Qualidade e Confiabilidade de Software
Assignment #2

Victor Oliveira[1], André Macedo[2]
Departamento de Engenharia Informática
Faculdade de Ciências e tecnologia
Universidade de Coimbra
{victorho[1], afmacedo[2]}@student.dei.uc.pt

Março 2016

Conteúdo

1 Introdução

O propósito deste trabalho incide em desenvolver uma aplicação baseada em N-Version programming que permita calcular a quantidade de insulina que um paciente com diabetes tipo-2 deve tomar. Pacientes com diabetes tipo-2 necessitam de injeções de insulina periódicas de forma a conseguirem processar açucar no sangue. É de suma importância que a quantidade de insulina tomada seja suficiente para metabolizar hidratos de carbono ingeridos, e ao mesmo tempo garantir que a dose não seja em demasia tal que provoque hypoglicémia, o que pode levar a um estado de coma ou até mesmo causar dano cerebral.

2 Diabetes tipo-2

O objetivo fundamental no controlo do diabetes é conseguir manter o nível de glucose no sangue entre os 80 mg/dl e 120 mg/dl. Este valor varia entre pacientes, sendo sempre um médico a prescrever a dose recomendada. O paciente mede então o nível de glucose e calcula a dose necessária de forma a manter a mesma em níveis satisfatórios. A quantidade de insulina injetada é medida em doses (ou unidades), e que somam a um total diário. Existem duas componentes na toma diária. Uma é a substituição basal de insulina, outra a substituição bolus de insulina. A primeira é necessária para satisfazer as necessidades de insulina entre refeições. A segunda é necessária para conseguir processar os hidratos de carbono ingeridos nas refeições.

3 Objetivos

O objectivo principal para este trabalho é o desenvolvimento de uma aplicação que permita calcular a dose de insulina necessária de tomar. Para tal, foi criado uma aplicação web que contem que implementa um votador, assim como uma interface web para o utilizador. Foi também desenvolvido um webservice capaz de efetuar os calculos necessários para a obtenção da dose correta de insulina. O calculo de unidades a tomar pode ser feito através de uma das seguintes três formas distintas:

- Calcular as unidades de insulina necessárias após uma refeição, com sensibilidade padrão
- Calcular as unidades de insulina necessárias após uma refeição, com sensibilidade pessoal
- Calcular as unidades de insulina necessárias entre refeições

4 Requisitos

4.1 Requisitos Funcionais

Os requisitos funcionais foram todos implementados consoante definidos. São utilizados os ids de referência originais conforme foram definidos no enunciado, e são os seguintes:

- **ID:FR1** Apresentar no ecrã os vários tipos de calculo possíveis, após seleção, os campos referentes a esse calculo devem aparecer.
- **ID:FR2** calculo de dose de insulina a tomar após refeição, utilizando a sensibilidade standard.
- **ID:FR3** calculo de dose de insulina a tomar após refeição utilizando uma sensibilidade personalizada.
- **ID:FR4** calculo de dose de insulina utilizando o peso do utilizador.
- **ID:FR5** Mostrar dados técnicos acerca do calculo de insulina efetuado.

Uma listagem por extenso dos requisitos funcionais e todos os seus atributos podem ser vistos no enunciado do projecto em anexo.

4.2 Requisitos Não Funcionais

De seguida são listados todos os requisitos não funcionais pedidos. Mais uma vez todos foram implementados. Uma explicação detalhada de como foi conseguido atingir cada um é dada na secção sobre a Arquitetura.

- **ID:NFR1** Efetuar um calculo confiável de dose de insulina através da utilização de 3 ou mais fontes de calculo, e um votador para validar os resultados.
- **ID:NFR2** Performance quanto à rapidez de calculo da dose de insulina. Esta deve ser efetuada até 4 segundos após o botão para efetuar o calculo ser premido.
- **ID:NFR3** Efetuar nova tentativa de calculo em caso de nenhum resultado válido ser obtido. A nova tentativa deve utilizar os mesmos dados. No entanto pode utilizar outras versões do web service se disponíveis.

4.3 Restrições de Design

Na elaboração da aplicação para o calculo de insulina, foram impostas algumas restrições de design. Estas incidiram nos seguintes aspetos:

- **ID:DC1** Utilização de N-Version programming
- **ID:DC2** Utilização da tecnologia web services

- **ID:DC3** Efetuar a verificação formal do votador da aplicação
- **ID:DC4** Utilizar um mecanismo votador para validar os resultados do calculo
- **ID:DC5** Utilização do tipo Java double para efeitos de calculo, como também efetuar um arredondamento só no final de todos os calculos intermédios estarem finalizados

Mais uma vez, todas as restriões de design impostas, foram conseguidas/implementadas.

5 Web Service

O web service foi implementado com a API Java-WS, conseguindo assim satisfazer o **DC2**. Este web service está disponível e pode ser acedido através do seguinte link: `http://webservice-sqdcourse.rhcloud.com/InsulinDoseCalculator`. O webservice tem quatro métodos implementados, três deles referentes aos tipos de cálculo possíveis de efetuar pelo utilizador, e um outro que é utilizado pelo cálculo de insulina após as refeições com sesibilidade pessoal. Cada um é proveniente dos requisitos acima descritos. Os métodos que a interface disponibiliza são então:

- int mealtimeInsulinDose(int carbohydrateAmount,
  int carbohydrateToInsulinRatio,
  int preMealBloodSugar,
  int targetBloodSugar,
  int personalSensitivity);
- int backgroundInsulinDose(int bodyWeight);
- int personalSensitivityToInsulin(int physicalActivityLevel,
  int[] physicalActivitySamples,
  int[] bloodSugarDropSamples);

6 Aplicação web

A aplicação web foi construida utilizando a framework de python Flask. Utilizamos esta framework por ser bastanve flexivel, leve e rápida de implementar. A aplicação é constituída por 2 componentes principais: a interface web e o votador.

6.1 Arquitetura

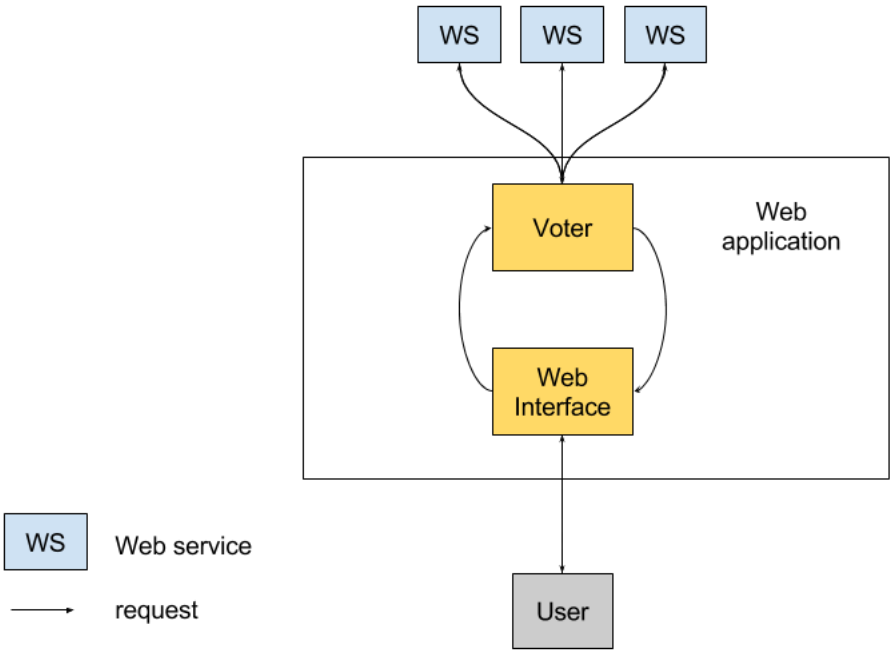

Figura 1 - Arquitetura da aplicação web

6.2 Interface web

A interface web foi desenvolvida utilizando HTML, CSS e Javascript. Para satisfazer todos os requisitos em relação às verificações de inputs, utilizou-se o Javascript, e as funções respectivas encontram-se no ficheiro js-functions.js. Foram implementadas algumas funcionalidades adicionais para aumentar ligeiramente a interatividade do utilizador com a aplicação, estas também estão contidas no mesmo ficheiro e devidamente comentadas.

6.3 Votador

Sendo os resultados produzidos por esta aplicação de natureza crítica, o componente onde mais esforços foram concentrados foi o votador. O votador contém 2 listas de webservices (uma principal e outra de backup), cada uma com 3 webservices escolhidos aleatóriamente da lista disponível. Quando é recebido um pedido de cálculo da interface web, o votador cria 3 threads (uma para cada webservice na lista principal) que vão fazer o pedido dos cálculos ao webservice respectivo. A partir deste momento o votador fica bloqueado durante no máximo 4 segundos (tempo definido nos requisitos para o timeout). Caso o timeout seja ultrapassado, o votador retorna com um array de resultados igual a -1. No caso de receber a resposta antes de acabar o timeout, o votador decide se existe maioria nos resultados obtidos, através do calculo da mediana. Caso a maioria não seja conseguida, é feito uma nova tentativa de cálculo, desta vez utilizando os webservices da lista de backup, com o tempo restante ao timeout de 4 segundos. Caso a maioria seja conseguida, o votador devolve um dicionário de python com um array que contem as respostas de cada webservice e o resultado da maioria, a lista de webservices principal, a lista de webservices de backup e também uma variavel a indical qual das listas foi utilizada.

6.4 Comunicação com o Web Service

A comunicação entre a aplicação e o web service é feita através de mensagens SOAP. É invocando no endpoint o método respetivo ao calculo a efetuar, com os devidos parametros, e a subsequente resposta é enviada.

7 Verificação Formal

A verificação formal foi feita utilizando a ferramenta Spin. O código do votador foi modelado na linguagem Promela o mais próximo possível ao original. Como o nosso código fonte está em Python, houve pequenas alterações de forma a poder adaptar à linguagem Promela. Nomeadamente em relação a tudo o que se relaciona com threads o Python é bastante diferente ao C, o que provocou as ditas alterações. Outra adaptação efetuada ao modelo do votador foi a simulação de valores recebidos dos web services. Isto foi feito de forma a se poder testar extensivamente todas as combinações de toda a gama de valores possíveis.
A função principal é a "n voter"que definimos como activa de forma a ser criado automáticamente o seu respectivo processo. É efetuada a simulação de obtenção de calculos aos web services e, após isto, iniciada a função responsável por decidir se os resultados são válidos ou não. Verificamos que é bastante dificil obter um resultado positivo (em que haja maioria) pois ao se testar todas as combinações, a explosão de espaços possíveis aumenta consideravelmente. Isto potênciado pela função timeout do spin, que por sua vez faz com que os processos sejam terminados de forma não determinística.