Pesquisador. Muito obrigado por participar da entrevista. Ela tem como objetivo compreender os processos de criação, de manutenção e verificação 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. Ok?

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

E. Tudo bem

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

E. eu vi a necessidade, por exemplo, eu trabalho a 2 anos e meio como desenvolvedor comum, desenvolvedor mesmo no dia a dia e chegou um tempo na empresa atual que estou que surgiu a necessidade de criar testes, dos testes serem testados automaticamente e aí a gente começou a fazer esses testes. Eu comecei a aprender sozinho, surgiu profissionais capacitados com testes software, e trocando ideias, estudando, comecei a ter conhecimento suficiente para implementar lá. Foi mais ou menos uma decisão a nível de time lá. A gente vive ali, trabalha no contexto de time, parecido com scrum, agilidade e tal. E foi assim. A gente começou a fazer um sistema lá apartado do sistema principal da empresa, que seria um sistema de financiamento para graduação e a gente começou a fazer um sistema de financiamento para outras coisas, como intercâmbio por exemplo. Agora que a gente vai começar a fazer um sistema novo e o legado não tem teste. Talvez essa seja uma informação legal, a maioria dos sistemas legados lá não tem testes, a gente começou a criar um novo sistema com um novo mindset, vamos construir algo com mais qualidade, pensando em TDD, pensando em teste unitário, e foi mais ou menos assim o fluxo, foi uma decisão a nível de time. Os times lá estão tendo mais liberdade para decidir, mas em questão do cto, a parte mais de cima da empresa também está começando a cobrar cada vez mais teste. Uma coisa assim que lá hoje não é tão empregado os testes, mas assim, a cobrança da parte estratégica da empresa é que tudo seja feito testes daqui pra frente.

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

E. Para mim a primeira foi de pessoa para pessoa mesmo. Eu aprendi bastante com colegas de trabalho lá na empresa, aprendi também lendo documentações, por exemplo, lá a gente trabalha com php e a gente usa o framework Laras e a documentação do Laras é muito boa assim de como você vai fazer os testes e tal, então uma leitura dali já te dá uma base melhor. Eu também li livros, eu li um livro na verdade e alguns artigos sobre teste. O livro que eu li é do kent back o xtreme programing, que ele fala um pouco sobre os testes, não como fazer, mas porque dos testes, porque do TDD que é uma prática do XP e os artigos é mais na prática de como fazer os testes de como vai pensar, YouTube também que as pessoas explicam um pouco como é cada teste, para que que cada teste serve e tudo mais, mas e mais ou menos isso.

P. Em quais linguagens você cria testes?

E. atualmente só php.

P. Qual plataforma do seu projeto atual?

E.

P. Como é seu processo de criação de teste?

E. A primeira coisa é que o teste unitário é o teste que a gente testa uma coisa pequena, uma coisa que não está acoplada com outra coisa então primeira coisa a se pensar é como separar várias coisas, ou seja vários componentes, cada pedaço de código, cada função em coisas separadas que a gente consiga testar ela de maneira separada. A primeira coisa é essa parte que talvez seja a mais difícil. Depois a gente tem que pensar em quais casos aquela função vai retornar alguma coisa diferente, então por exemplo, um exemplo bem básico, uma função de soma, ela recebe dois argumentos, se o primeiro argumento e o segundo argumento for dois, ela tem que retornar 4, mas se essa soma o primeiro argumento for dois e o segundo argumento for negativo, por exemplo -2 ela te que retornar 0. Então a gente tem que estar mapeando todas as partes que a gente nunca vai mapear todas as partes que uma coisa vai fazer, é muito difícil, quando a gente mapeia, toda a necessidade que a gente tem naquele pedaço de código, e a gente faz um teste para confirmar que aquilo está sendo feito.

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

E. Não. não existe.

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

E. que já foram utilizadas, teve algumas. Teve o PHPUnit, não sei se se encaixa em ferramenta, mas a gente usou o BDD que é pensar no comportamento para o código, por exemplo, a gente pega uma história, alinha qual o comportamento que aquela história tem que ter, história em ágil mesmo, um pedaço de trabalho, uma jornada do usuário ali. E aí a gente começa a fazer os testes, mas de ferramenta é o phpunit e os frameworks Lumen ou Laravel que ajuda bastante a gente fazer os testes. Já vem com coisas integradas, out of the box.

P. Como é o processo de verificação de qualidade do código de teste criado, por exemplo: verificação da escrita, existe algum processo de que alguém revisa o teste depois que o programador cria, compatibilidade com o requisito...?

E. Como eu disse, para valer a nível de time, eu já participei de dois times lá. Um time, que eu era mais ativo com esse controle dos testes, a gente realmente tinha code review, que é o processo mesmo de por exemplo ter uma pull request para fazer a implementação, a integração com produção e realmente antes disso os desenvolvedores batiam o olho, não era só uma pessoa mas todo mundo batia um pouco do olho para ver onde poderia ser melhorado, onde aquele código poderia ser mais legível, essas coisas. A gente tinha o próprio teste, porque toda hora que a gente roda todos os testes se tiver algum problema por exemplo, já mostra para a gente, por exemplo estou criando uma nova feature, a gente já escreveu diversos testes anteriores a essa feature e essa feature quebra algum outro teste a gente garante que ela não vai quebrar porque daí já não vai subir a aplicação. A gente tinha um code review, a gente tinha peer programming, que tinha 2 pessoas trabalhando para resolver a mesma coisa daí a ideia disso era ter troca de informações, troca de métodos, métrica de pensamentos de como fazer as coisas e acho que é isso: code review, peer programing e testes. Só para deixar claro que é a nível de time. Agora eu fui mudado de time e agora ele está bem atrás do time que eu estava, ele está começando a chegar em ter peer programing da maneira correta, code review da maneira correta, fazer testes, não é uma coisa muito empregada em toda a empresa, é a nível de time.

P. Esse code review que você já trabalhou, quantas pessoas revisavam um código submetido?

E. Obrigatoriamente uma pessoa de qualquer nível do time desenvolvedor, já revisava. a gente abria mais para discussão mesmo. A gente criava pull request ali, mandava no chat e todo mundo revisava, todo mundo tinha a boa vontade de revisar, mas a aprovação de uma pessoa só já subia

P. Como era ou é realizado o processo de manutenção do código de teste?

E. Essa pergunta fica um pouco ampla, na verdade. Porque o que é que a gente tenta fazer: a gente tenta refratora, por exemplo. Quando a gente vai fazer a manutenção ali e a gente percebe que vamos criar um bug ali, a gente tenta alinhar os requisitos ali e refatorar tanto os testes quanto o código (de produção) ali para o funcionamento correto. Eu não sei qual o nível de detalhe você gostaria nessa pergunta.

P. Por exemplo, o teste você identifica uma necessidade de refatoração, é um momento específico para análise desse teste ou é a partir do momento que você precisa realizar uma alteração do código de teste?

E. por exemplo quando a gente identifica um erro, por exemplo. Passou no teste por exemplo e identificou algum erro. A primeira coisa que a gente sabe é que o sistema está errado, mas a segunda coisa que a gente sabe é que o teste também está errado. Porque ele passou. O teste é para identificar o erro, então se ele não está identificando o erro, ele está errado. Então a gente já sabe que tem que refatorar um teste, por exemplo, quando uma feature ela tem que ser implementada, provavelmente algum teste vai precisar ser refatorado, porque a gente usa o tdd lá. Então a gente usa tanto os testes unitários quanto outros testes, por exemplo como os testes de integração, testes funcionais e tudo mais. Tudo a nível automático, então é mais assim, a gente verifica se tem algum erro ou se tem alguma característica que precisa de refatorações no teste para a gente conseguir refatorar.

P. O que você conhece sobre test smell?

E. Não conheço nada. Não conheço test smell. Primeira vez que eu escuto esse termo.

P. Eu vou te passar o conceito sobre o assunto. Vou te mandar 2 exemplos que estão em java mas como o tema é muito mais a ver com a estrutura que com a linguagem, eu acredito que fique fácil de exemplificar, mas caso você tenha alguma dúvida, a partir dos exemplos eu tiro, porque as perguntas a seguir tem a ver com esse assunto, tem a ver com esse tópico test smell, ok?

E. Ta 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.” OK?

E. Ok.

P. vou te mandar os exemplos e a gente conversa um pouco sobre eles. O primeiro é o chamado Empty test smell. Ele é considerado um smell porque não tem nenhum script executável. ou seja, todas as linhas desse teste estão comentadas. então isso prejudica a atividade de teste pois não tem nenhuma informação de porque esse teste está comentado ao invés de excluído por exemplo. isso pode impactar na sustentabilidade da atividade de teste.

E. engraçado, pois a ferramenta que eu uso, se a gente faz um teste desse jeito ela até dá um warning do tipo "você criou um teste, mas esse teste não está testando nada". Então é meio arriscado isso.

P. O outro smell é o chamado Conditional Test Logic smell. Ele é considerado um smell porque ele tem uma estrutura condicional e de repetição e isso faz com que seja mais difícil identificar, caso esse teste falhe, o contexto exato que esse teste falhou, ou ao depender da experiência do desenvolvedor, compreender a lógica desse teste, se for por exemplo um desenvolvedor inexperiente ele pode ter dificuldade para entender a lógica desse teste. Então existem smells especificamente relacionados a forma como foram escritos, como esses dois, mas tem smells relacionados a asserts a valor de parâmetro, a construtor.. todos eles acabam impactando na sustentabilidade da atividade de teste, a partir da complexidade que eles acabam gerando dentro desse código.

E. Entendi.

P. Alguma dúvida?

E. Não, por enquanto estou conseguindo seguir de boa.

P. Como test smells são tratados no processo de criação do código de teste? Você falou que o phpunit detecta um teste que não tem nenhum script executável mas em se tratando de outros itens que impactem na compreensão e na independência dos testes. Como eles são tratados na criação do código de teste?

E. nesse caso como o teste é vazio a ferramenta já verifica e informa para gente o que está acontecendo com o teste. E nas outras questões a gente não tem um tratamento, a gente evita, com boa prática, não é muito bom ficar colocando muita logica no teste, justamente por conta dessa coisa de não saber, por exemplo qual é o estado do teste deu erro, então a gente evita fazer isso, só que hoje a gente não tem uma ferramenta por exemplo que identifica essas situações. É uma coisa de tentar usar o bom senso, tentar não fazer esse tipo de coisa, por exemplo, com certeza, se a gente identificou uma coisa dessas num code review, a gente alerta que o jeito de fazer esse código aqui está errado. Não sei se é um test smell também, testes que são acoplados um com outro. Esse tipo de coisa a gente não deixa passar no code review, mas já teria uma ferramenta que vai apontar automaticamente isso para gente.

P. Isso acontece tanto na criação quanto na manutenção?

E. Sim.

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

E. Eu acredito que seria uma ferramenta, por exemplo, eu não sou especialista em teste, por exemplo, que possa ter uma ferramenta que possa identificar, que tenha alguma complexidade que ele não possa ter ou que ele tenha um acoplamento contra um outro teste por exemplo. Eu acho que que seria legal. Eu sei que uma coisa que a gente está implementando lá, é que antes de mandar códigos novos para produção, não sei se estou confundindo aqui, mas é uma coisa tipo avaliador de código. Ele pega assim, com chegar a essa informação. Ela aumenta ou diminui a complexidade do sistema. Eu acho que uma ferramenta assim seria legal. E capacitar os profissionais. Porque para eles, na empresa que estou estamos ainda muito cru nessa questão de testes. A maioria está começando a aprender teste agora com a necessidade, não tinha essas experiências. Eu sou um exemplo disso. Eu sou a nível jr para pleno, então tenho 2anos e meio de desenvolvimento, então comecei a mexer com teste de software a um ano atrás, então um ano mais ou menos de experiência, eu acho que, o que me faria por exemplo a não cometer tantos erros, seria me capacitando. Tanto eu mesmo me capacitando quanto a empresa me capacitando.

P. Estou chegando ao final da entrevista, você tem alguma pergunta para mim, quer acrescentar alguma informação ou ainda tem alguma sugestão para melhorar nossa entrevista?

E. Não, não chego a ter dúvida não, achei interessante as perguntas, posso ate dar um feedback, achei as perguntas bem colocadas, essa questão de test smell eu não conhecia, foi interessante também.

P. mais uma vez eu te agradeço por participar da entrevista. Suas informações vão ajudar bastante no mestrado.