Proposta de Trabalho

Motivações e Justificativas

Os testes automáticos são uma importante etapa no ciclo de desenvolvimento de um software. Através deles é possível testar partes de um sistema em um tempo muito menor que testes manuais, o que torna possível testes mais frequentes e, consequentemente, diminui o intervalo médio entre a introdução de um erro no código e a sua correção. Além disso, a prática de testar o código leva a uma melhor arquitetura de sistemas, já que é mais fácil de testar um código bem feito, coeso e desacoplado.

Os testes automáticos podem ser divididos em três categorias: testes de unidade, testes de integração e testes de aceitação. Os testes de unidade testam os métodos e rotinas do sistemas de forma isolada, garantindo que estes executem suas tarefas conforme a especificação; Os testes de integração, por sua vez, testam se a comunicação entre os módulos do sistema foi implementada corretamente; Já os testes de aceitação simulam casos reais de uso do sistema e verificam se o sistema cumpre os requisitos independentemente de sua implementação.

Para se realizar um teste é necessário preparar o contexto que se deseja simular. Os testes de unidade têm contextos simples, pois apenas testam uma pequena parte do sistema. Já os testes de aceitação e, em menor escala, os testes de integração, tem contextos mais complexos, pois simulam uma interação mais complexa, seja a comunicação entre dois módulos, seja um caso de uso completo. Quando testamos um caso de uso, normalmente o estado inicial do banco de dados faz parte do contexto e, em sistemas grandes, ou com dados complexos, preparar o banco de dados antes de cada teste é um desafio por si só.

Objeto/problema

Lidar com a criação do contexto de um teste que depende do banco de dados traz diversos desafios. Uma abordagem possível é manter uma base dados com exemplos de dados e executar os testes apontando para a mesma. Mas isso pode gerar uma não desejada volatilidade nos testes, já que muitas vezes os testes alteram o estado do banco de dados, ou seja, testes podem falhar graças ao resultado de testes anteriores e não por um erro no sistema. Uma solução para esse problema é recriar esta base de dados antes de cada teste que a acesse, mas o tempo de execução de cada teste aumentaria significativamente, que para sistemas grandes seria muito custoso. Assim, uma solução razoável, é limpar essa base auxiliar antes de cada teste e apenas preenchê-la com dados relevantes ao contexto do próprio teste.

Mesmo assim, ainda há algumas dificuldades: linguagens de banco de dados como SQL não são muito fáceis de manter, muitos comandos de SQL são específicos para determinadas plataformas, Não é simples reaproveitar scripts de banco de dados em diferentes testes. Outro ponto negativo desta solução é o fato de ser mais fácil para o desenvolvedor descrever também o contexto dos testes utilizando uma linguagem de programação. Foi para lidar com esses desafios que a ferramenta dbunit foi criado e espero conseguir avaliar como a comunidade está utilizando esta ferramenta neste trabalho.

O dbunit é uma extensão do Junit feita para controlar o estado do banco de dados antes de cada teste. Ele é a principal ferramenta para integração de testes automáticos com o banco de dados em Java, além de ter versões para outras linguagens como php e .NET. Basicamente o dbunit permite que o desenvolvedor descreva o estado do banco de dados antes de cada cenário através de arquivos escritos com uma linguagem de marcação (no caso XML). O dbunit também permite validar o estado resultante do banco de dados durante um teste.

Objetivos do estudo

Neste trabalho pretendo fazer uma analisar o que a comunidade tem a dizer sobre o dbunit. Para isso, pretendo coletar e analisar postagens feitas no site Stack Overflow relacionadas com o dbunit. O Stack Overflow é uma rede social de perguntas e respostas, onde desenvolvedores podem tirar suas dúvidas sobre diversos temas relacionados a programação de computadores. Suas perguntas são respondidas por outros usuários e a resposta que resolver o problema é identificada pelo autor da pergunta como a solução utilizada. Além disso, usuários com uma determinada reputação podem ranquear tanto as perguntas quanto as respostas postadas no site. Eles fazem isso adicionando ou removendo um ponto, quanto mais pontos, melhor.

O objetivo desta análise é apresentar uma amostragem do que está sendo discutido em relação ao dbunit para, assim, descobrir tendências dessa área, encontrar gargalos na tecnologia, sugerir melhorias na documentação, entender melhor os casos em que o dbunit é, ou não, utilizado etc. Além disso, faz parte do escopo deste projeto, comparar os resultados com trabalhos similares a este. Com isso espero entender melhor os resultados a partir do contraste com dados de outras comunidades, ou seja, descobrir se o perfil de dúvida de um usuário do dbunit é similar ao perfil de dúvida de um usuário de outras tecnologias da área de desenvolvimento de sistemas.

Principais atividades previstas

A principal atividade do projeto é a extração dos dados do Stack Overflow e o seu mapeamento. Uma atividade importante neste trabalho será entender o tipo de pergunta que é feita pelos os usuários do site. Para isso, pretendo classificar as postagens em três modalidades: Perguntas sobre “o quê” fazer, perguntas sobre “como” fazer e perguntas sobre “por quê” fazer. As perguntas do tipo “o quê” são postagens das quais o autor tem um problema

para resolver e quer saber qual é a melhor forma de utilizar o dbunit para resolvê-lo. As perguntas do tipo “como” são postagens nas quais o autor tem uma solução incorreta ou imperfeita para um problema e quer saber o que está fazendo de errado. Já as perguntas do tipo “por quê” são postagens das quais o autor conhece uma técnica comum de utilizar o dbunit, mas quer saber o por quê daquela técnica ser implementada de uma forma específica.

Para fazer essa classificação dos mais de 2000 posts disponíveis no site, será necessário utilizar técnicas de mineração de dados, como stemização e aprendizado de máquina. Stemização consiste em reduzir as palavras de um texto a sua raiz, ou seja, reduzir flexões diversas de uma raiz semântica para uma mesma “string”. Por exemplo: “pesquisa”, “pesquisar” e “pesquisado” se referem ao conceito de pesquisa, por tanto, o processo de stemização transformará cada ocorrência dessas palavras em um único símbolo (“pesquis” por exemplo). Esse processo ajuda a representar melhor a ocorrência de um mesmo significado entre as postagens. Após o processo de stemização, utilizarei algoritmos de aprendizado supervisionado para classificar as postagens nos tipos descritos anteriormente. Essas técnicas necessitam que algumas postagens sejam classificadas previamente para servir como treino para estes algoritmos, esses dados de treinamento serão classificados manualmente pelo autor deste trabalho.

Outra análise que proponho é encontrar grupos de temas nas postagens. Para isso, utilizarei a técnica Latent Dirichlet Allocation (LDA) que encontrará grupos de palavras recorrentes nas postagens possibilitando a classificação destas por temas. Além disso irei coletar informações do próprio Stack Overflow para avaliar a qualidade das respostas, mais especificamente, se houve uma resposta marcada como solução pelo autor da postagem e o número de votos que cada resposta recebeu dos usuários do site. Esses dados qualitativos poderão ser cruzados com a classificação de tipos para entendermos como a comunidade do dbunit se comporta com cada tipo ou tema de uma postagem.

E, finalmente, proponho comparar os resultados obtidos com trabalhos similares que utilizam os dados do Stack Overflow para avaliar outras tecnologias de desenvolvimento de software. Com isso espero entender melhor os desafios que os desenvolvedores enfrentam ao utilizar esta ferramenta em comparação com o resto do ciclo de desenvolvimento. Por exemplo, será que há a mesma preocupação conceitual ao se escrever testes com dbunit do que há em outras partes do ciclo? Será que o usuário de dbunit está tendo suas dúvidas respondidas com a mesma frequência que outras tecnologias?

Cronograma aproximado

- A fazer