Relatório - Exercício 4: Implementação dos Requisitos de Teste

Rodrigo Duarte Louro e António Martins Miranda

05 de maio de 2014

1 Utilização do Recoder

A ferramenta recoder foi utilizada em praticamente todos os passos do desenvolvimento desse projeto. A Árvore de análise sintática gerada por essa ferramenta nos deu toda a base para o desenvolvimento dos diversos critérios para a implementação dos requisitos de teste.

A ferramenta possui documentação disponível em recoder.sourceforge.net /doc/api/ tal documentação, que se trata de uma espécie de javaDoc, foi um dos empecilhos para o melhor andamento do projeto, dado que perdemos muito tempo tentando adivinhar qual seria o nome da classe que apresentava a funcionalidade que desejavamos.

O tutorial disponibilizado pelo Alexandre foi de vital importância, pois depois de não conseguirmos desenvolver o projeto com o codeCover já tinhamos tentado fazer o tutorial sugerido no site do recoder e não tinhamos conseguido termina-lo.

2 Todas Decisões

O critério de todas as decisões faz com que todas as decisões do programa sejam testadas em seus dois possíveis valores (verdadeiro ou falso). Assim sendo, utilizamos a árvore de análise sintática gerada pelo recoder para pegar todos os métodos de todas as classes presentes na pasta passada como parâmetro.

Para cada método criamos uma nova árvore sintática local que analiza apenas aquele método. A árvore sintática do método é totalmente percorrida e se encontramos um decisão adicionamos ela em uma lista. Isso é feito através das classes do proprio recoder (If.class, while.class, for.class, do.class).

Depois de termos todas as decisões do método, o programa duplica essas decisões e as diferencia apenas pelo atributo valor presente na classe Decisão, uma fica com valor verdadeiro e outra com valor falso.

3 Todas Condições

O critério de todas as condições faz exatamente o que o de todas as decisões faz com mais alguns passos. Dentre esses passos a mais um possui uma grande diferença, o objeto instanciado que guarda as informações de uma decisão não é da classe Decisão.java e sim da classe DecisaoAux.java. A diferença principal entre as duas é que a DecisãoAux possui uma lista de condições, isso foi feito apenas para que o arquivo XML gerado fosse mais claro. Nessa abordagem cada decisão possui uma lista de condições. Para cada condição devemos setar o valor como falso e na condição semelhante o valor verdadeiro. Assim cada decisão(DecisaoAux) possui 2n condições no arquivo xml (supondo uma decisão com n condições) dessas, n possuem valor falso e as outras n, com as mesmas informações das primeiras n, possuem valor verdadeiro.

4 MCDC

O critério MCDC que implementamos executa os mesmos passos do critério sobre todas as decisões. Após possuirmos todas as decisões devidamente preenchidas em uma estrutura de árvore onde todos os nós são decisões com exceção das folhas que são condições, criamos uma tabela verdade para aquela decisão e filtramos os testes independentes. Apenas os testes independentes são colocados no arquivo XML. Infelizmente não nos sobrou tempo para o desenvolvimento do método que eliminaria as máscaras.

5 XML

Os arquivos XML foram criados a partir da biblioteca xstream.jar. Essa biblioteca converte um objeto qualquer em um arquivoXML utilizando como tags os nomes de seus atributos. Para isso criamos 3 classes, cada uma represetando um dos 3 critérios a serem analisados. As infomações foram obtidas a partir de toda a modelagem executada pelo nosso programa, que tenta simular um package que contém classes java

6 Como Usar?

Para utilizar o protótipo deve-se passar por parâmetro o caminho absoluto da pasta que contém as classes java a serem analisadas. A saída é composta por 3 arquivos XML, cada um correspondente a dos critérios pedidos no enunciado.

7 Testes

Em anexo no zip que será entregue no paca estão as três classes que utilizei para testar o programa e os três arquivos XML gedados.

8 Conclusões

- Pontos Positivos:
  - Estruturas em Árvore
  - Aprender como o recoder funciona
  - Projeto em java e utilização do github
- Pontos Negativos:
  - CodeCover
  - Fases Anteriores do projeto.
  - Modelagem do problema
  - Não termos feito na semana do break
  - Sumiço de um dos membros do grupo.