RELATÓRIO TRABALHO 2 - MÉTODOS FORMAIS

Repositório do Trabalho: https://github.com/luisEduardoBet/TrabalhoMFO2

1. Descrição (Copiado do Enunciado):

Você recebeu um software em C++ que implementa um banco, e sua missão é encontrar e consertar bugs deste código. Você já escreveu uma especificação em *Quint* com as funcionalidades desse software, verificando propriedades relevantes e ajustando a especificação até que todas elas fossem satisfeitas (assim como fizemos no trabalho 1). Com isso, você consegue afirmar que o modelo está correto. Contudo, a sua missão é sobre o código, não sobre o modelo. Então, agora você precisa utilizar o modelo para aumentar a confiança no software em si. Não é possível, com testes, garantir que o modelo e o software correspondem 100%. Mas, quanto mais testes forem rodados, menor a chance de termos diferenças de comportamento. Conecte o modelo existente ao código existente com testes baseados em modelos. Para cada teste que falhar, ou seja, para cada execução onde o comportamento do modelo for diferente do comportamento do código, ajuste o código para que o teste passe a suceder.

2. Estruturação dos Testes:

Inicialmente, para a estruturação dos testes, todas as ações que o sistema poderia executar em cada estado foram conectadas ao código C++. Simultaneamente, utilizou-se a variável ***error*** para armazenar os possíveis erros retornados pelo código, com o objetivo de permitir comparações posteriores com os erros produzidos pelo modelo em Quint.

Para comparar os resultados produzidos pelo código em C++ com os gerados pelo modelo em Quint, foi criada a função ***printBankMap***, cuja função é exibir, em cada estado, os valores das variáveis essenciais do banco (presentes em um mapa). Durante os testes, comparou-se o erro previsto pelo modelo (erro esperado) com o erro devolvido pelo código C++ e, sempre que surgia alguma divergência, analisavam-se os valores impressos das variáveis de estado do modelo e do código para localizar a causa do problema e corrigi-la no código.

Essas implementações descritas podem ser visualizadas nos commits:

- 910ae48
- 85c5cdf - Ajuste feito para melhorar a visualização do mapa “printado”.

3. Correções:

Conforme descrito na Seção 2, o processo de correção consistia na comparação entre o erro emitido pelo código e o erro esperado. Caso houvesse divergência (ex.

um dos lados emitir erro e o outro não), analisava-se a saída impressa contendo os mapas de estado e a ação executada, a fim de identificar a causa do problema e realizar os ajustes necessários no código.

Segue as correções realizadas ao longo dos testes:

- ebd8768 →Montante deve ser maior que zero ao comprar um investimento;
- eb785f3 → Montante sacado deve ser menor ou igual ao saldo da conta;
- 89ce014 → Montante deve ser maior que zero ao realizar uma transferência;
- fec7501 → Montante enviado em uma transferência deve ser menor ou igual ao saldo da conta de quem está enviando;
- 3b8fe0a → Um investimento não pode ser vendido sem ter sido comprado;
- d576baa → Montante deve ser menor ou igual ao saldo da conta ao comprar um investimento;
- 353d058 → Montante deve ser maior que zero ao realizar um depósito;
- 0cbb45f → Montante sacado deve ser maior que zero;
- b7d6e4f → Um investimento só pode ser vendido pelo seu dono.
- feafb28 → Um investimento vendido deve ser removido da lista de investimentos ativos;

OBS: As mensagens dos commits descrevem o problema que buscou-se resolver. Isso pode ser visualizado no repositório presente no Github.