

Instituto de Matemática e Estatística - USP

Proposta de Trabalho

Percepção coletiva da comunidade de desenvolvimento em
relação ao dbunit

Vinícius Nascimento Silva,

São Paulo, 2017

Motivações

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, conseqüentemente, 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.

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.

Lidar com a criação de um contexto 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. Sendo 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 e 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 o contexto dos testes utilizando uma linguagem de programação. Foi para lidar com esses desafios que a ferramenta dbunit¹ foi criado e espero conseguir avaliar como a comunidade está utilizando esta ferramenta neste trabalho.

¹ <http://dbunit.sourceforge.net/>

O Problema

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.

Apesar de esta ferramenta ser conhecida e utilizada por muitos desenvolvedores, pouco se sabe sobre o seu uso. Além disso, não há muitas alternativas ao dbunit no momento em que este trabalho está sendo escrito. O candidato que mais se destaca é o DbSetup², mas ainda é pouco utilizado pela comunidade, o que torna inviável fazer um levantamento quantitativo de dados sobre o seu uso.

Objetivos do estudo

Neste trabalho pretendo 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 este software. 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.

Outros trabalhos já foram feitos utilizando os dados do Stack Overflow com resultados interessantes. Rosen e Shihab³ verificaram que desenvolvedores de aplicações móveis estão perguntando sobre distribuição dos aplicativos, APIs de programação móvel, gerenciamento de

² <http://dbsetup.ninja-squad.com/index.html>

³ https://users.encs.concordia.ca/~eshihab/pubs/Rosen_EMSE2015.pdf

dados, contexto e sensores, ferramentas de programação móvel, e desenvolvimento de interface de usuário. Delfim⁴ mostrou que há uma forte associação entre a quantidade de informação sobre uma API no Stack Overflow e a sua utilização em sistemas na vida real.

Questões de Pesquisa

A principal atividade do projeto é a extração dos dados do Stack Overflow e o seu mapeamento. E, com esses dados pretendo responder às seguintes questões de pesquisa: **Q1**: Qual é o tipo de pergunta mais feita pelos usuários do dbunit no Stack Overflow. **Q2**: Quais são os temas mais comuns nas postagens do Stack Overflow relacionadas ao dbunit como. **Q3**: Qual é a qualidade das respostas oferecidas pelos usuários do site. **Q4**: As diferenças entre as dificuldades dos usuários do dbunit e de outras tecnologias de desenvolvimento de software.

Para lidar com a **Q1** 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.

Para lidar com a **Q2**, utilizarei a técnica Latent Dirichlet Allocation (LDA)⁵ que encontrará grupos de palavras recorrentes nas postagens possibilitando a classificação destas por temas. Com isso, poderemos extrair os tópicos mais comuns nas postagens e verificar para cada postagem quanto ela se aproxima de cada tema.

⁴ <https://link.springer.com/article/10.1186/s13173-016-0049-0>

⁵ https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

Para lidar com a **Q3**, 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, para lidar com a **Q4**, 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?

Proposta de Cronograma

- Extração e stemização dos dados do Stack overflow para uma base de dados. - **Até 22 de Maio**
- Classificação manual de uma amostragem de 200 postagens entre perguntas dos tipos *por quê*, *o quê* e *como* - **Até 5 de junho**
- Exploração de técnicas de aprendizado de máquina para classificar o restante das postagens - **Até 3 de junho**
- Extração de temas das postagens através do LDA - **Até 31 de Julho**
- Análise dos resultados e elaboração de respostas para as questões da pesquisa - **Até 28 de agosto**
- Escrever a Monografia - **Até 1 de Novembro**
- Preparar o Pôster - **Até 1 de Novembro**