Pesquisador. Gravação iniciada. Muito obrigado mais uma vez por ter aceito participar da entrevista. Ela tem como objetivo compreender os processos de criação, 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. Fechou.

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. Tudo bem?

E. Fechado

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

E. Eu tenho mais de 20 anos de profissão e quando eu comecei lá atrás, o gerente de software era inimaginável em pequenas empresas. Qualquer tipo de método de engenharia de software era muito raro, poucas empresas faziam. Hoje em dia empresas pequenas, as vezes andam até muito mais rápido que empresas grandes, mas na época que eu comecei não tinha nenhum contato com isso, e aí eu trabalhei muito tempo fazendo aplicações comerciais com Clipper, isso remonta lá a década de 90, e quando eu migrei do Clipper para o Delphi eu adquiri um livro chamado code complete, segunda edição desse livro. Era um livro na época um livro da Microsoft e caiu na minha mão, mas esse livro mudou a minha mentalidade absurdamente muito cedo.. Dei muita sorte de pegar ele no começo da minha carreira porque lá atrás, no começo da década de 2000. Eles já falavam sobre isso, eles já falavam sobre essa questão. Claro que a gente está falando da Microsoft, mas uma Microsoft mais errática naquela época do que hoje em dia. Mas o livro ele tinha entre outras coisas falava sobre teste automatizado. Paralelamente a isso, eu também tive contato com blog na época de um cara que depois veio a se tornar o fundador do Slackoverflow e criador do Trello, chamado Joel Spolsky, não sei se você conhece, mas é um cara super visionário e em 2002, 2001 ele já blogava sobre teste automatizados, sobre teste de usabilidade, ele blogava sobre integração contínua em 2001. O que é hype hoje em dia com DevOps em 2008, o cara falava sobre isso em 2001. Eu tive sorte também de ter contato com isso muito cedo e por ter contato com isso muito cedo eu comecei a aplicar também muito cedo ou tentar aplicar. Partiu de uma curiosidade minha pessoal de aprender. E aí obvio que eu cheguei no Kent Back, no famoso livro dele de TDD e aí foi onde a coisa começou a ficar séria, com o advento de XP, e do TDD, de aplicar isso, principalmente teste de unidade, de aplicar a nível profissional. Então passou por um momento meu de descoberta, de amadurecimento como profissional, por ter acesso a literaturas voas, por conhecimento do Kent back e em princípio das metodologias ágeis e aí eu não parei mais. Mesmo quando eu parei em empresas que não fazia, eu fazia.

P. Você citou algumas, mas quais foram suas fontes de aprendizado sobre código de teste?

E. A principal delas, eu já citei. Foi o Kent Back com o livro de TDD, depois eu tive muitas outras leituras, todos os livros do uncle Bob, por exemplo que bate muito nessa tecla, mas para mim o marco fundamental foi o Kent Back e o livro TDD. Teve uncle Bob, teve coisas da própria Totus com o Fowler, mas eu diria que com Kent Back que eu peguei gosto pela coisa e aprendi a fazer. Os livros do uncle Bob, você fica bem empolgado, ele é um bom escrito, mas eu já estava comprado na época que eu peguei os livros do uncle Bob. Eu não precisava dele para me ensinar TDD. Eu usei a literatura do uncle Bob para outras coisas, para design de sistemas, mas TDD eu já estava comprado já.

P. Em quais linguagens você cria testes?

E. teste de unidade hoje em dia eu uso o .NET. Majoritariamente eu uso C#. Escrevo muito código em Python, Javascript, Typescript, mas é uma relação de 80/20 C#. Aí tem Ruby também, mas a maior parte do tempo C#.

P. hoje você trabalha com qual linguagem especificamente?

E. C#. É aquela coisa, agora mesmo estou escrevendo teste em Python, antes da gente começar, mas eu estou citando C# que é a maior parte do meu trabalho está em C#

P. Como é seu processo de criação de teste de unidade?

E. Se eu estou mexendo em alguma coisa que eu vou refatorar, que acontece bastante, eu trabalho em cima de uma base de código de mais de 20 anos. Então eu faço muita coisa nova, mas eu trabalho em cima de um legado absurdo. E aí se eu vou refatorar e o código não tem teste algum, eu crio os testes antes de refatorar, antes de mexer em qualquer coisa, sempre coloco nos meus orçamentos um tempo para testar antes para escrever os testes antes. Então eu escrevo os testes, mexo e eu uso os testes para apoiar o desenvolvimento. Se eu estou criando alguma coisa nova, o processo é o mesmo, só que eu cheguei num ponto hoje em dia que eu escolho o que eu quero testar. Te dar um exemplo. Eu trabalho numa empresa bem bacana que faz sw para atendimento ao consumidor. Eu chamo de Consumer Care, mas basicamente as empresas usam a solução para atender os seus consumidores. Uns programas bem legais de escala, mas tem muita coisa que a gente faz que é aquele bê a bá, crude e quando eu pego uma coisa dessas, eu hoje em dia eu escolho o que testar, especialmente teste de unidade. Se eu vou pegar alguma coisa que não cai muito nesse básico, aí eu escrevo teste para aquilo que eu acho que precisa de garantia. Eu hoje eu procuro menos cobertura, numa fase diferente que tive outras vezes. Eu sempre meço a cobertura, isso é importante dizer, mas uma cobertura alta para mim não significa necessariamente qualidade. Posso até ter uma cobertura alta e não testar o que importa, então hoje eu tento identificar e escrever os testes antes, mas daquilo que para mim eu considero, eu julgo importante.

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

E. Não existe.

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

E. Como eu uso .NET eu uso algumas bibliotecas. Uso uma biblioteca de teste chamada NUnit, que é uma biblioteca de testes do .NET que é o mesmo porte da unit do Java para .net e aí eu uso algumas ferramentas para apoiar o teste que chama-se Fluent Asserts. Que é uma ferramenta para você escrever as asserções dos testes de forma fluente. Quebra um galho isso, especificamente quando você tem que lidar com asserções um pouco mais complexas. Eu trabalhei agora num projeto a um tempo atrás que envolvia grafos fortemente e aí as operações, quando você sai de uma operação, o resultado as vezes é um subgrafo e fazer asserção em grafo é uma coisa tediosa, que você tem que checar vértice por vértice, aresta por aresta e aí você no fundo você quer saber que aquele subgrafo resultante da operação é o que você espera, é super tedioso fazer isso na mão e aí você tem umas bibliotecas que te auxiliam a escrever esse tipo de coisa, ajuda. Que é o caso desse Fluent Asserts, mas é basicamente isso. Para escrever os testes não tem uma ferramenta específica, tem bibliotecas. Aí para medir, que é o pós, quando o software está no pipeline de distribuição dele, eu uso atualmente uma ferramenta chamada SonarQube que ele, depois dos testes executados, ele mede a cobertura dos testes em relação ao código.

P. Como é o processo de verificação de qualidade do código de teste criado, por exemplo: verificação de escrita, se tem um code review, se uma pessoa revisa o código de outra, verificar compatibilidade com o requisito? Você falou sobre o Sonar, o Fluent Assert para melhorar a escrita, mas além disso, como funciona essa verificação da qualidade do código

E. Atualmente eu trabalho com pull request e todo o código é passível de revisão, inclusive o código de testes. Quando eu digo que eu escolho testar, esse juízo ele é colocado a prova numa revisão de teste, numa revisão de código. Então existe um processo de revisão de código no pull request e o código de teste está envolvido nisso. Da mesma forma, a análise do git de qualidade do código. Código de teste também é código então está sujeita a todas as práticas de todas as boas práticas de codificação que a gente usa no nosso código de aplicação. Não há diferenciação. Não é aceitável um método com mais de X linhas, só porque ele está no projeto de testes, então se a gente define no quality gate que um método tem que ter no máximo 30 linhas, esse critério de avaliação também é aplicado no código de testes.

P. quando você fala que esse teste é submetido a uma avaliação você quer dizer que ele passa por um desenvolvedor analisando ou seria essas ferramentas?

E. As duas coisas. O review é feito por um humano, e isso faz parte do pull request. Depois do pull request esse código é mergeado, mesclado no branch que vai para produção e aí para ir para produção, no pipeline de distribuição na esteira de distribuição ele é verificado pelo SonarQube, no meu caso hoje em dia. então se ele ferir alguma regra do git de qualidade, ele é barrado. Mesmo que ele tenha passado no pull request.

P. Quantas pessoas fazem essa revisão do pull request?

E. hoje em dia uma pessoa.

P. Como é o processo de manutenção do código de teste? Em que momento você realiza edição, exclusão de código de teste?

E. Esse é um aspecto interessante. Os requisitos eles mudam o tempo inteiro, isso faz parte do nosso processo aqui. Ainda ontem eu tive uma reunião sobre isso e o tempo inteiro a gente se fala sobre isso. Não existe uma regra que alguém não possa mexer num código de teste para excluir um teste, o mesmo para mudar um caso de teste. Da mesma forma como um teste nasceu ele vai ser avaliado na alteração. Então a gente acredita muito na severidade das pessoas envolvidas e a gente não tem um processo rígido em cima disso. Passa pela revisão como qualquer outro código. "você removeu um caso de teste, porque removeu?", já aconteceu de alguém comentar numa revisão do código meu: "você removeu 3 casos de teste, porque?" E ele não viu o código. Se eu não fui claro no meu commit ele já bate e volta. ele não é analisado.

P. O que você conhece sobre test smell?

E. Nada. Com esse nome? Nada. Eu já ouvi bad smell, code smell, mas test smell, não.

P. Eu vou te passar o conceito. Ele está relacionado aos code smells os test smells descendem dos code smells. Caso você tenha alguma pergunta a gente alinha esse conhecimento, sobre o assunto, porque as perguntas a seguir têm a ver com esse tópico, tudo bem?

E. Tudo bem.

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.”

E. Está bom.

P. vou te mostrar 2 exemplos só para fixar mesmo.

E. No fundo eu não conhecia a expressão, mas ontem mesmo eu tive uma reunião sobre isso. Consigo até de dizer o exemplo. Se quiser nem precisa me mostrar.

P. Beleza

E. A gente está com um problema muito grande no teste de unidade, teste de ponta a ponta, mas o teste está produzindo um código muito difícil de manter. Aí a gente foi ler o código e identificou-se lá que eles usam framework para fazer a automação do browser, nem lembro o nome do framework, mas é algo tipo Selenium, e como o design muda o tempo inteiro, o código de teste está junto com o código do framework, então se mistura um pouco o que é coisa específica de infraestrutura para o teste do código de teste em si. Quando estava lendo o código eu identifiquei a necessidade de aplicar alguns padrões, anticorrupção (anti corruption layer). Aí sentei ontem com o time e a gente definiu de aplicar esse padrão e definir que é o modelo do teste para separar o que é código testável e com o modelo desse teste e de código que tem a ver com a estrutura do teste. Está claro para mim o conceito, eu só não tinha escutado ele para testes. Code smell a gente escuta o tempo inteiro. Se você tivesse dito "Code smell para teste" eu tinha te dado esse exemplo, mas beleza. Obrigado pela informação.

P. É que hoje existe uma suíte de test smells distinta de code smells. E...

E. Estou vendo aqui. É bem bacana. Depois eu vou dar uma fuçada até. Test smell detector, Test smell search, bem legal mesmo.

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

E. Eu acredito muito em educação. Educação no sentido de mostrar práticas que são comprovadas, isso para mim não tem muito a ver se é código de teste ou código de produção. As pessoas que escrevem são as mesmas pessoas, no caso do teste de unidade. Então quando a gente vai falar sobre code smell a gente fala de uma forma mais voltada para design de sistema e tudo que a gente fala de código de produção se aplica a código de teste, então por exemplo, se a gente fala de scob no código de produção, a gente também quer scob no código de testes. Se a gente fala de responsabilidade única, a gente também quer responsabilidade única no código de teste. Então para falar a verdade eu nunca pensei em analisar antes de ver isso aqui que você me mostrou, nunca pensei em analisar padrões específicos para o código de testes, sabe? Então minha resposta para você é: eu trato igual hoje em dia. Igual ao código de produção. Então quando eu analiso smell eu analiso smell igualmente. Tanto o código de produção quanto o código de teste.

P. Vocês usam ferramenta para isso?

E. As regras que a gente usa no Sonar hoje elas não batem em questões de design de sistema.

P. Isso vale tanto para criação quanto para manutenção?

E. Vale para tudo

P. Você comentou um pouco em relação a educação do profissional, mas você teria algum outro tipo de sugestão que você identifica como possível para prevenir ou evitar a inserção de test smell no momento da criação desse código

E. Eu sou meio cético em relação ao uso de ferramentas que sirvam como guard rail para durante o desenvolvimento, mas isso é uma característica minha. Eu entendo que uma das formas de fazer isso era aplicar ferramentas para identificar isso muito cedo, enquanto a pessoa está desenvolvendo. A gente só não usa. Hoje em dia como eu estou numa posição hoje de liderança de time, acabo não investindo nisso porque não acredito nisso, mas tem um pouco a ver com as experiências que eu tive que no fundo eram ferramentas muito rasas, e um pouco do que eu acredito pessoalmente sobre a profissão de desenvolvimento de software.

P. Então nesse caso, só para deixar formalizado, para você, a forma que você enxerga para prevenir ou evitar seria o aumento do conhecimento do profissional...

E. A qualificação. Sem dúvida. Prefiro perder tempo nisso do que investir tempo em ferramenta, sabe? Eu acredito que o design da ferramenta não pega tudo que deveria pegar. As ferramentas que existem hoje pelo menos.

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

E. Não. Muito bacana. Só agradecer por ter entrado em contato e desejar boa sorte para você na conclusão de seu trabalho.

P. Muito obrigado.