Pesquisador. Obrigado por ter aceito participar dessa entrevista. Ela tem como objetivo compreender os processos de criação e de manutenção de código de testes e a qualidade do código de teste é realizada nesses dois processos. 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. 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 comecei como desenvolvedor por uns 8 anos e foi onde eu comecei com tecnologia. E na empresa que eu estava surgiu uma oportunidade de ser líder da equipe de testes porque o coordenador de testes da época tinha saído. E eu vi como oportunidade de assumir uma liderança que era o que eu desejava naquela época. Daí eu me candidatei internamente, foram feitas as entrevistas com outras pessoas lá e fui escolhido para ser coordenador. Só que eu não sabia nada de testes. Tanto que, uma coisa interessante, eu dava algumas palestras antes sobre a parte social da empresa e não testes, não tecnologia, mas a parte social da empresa e o coordenador de testes antigo foi dar uma palestra comigo. Ele ia falar uma hora e meia, e eu pensei "o que você vai falar uma hora e meia?" Que ignorância a minha! "o que tem tanto de testes para falar uma hora e meia?" Tanto que eu dei um curso depois de 140 horas e é pouco ainda. É básico para teste de software. Então eu entrei em testes, não sabia nada. Comecei a fazer alguns cursos comecei a participar de eventos, comecei a ler muito e até escrever alguns artigos do que eu ia aprendendo, e nisso eu comecei a evoluir. Eu gostei da coisa, "está aí, legal". Eu gostava de desenvolvimento, com isso foi muito fácil de eu ir para a área de automação, porque eu gosto ainda de desenvolver, eu pego algumas coisas ainda para desenvolver. A ideia da empresa era implantar a área de automação na empresa que não tinha. Tinha uma equipe de 6 pessoas na época e eu fui em busca disso. Era uma empresa waterfall no processo, bem tradicional. E eu comecei a montar a equipe e os processos, pois não tem processos bem definidos. Aí passei para modelo V, pela montagem da equipe. Fiquei 4 anos e meio lá fazendo tudo passo a passo. Nada foi atropelado. Eu tentei ser atropelado, pois eu estava muito empolgado, mas quando eu saí de lá estava tudo implantado o que eu tinha previsto no início. Demorou uns 4 anos e meio para implantar tudo que a gente queria. Era teste funcional, era só teste de sistemas, não tinha testes de integração. Tinha teste de banco de dados, mas não tinha a integração em si. Eu até desenvolvi uma ferramenta na época que fazia os testes automáticos do banco, era tipo um unit teste para banco. Não era um unit test, mas era mais ou menos. A gente conseguia testar aspects do banco. Então isso se tornou muito mais fácil para a equipe. Aí depois de 4 anos e meio eu saí de lá. Eu entrei lá em 99 e fiquei até 2007 como desenvolvedor e de 2007 a 2011 fiquei como coordenador de testes. Então como coordenador já estava totalmente empolgado. Tiveram algumas pessoas que me apoiaram. Eu participava muito da comunidade, a equipe de uma empresa estava surgindo naquele momento no Brasil, hoje até não existe mais, mas me ajudou muito naquele momento para meu crescimento. Me ajudou bastante a tirar a certificação naquele momento, o SBTS da LAX e o CTFL do ISTQB. Essas duas certificações me ajudaram, elas não são essenciais, obrigatórias, mas o estudo me ajudou. O estudo para elas me ajudou a crescer, como outras pessoas: José Correia, que me deu uma força enorme. O Cristiano Caetano me deu uma força enorme na época. Então tem algumas pessoas que me ajudaram e eu comecei a me desenvolver a partir daí.

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

E. Quando eu comecei, a gente implantou na empresa H uma aplicação com test complete. Era desktop, não era web e a gente não tinha o inion que é free para usar no Java e tudo mais. Eu comecei a usar o test complete. E lá eu não achava documentação na internet sobre padronização de patterns de código de automação. Só que eu tinha um receio de mudança, mesmo em desktop, existia os IDs, existia os nomes lá que precisavam ser identificadores de cada elemento. E eu tinha medo de mudar um id e quebrar todos os testes. Então mesmo sem saber que era isso, eu comecei a usar Page Object. Eu criava classes de cada tela, cada item mapeado e esse mapeamento ficava preso. Eu até pensava em escrever um artigo sobre isso, mas eu não sabia o nome para isso. Não tinha, porque eu não conhecia e eu não procurava fora do Brasil na época, era 2008 mais ou menos e eu não achava nada aqui que tivesse isso. Então quando surgiu, que eu conheci o Page Object, aí eu acho que foi meio que natural eu ir por esse caminho. Usando classes, mesmo o desenvolvimento da coisa era em VB script e VB script não tinha classe, então eu usava meio que uma parte de classes, uma parte de objetos. Não existia encapsulamento, isso não tinha em VB Script. VB 6 na época. Mas eu fazia as classes separadas de cada página e tudo mais e começou a vingar. Só que era muito custoso para os QAs fazer tudo isso na época em VB6. Eu até fiz algumas ferramentas que ajudavam e tudo mais, e depois até que morreu isso porque estava muito demorado. A partir daí eu segui esse padrão. Até que eu fui para linguagem orientada a objeto. Eu comecei a trabalhar em outra empresa, um data center, e lá era Ruby. Daí a gente começou a desenvolver em Ruby lá. Eu entrei como especialista, não era coordenador, não tinha automação, mas a tendência era automatizar. Eu lembro que meu gerente na época já falava do Cucumbner e eu não via no mercado brasileiro falar de Cucumber ainda e ele já estava falando. E isso já existe desde 2002, 2003 o Cucumber e ele já falava isso que era meio que uma padronização de código, que era o Gherkin lá para padronizar o código, não o código efetivo da automação, mas a camada que comunica com o cliente. Então daí começamos a olhar mais para o Gherkin. Só que o Gherkin não dava valor ainda, porque as pessoas eram muito técnicas, o PO (Product Owner), os desenvolvedores, então começamos a fazer sem o Gherkin que era uma camada a mais, só que hoje eu enxergo que o Gherkin mais ajuda que atrapalha, não é uma camada a mais que atrapalha, ele ajuda mais. Só que a primeira vez que se escreve que é ruim, mas depois você pode reaproveitar muita coisa, mas na época a gente só fez um projeto com o Gherkin. Tinha toda a padronização com o Page Object, do steps, Page Object, que é o padrão de automação web e de API também, mas só em um projeto que a gente usava Cucumber porque tinha valor. Então a gente usava para o que dava valor, seguir o padrão, os patterns de desenvolvimento. O que a gente cresceu muito foi quando a gente começou a usar a linguagem, olha o que interessante, lá eu assumi a coordenação de testes depois de um ano e meio, hoje cresceu muito a equipe hoje tem 13 pessoas nesse data center e no início a gente começou a automatizar. A gente contratou um especialista no mercado, um cara muito bom, mas ele só conhecia Java e lá os desenvolvedores eram todos em Ruby, 95% Ruby e 5% Java. Aí ele falou que se fosse para desenvolver em Ruby ele iria demorar mais pois não conhecia e teria que aprender. Mas em Java ele tinha um framework pronto e que poderia fazer. E eles optaram por Java. O que acabou sendo uma grande besteira. Ele fez uma automação do SAP, da parte web. Ficou muito bom, mas depois de 8 meses ele saiu da empresa e aí tinha uma automação muito boa, a gente sabia mais ou menos executar a automação, pois como eu não estava na base, eu não estava pegando como a coisa estava sendo feita, dar manutenção nele ficou muito difícil e os desenvolvedores falavam que não mexiam em Java. Resultado: perdeu-se a automação inteira. Foram 6 meses de automação perdida, estava pronta, era muito legal, mas que se perdeu porque foi feito em Java. Depois disso eu falei "esquece Java, vamos para Ruby" e assim e passamos para Ruby, a gente cresceu muito porque os desenvolvedores passaram a ajudar a gente. E eles começaram a mostrar os patterns de desenvolvimento para a gente. Teve treinamento de Ruby, de Rails e começamos a usar as melhores práticas de desenvolvimento no teste. Sempre utilizando a regra do Page Object.

P. Em quais linguagens você cria testes?

E. Eu crio testes sempre na linguagem que o desenvolvimento está trabalhando. Então eu já criei testes em Phynton, em Ruby, C# .NET e Java. Depende do projeto.

P. Qual plataforma do seu projeto atual?

E. Java, web, mas responsivo.

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

E. Lá eles não usam TDD, é feito depois do desenvolvimento, mas está funcionando muito bem no projeto que a gente está. A gente tem o score, que olha para todos os testes, então tem o teste de unidade, depois passa para o Sonar, para o Veracode, passa pelo teste de integração ou sistema, depende do projeto, e teste de performance. Apesar de eu achar que não precisa de teste de performance na esteira. Teste de performance viria em outro momento, mas eles colocaram na esteira. Aí a gente faz um pequeno teste de performance na esteira no ambiente que tiver. E tem o score. Ele soma algumas notas e dá um score total do projeto, para subir em produção automaticamente, para ter o CD (continuos delivery) além do CI (continuos integration) automático para a produção sem precisar de uma intervenção. E o nosso score é o mais alto de lá. Então a galera dos SREs (não soube dizer exatamente o significado da sigla, mas seria a pessoa do DevOps que participa da administração do sistema) vieram parabenizar a gente. Isso porque tem desde o início, desde o teste de unidade a parte de automação até o de performance está tudo completo e bem feito e bem coberto.

P. Existe algum fluxograma, template... que padronize esse processo?

E. Lá é muito grande. Como eu trabalho com consultoria, não é na minha empresa. Não sei te falar se existe, mas como tem a pipeline e é obrigatório algumas coisas, não tem como não fazer. Não tem um fluxograma, mas é obrigado a ter o Sonar, é obrigado a ter o Veracode, você é obrigado a ter teste de unidade, um pull request, mesmo que não seja olhado. O processo é obrigatório. Não tem como não fazer. Pode não olhar, pode colocar para ignorar no Sonar, está furando algo que deveria ser feito.

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

E. Como é Java a gente está utilizando o JUnit. Algumas pessoas utilizam o TestNG. O testNG é legal para fazer testes concorrentes, mas não houve necessidade ainda, então a gente faz tudo pelo JUnit mesmo. Uso também o Powermockito, o mockito em si, para fazer os mocks.

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

E. Existem 3 níveis: o code review, que é uma outra pessoa que olha o código e vê, inclusive eu vejo os códigos dos desenvolvedores e até critico "está faltando um teste aqui. Isso não está sendo testado". A segunda fase vai ser executado o teste de unidade na esteira. Ele vai "buidar" e executar os testes de unidade na esteira. A terceira fase é o Sonar, ele verifica algumas vulnerabilidades e alguns bugs no código e robustez de código. Depois passa pelo Veracode para segurança. Ele é uma ferramenta paga, muito boa, e o .JAR sobe para o Veracode, ela faz uma varredura e mostra as vulnerabilidades que ela tem. Não é um pen test porque não é a parte web ainda, mas ele mostra se tem uma biblioteca falha, se tem uma versão mais nova, então pega todas as vulnerabilidades de código dentro e te retorna. Olhando para código, é isso que é feito.

P. Só confirmando... Esse processo é apenas para o teste de unidade ou para o código todo?

E. É para os dois. Na verdade, a gente remove os testes de unidade da verificação do Sonar, mas mantém na verificação do Veracode.

P. Quantas pessoas fazem o code review?

E. A gente inicialmente colocava duas pessoas para analisar, mas vimos que não precisava e agora só uma pessoa.

P. Passando para o processo de manutenção do código de teste, como ele é feito na empresa?

E. Eu vou falar no projeto pois mesmo dentro da mesma empresa, pode ser diferente. No projeto é por demanda. A gente tem uma cobertura de 80% mínima. O nosso chega a 94%. Não existe uma obrigação de ter 80% sendo que os 20% que falta pode ser o mais importante. A gente vê onde tem regra de negócio, onde tem regras importantes e isso o time está bem maduro para fazer. E ali que a gente foca a maior parte dos testes. Alterou alguma regra, o teste ali não vai estar 100%, então a gente cobre aquela parte em 100%. Onde tem uma regra mais crítica, que houve várias modificações de regra, é onde a gente olha. A gente olha no 80/20 mesmo, vendo os 20% mais importante que é ali que a gente tem que focar a maior parte dos testes. Quando tem essa parte a gente faz 100% cobrindo tudo que é possível. Se mudou uma regra, vai quebrar os testes. Como está 100% a classe, vai quebrar. Ou se não quebrar, a classe não vai estar 100% e aí a gente também cobre com 100%. Então é essa a ideia. Sempre olhando onde tem regra de negócio. Getter and setter não faz sentido ficar colocando teste em Getter and setter, se não tem nenhum a regra ali. Agora se tiver alguma regra ali, aí sim vale a pena. Mas como na maioria não tem, a gente não olha, a gente deixa. Esse é o 5%, 10% que a gente nem faz. Não precisa.

P. O que você conhece sobre test smell?

E. Test smell, o Sonar fala muito para a gente sobre os códigos smell e entra no meio de tudo. A gente tenta zerar os smell codes lá. A gente tenta zerar tudo. Às vezes surge, a gente tenta zerar de novo os smell code. Não é o foco principal, pois o foco é a entrega e garantir que não tem bug, mas sim. De tempos em tempos a gente olha e mata todos os smell codes que tem.

P. Como você conheceu este assunto?

E. Pelo Sonar. A partir do momento que o Sonar começou a mostrar, não sei se estamos falando a mesma coisa. Ele passou a mostrar os códigos smell que são alguns códigos que estão meio ruim então, os testes em si, o Sonar faz, a gente corrige e cobre com teste de unidade. O Sonar já te fala quais são, pois a gente não sabe todas as regras smell que podem acontecer. O Sonar tem mais robustez, mais conhecimento da comunidade para identificar isso.

P. Você possui alguma dúvida sobre o assunto? Sobre test smell?

E. Eu acho que não. Se for sobre isso que estou imaginando.

P. É porque as perguntas a seguir têm a ver com test smell. Eu vou te mandar um exemplo de teste com smell e depois a gente continua, pode ser?

E. Ok. Tudo bem.

P. Esse exemplo que eu te mandei, ele é chamado "Conditional test Logic". Esse smell é considerado um smell porque ele tem estrutura de repetição ou condicional dentro dele. E isso dificulta a compreensão, uma vez que esse teste falhe para saber em que momento da execução esse teste falhou. O ideal é que para cada condição desse teste seja criado um teste específico para que no momento que el falhe, seja muito mais fácil a identificação de que em qual momento esse teste falhou. Tem a ver com o conhecimento que você já tem? Tem alguma coisa diferente?

E. Eu concordo com isso, mas às vezes nem tudo precisa separar. Vamos pensar na parte de integração. Eu tenho um example. Eu não uso um example, eu uso uma tabela a parte. Se eu colocar a informação de qual teste está falhando apenas por uma mensagem no console, já resolve o meu problema. Eu não precisaria fazer um teste para cada item. Apenas por uma mensagem eu conseguiria. Concordo com o test smell, mas as soluções, depende muito. É que só uma mensagem no console indicando qual teste que estou fazendo dentro do “for”, já não tem test smell. Para mim, isso é um smell code, é muito difícil ler esse teste. É interessante que lá a gente usa os AAA: Arrange, Act e Assert. Então a gente consegue dividir bastante isso para diminuir esse quesito.

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

E. A gente tenta evitar isso com esses AAA que eu te falei: O Arrange, a preparação, Act a ação e Assert. Então normalmente a gente olhando para isso fica mais fácil, além de mocks. Usando os mocks a gente consegue diminuir um monte de coisa que teria que ser feita.

P. Como eles são tratados no processo de manutenção? Existe alguma diferença que no processo de criação?

E. Não. É refatorado quando for mexer nele. Senão a gente não fica focando nele. Ele não está dando valor do jeito que está por enquanto? É mesmo! É um smell para uma solução inicial, está cobrindo o que tem que cobrir e está dando valor, a gente mantém. Começou a ter algum problema de valor, tipo, está ficando muito demorado nossos testes e tem que melhorar, aí a gente começa a refatorar o que tem que ser feito. Ou mexeu numa regra e tem que mexer nele, aí a gente já refatora para ficar da melhor forma. Então é por demanda. Tem poucos test smells lá, como é muito pequeno, a gente tentou enxugar muito então a manutenção é baixa. Quase nem tem. E quando tem, vamos dizer, vale a pena fazer nesse momento? Então a gente faz quando valer a pena fazer, que der valor. Se não dar valor, a gente fala "Deixa para outro momento".

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

E. Ter claro para você quais testes você quer fazer. Se você tem essa clareza, é o primeiro passo. Depois é usar as técnicas que a gente já comentou, os AAA, usar mocks e focar em dividir, deixar bem claro os testes que você quer fazer. É isso.

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

E. Não, de boa. Está tranquilo. Uma coisa que poderia falar, comentar mais sobre TDD. Que é uma coisa que é interessante em teste de unidade. Apesar de não usar. Não sei se é o seu foco, mas é uma técnica também que evita o smell code.

P. Como cada empresa trabalha de uma forma, eu acabo não falando sobre o TDD e também o foco da minha pesquisa é na qualidade do código de teste de unidade, mais especificamente sobre test smell. Então ele pode estar incluído dentro do TDD como ele pode não está. Como tem empresas que não conhecem o termo test smell, mas ela já trabalha com a refatoração, com a verificação disso sem conhecer o termo. Então como o meu foco é específico na qualidade do teste de unidade, eu não entro nesses outros assuntos, para deixar algo mais enxuto.

E. Beleza.

P. Muito obrigado pela sua participação.