Pesquisador. Muito obrigado por 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 de unidade. Não há respostas certas ou erradas ou desejáveis ​​ou indesejáveis. Dessa forma, senta-se à vontade para dizer o que realmente pensa.

Entrevistado. Obrigado pela oportunidade de participar.

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 no 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. A minha experiência atual desde o início, diretamente relacionada com desenvolvimento de software, então, especificamente falando em teste, a minha experiência é em teste unitário e em teste de integração.

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

E. A primeira das minhas fontes foram fontes acadêmicas, primeiramente durante a graduação, onde eu aprendi por qual motivo se deve realizar testes de unidade, mas isso não me impediu de buscar fontes externas através de artigos, blogs especializados em desenvolvimento com foco em qualidade de software e vídeos. Depois durante o decorrer da minha carreira de desenvolvimento, eu aprendi com meus pares de equipe para garantir a qualidade do código.

P. Em quais linguagens você cria testes?

E. meu primeiro contato foi em java, mas atualmente em C#.NET.

P. Qual plataforma do seu projeto atual?

E.

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

E. Durante meu período de graduação, onde a gente tem oportunidade de ter o primeiro conhecimento e durante um curso que eu fiz a parte sobre teste de unidade, eu aprendi que é importante trabalhar com o TDD onde primeiro você cria o teste, você trabalha em cima do teste que você desenvolveu, criando um novo código e depois refatora. E esse é um cenário ideal em que ele evita que o software tenha alto acoplamento e baixa coesão. Você garante que você consegue isolar um teste de unidade separadamente com pouca ou nenhuma dependência que você possa ter, dependendo o mínimo possível de outras classes, de conexão com o banco entre outros fatores que mitigam dependências. Esse é cenário ideal para promover o melhor desenvolvimento do código. Essa é a teoria, mas na prática eu não encontrei nenhum processo que dissesse que a gente tinha que usar TDD. Minha experiência, a que eu já tive, é que o código já existe, já está sendo utilizado e aí após ser desenvolvido, testado normalmente, o teste unitário é criado para garantir que aquele código não vai ser quebrado, mas geralmente gera algumas questões, como é que eu vou separar o teste da aplicação, como é que vou isolar o teste para poder ter menos dependência possível, entre outras coisas. Aí você já começa ter teste de integração porque já começa dependência.

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

E. A Microsoft disponibiliza alguns documentos de referência para criação de teste de unidade. Em termos de empresa eu nunca vi documentação para teste de unidade.

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

E. A própria interface de desenvolvimento é utilizada para criar testes de unidade, ou seja, temos um conjunto de projetos e esse é um cenário que a gente tenha criar um teste de unidade para um componente. A gente cria um outro projeto de teste, direcionado a esse teste inicial. No Java tem a IDE do Eclipe. No C# tem o Visual Studio. A gente tem o NUnit. A gente tem o mock para criar instâncias fakes, criar os objetos para fazer os testes de unidade.

P. Como é realizada a verificação de qualidade do código de teste criado, por exemplo: verificação de escrita, alguém revisa o código de teste depois de criado?

E. O cenário ideal é que todo o código gerado seja revisado por alguém mais sênior ou seja independente de ser teste de unidade ou não, ele precisa ser revisado. Toda vez que crio um código e submeto para o repositório de desenvolvimento, ele só deve ser aprovado, se ele foi revisado. Se houvesse alguma falha, em termos de requisito que alguém achava que estava errado ou qualidade de código ela solicitado e aprovado se eu corrigisse. Normalmente de 2 a 3 desenvolvedores faziam essa revisão. No Visual Studio toda vez que eu fazia o commit eu registrava quem faria essa revisão de código para mim. Eu assinalava de 2 a 3 desenvolvedores, já existia uma regra pré-definida de quais programadores e daí até o final da sprint era executado e ele só subia depois de revisado.

P. Como era realizado o processo de manutenção do código de teste, me conte como é este processo na empresa?

E. Na empresa que eu trabalhei era uma regra que depois do sprint que elas precisavam dos testes, a lista dos testes de unidade era executado. Caso algum teste de unidade quebrasse era verificado ao autor do teste, ou a quem fosse ciente de manutenção que corrigisse. Ou que fizesse a devida manutenção.

P. O que você conhece sobre test smell?

E. Eu conheço sobre bad smell. é em relação a isso?

P. Tem a ver. Bad smell é o conceito geral e recentemente foram separados os conceitos, code smell para o código do software e test smell para o código de teste, mas o princípio é o mesmo, bad smell. O que você conhece nesse caso sobre bad smell?

E. Bom, o teste de unidade não é somente para testar o software e garantir que ele está funcionado. Tudo começa quando você quer garantir qualidade para o seu software, garantir qualidade para arquitetura, desde o início tudo deve ser testado. Como você não tem este de unidade você corre o risco de adquirir bad smells que é um termo que se dá para código que não tem um uma qualidade muito boa, tem alta dependência ou seja, muito acoplamento, muita dependência com o banco de dados, outras classes, entre outras coisas. Então a partir do momento que você não tem teste de unidade você corre o risco de ter baixa coesão. Isso seria um bad smell.

P. Como você conheceu este assunto?

E. Foi eletronicamente, comecei a me aprofundar por conta própria através de cursos independentes ou através de artigos ou de pessoas que se dispõem a fazer vídeos e que são referências no assunto para a comunidade de desenvolvimento. Foi dessa forma que eu aprendi e praticando nos determinados projetos ou por conta própria.

P. Você possui alguma dúvida sobre esse tema?

E. Não sei se é necessariamente uma dúvida, mas qual seria a cobertura de empresas ou de projetos que se dispõem a usar TDD hoje em dia, ou seja já começa a arquitetura do software, mas antes de desenvolver já começa testando.

P. Nesse caso não é relacionado a teste smells está mais relacionado a práticas de projeto e no momento eu não teria essa informação para te passar. Isso é algo que durante as entrevistas, o que eu posso te dizer, é que eu tenho tido relatos diferentes. Então tem empresas que trabalham com TDD e tem empresas que não trabalham. Mas não é uma maioria que trabalha ou não trabalha. E isso diverge até de projeto. Mas sobre bad smell, alguma dúvida? Porque as perguntas a seguir são relacionadas a esse tema. Então se você tem alguma dúvida, alguma pergunta específica sobre bad smell, te passo a informação pois as perguntas a seguir vão estar relacionadas a esse tópico.

E. ok, podemos seguir então

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

E. Utilizando o TDD, eles devem ser tratados da seguinte forma. Quando criamos o teste de unidade e nós queremos desenvolver de fato, e nós refatoramos, nós evitamos que bad smells aconteçam através de test smells. No meu projeto o cuidado está em utilizar ferramentas como o mock para poder cobrir as dependências e não sendo ideal como o TDD, nesse caso ou você refatora ou você cria antes.

P. Como test smells são tratados na manutenção do código de teste?

E. Como boa prática, todo teste deve ser revisado por alguém que tenha mais experiência, que seja capaz de identificar quaisquer más práticas do código, sejam elas de dependências, sejam de regras de negócio, seja elas de qualidade de código. Suponhamos que temos um cenário precisando alterar determinada regra, só que foi feito com algum código muito grande, com várias dependências, tudo mais. Digamos que eu submeta um código para o revisor e ele retorne com determinadas dicas para que eu possa resolver, ou seja eu refatoro o que eu fiz, rodo novamente teste de unidade para verificar se as revisões que o revisor propôs não quebraram meu teste e se resolveram da solicitação do revisor.

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

E. primeiro ponto é garantir que alguém com mais experiência possa revisar isso e outro ponto é usar uma ferramenta de mock para evitar o máximo possível dependência de código

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

E. não sei se é uma sugestão mas queria apresentar o que eu penso para que serve o teste de unidade. Para mim ele é mais que o código somente para testar, para mim é uma garantia de qualidade em termos de isenção de dependência, em termos de desenvolvimento, em termos de acoplamento de coesão e de termos gerais de arquitetura. A partir do momento que você tem teste de unidade ou mesmo o TDD, isso te ajuda e muito a melhorar o código e arquitetura. Eu acho que é isso.

P. Ok. Muito obrigado por ter aceito participar da entrevista. Ela vai ajudar bastante dentro do meu mestrado para análise das informações. A ideia é que os resultados não fiquem estritamente na academia, de alguma forma a gente consiga devolver para indústria as informações coletadas então, eu te agradeço pela participação. Eu vou parar a gravação. Caso você tenha algum contato que possa me indicar para que eu possa realizar a entrevista, eu vou agradecer bastante.