que a gente aprende fora da academia. No mundo corporativo real.