

N-version Programming Web Services

QUALIDADE E CONFIABILIDADE DE SOFTWARE

Trabalho Prático 1

Bruno Madureira, Fábio Moura e Nuno Miranda.

Departamento de Engenharia Informática
Universidade de Coimbra

Índice

I. INTRODUÇÃO	2
II. IMPORTÂNCIA DA QUALIDADE E CONFIABILIDADE NESTE PROGRAMA	2
III. TÉCNICAS E TECNOLOGIAS USADAS	2
VI. ARQUITETURA	3
A. Casos de Uso	3
B. Vista componente e conetor	5
C. Validação.....	7
V. CONCLUSÃO.....	8

I. INTRODUÇÃO

Este trabalho apresenta e documenta uma implementação de um calculador de doses de insulina para doentes com diabetes tipo 2 altamente confiável. Este programa foi implementado usando *web services* e *N- Version programming*.

Neste relatório apresentaremos a arquitetura deste programa, detalhando os módulos principais e iremos também realizar a verificação formal de um dos módulos principais: o votador.

II. IMPORTÂNCIA DA QUALIDADE E CONFIABILIDADE NESTE PROGRAMA

A Diabetes tipo 2 é uma doença que em 2010 afetava cerca de 285 milhões de pessoas. Esta doença é um distúrbio metabólico caracterizado pelo elevado nível de glicose no sangue, elevada resistência à insulina e insuficiência relativa de insulina.

Em casos controlados não é necessária a administração de insulina, no entanto em casos mais graves é necessário. O nosso programa recebe vários parâmetros, devolvendo quantas unidades de insulina o utilizador deve administrar.

A administração de quantidades erradas de insulina pode ter graves consequências para o doente. No caso da administração de insulina a menos o doente pode entrar em hiperglicemia e no caso de administração de insulina a mais o doente pode entrar em hipoglicemia. Ambas as situações são perigosas para o doente, principalmente a hipoglicemia. Para evitar estas situações é imperativo que o programa que seja responsável pelo cálculo da quantidade de insulina a ser administrada seja altamente confiável.

III. TÉCNICAS E TECNOLOGIAS USADAS

Era pedido que para construir este sistema usássemos *Web services* e *N-version programming*. *Web services* são sistemas desenhados para suportar interação ininterrupta entre sistemas remotos.

N-version programming é uma técnica que vários programas com funções equivalentes são gerados de uma especificação inicial. Esta técnica visa minimizar os erros que o sistema pode ter através da replicação e variabilidade. Os resultados de cada programa são comparados através de um sistema de votação.

IV. ARQUITETURA

A. CASOS DE USO

Eis os casos de uso considerados para este sistema (figura 1):

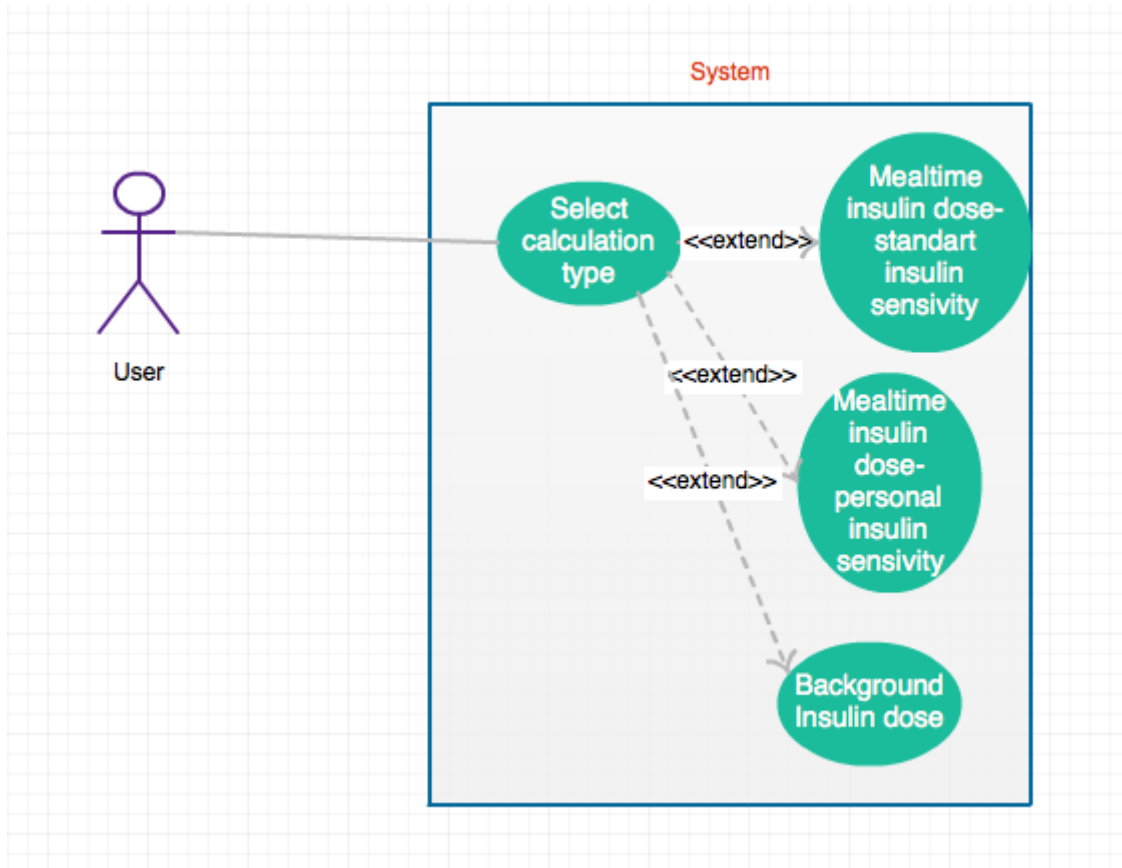


Figura 1 - Casos de uso

Select calculation type

Atores primários	User
Atores secundários	
Descrição	Esta caso de uso permite selecionar o tipo de cálculo
Trigger	User pretende saber se necessita de tomar insulina
Pré-condição	Momento em que seja possível necessitar de insulina
Pós-condição	

Mealtime insulin dose standart insulin sensitivity

Atores primários	User
Atores secundários	
Descrição	Cálculo de Mealtime insulin dose-standard insulin sensitivity
Trigger	Desejo de medir Mealtime insulin dose-standard insulin sensitivity
Pré-condição	Select calculation type
Pós-condição	

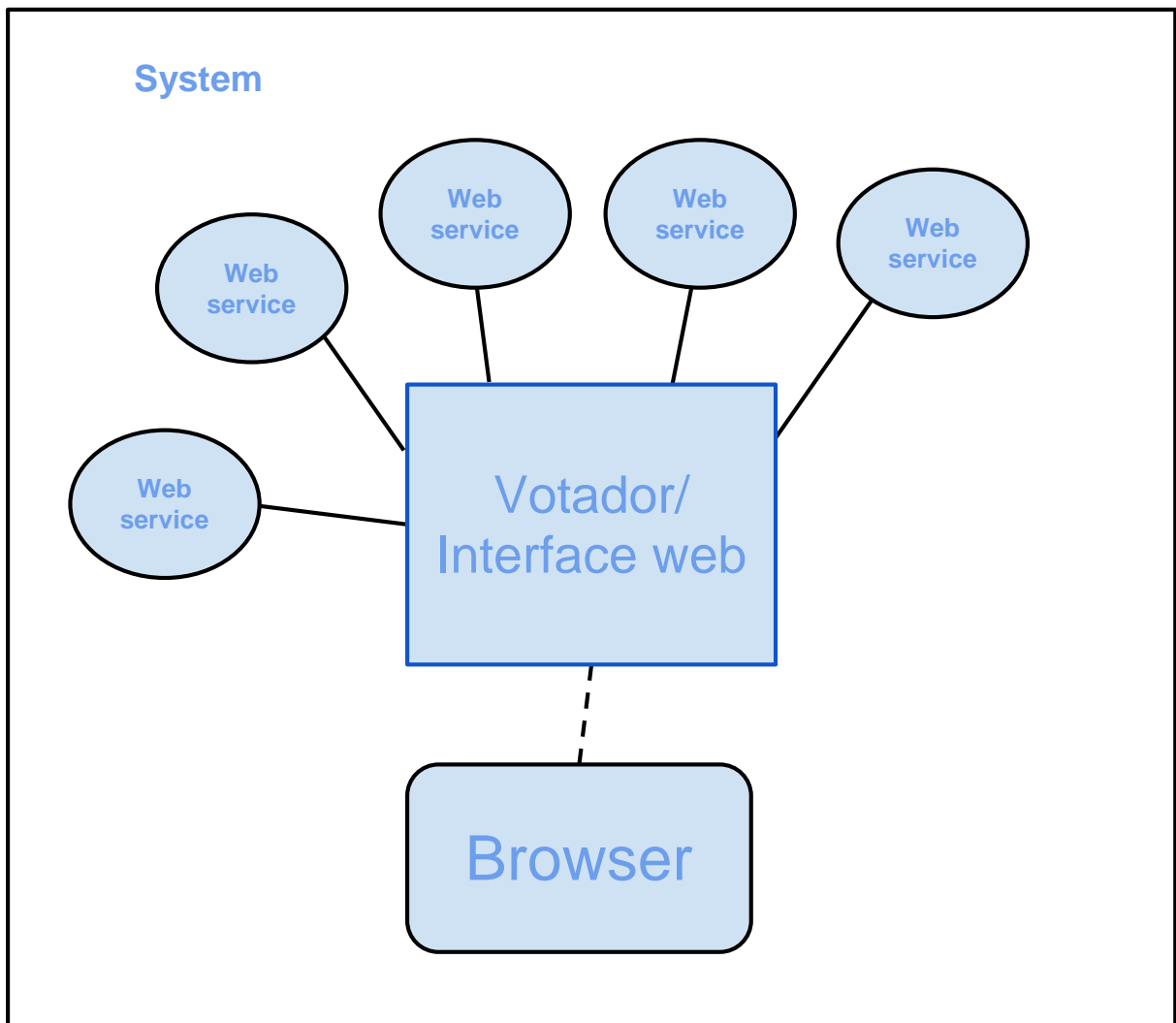
Mealtime insulin dose personal insulin sensitivity

Atores primários	User
Atores secundários	
Descrição	Cálculo de Mealtime insulin dose personal insulin sensitivity
Trigger	Desejo de medir Mealtime insulin dose personal insulin sensitivity
Pré-condição	Select calculation type
Pós-condição	

Background insulin dose

Atores primários	User
Atores secundários	
Descrição	Cálculo de Background insulin dose
Trigger	Desejo de medir Background insulin dose
Pré-condição	Select calculation type
Pós-condição	

B. VISTA COMPONENTE E CONECTOR



Descrição geral

O nosso sistema é constituído por 3 componentes principais: Diversos *Web services* desenvolvidos em java (numero variável mas sempre impar e maior que 3), um componente desenvolvido em java que agrega o votador e a interface web e um Browser que é responsável por aceder a interface web.

Descrição dos módulos principais

O *Web service* implementa 3 métodos:

- mealtimeInsulinDose.
- backgroundInsulinDose.
- personalSensitivityToInsulin.

Não vão ser explicados os *inputs* e *outputs* destes métodos já que eles estão definidos no *Javadoc* fornecido pelo professor.

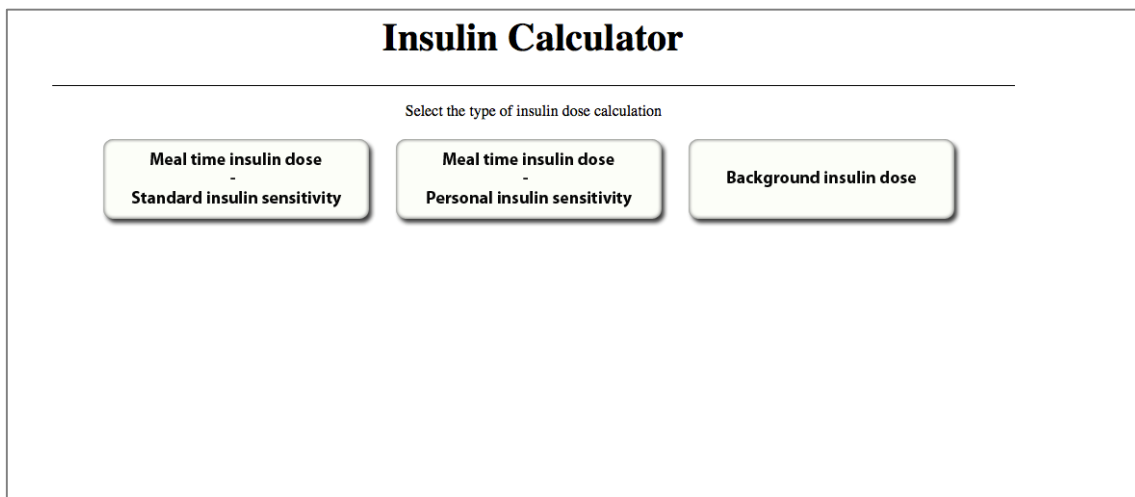
Votator

O votador foi escrito em Java e é constituído por $n+1$ threads, sendo que uma é a principal e as outras estão encarregadas de ligarem-se a um *web service*, chamarem um método e colocarem o seu resultado num *arraylist* de votos.

A thread principal vai aceder a esse *arraylist* e através de uma função de validação verificar se há algum resultado em maioria. Para fazer isto percorremos metade+1 de todos os votos do *arraylist* e para cada um desses votos contávamos as ocorrências nesse *arraylist*. Quando um tivesse mais que (número de *webservices*/2) votos seria eleita a resposta correta e retornada. Caso nenhum elemento do *arraylist* satisfizesse as condições seria retornado o código de erro -1, e a interface web diria que não existe resposta satisfatória.

Interface Web

Desenvolvemos uma interface web simples usando java. Para isto foram criados um *servlet* e vários *jsp*s. Para que o utilizador não inserisse valores indivíduos criamos vários scripts em *Jquery* para proteger o sistema destas situações.



The image shows a web interface titled "Insulin Calculator". Below the title is a horizontal line, and then the text "Select the type of insulin dose calculation". There are three buttons arranged horizontally: "Meal time insulin dose - Standard insulin sensitivity", "Meal time insulin dose - Personal insulin sensitivity", and "Background insulin dose".

Figura 2 - interface web

C. VALIDAÇÃO

Para validar formalmente o programa construímos um modelo na linguagem *promela*. Para analisar o modelo usamos o *spin*. Os programas concorrentes podem ter problemas como acessos indevidos ou não sincronizados a variáveis. Para simular construímos este modelo:

```
#define NUMPROCS 3

byte
count = 0;
byte array[NUMPROCS];

proctype incremter(byte id)
{
    int temp;

    atomic {
        temp = count;
    }

    count = temp + 1;
    array[id] = 1;
}

init {
    int i = 0;
    int sum = 0;

    atomic {
        i = 0;
        do
            :: i < NUMPROCS ->
                array[i] = 0;
                run incremter(i);
                i++;
            :: i >= NUMPROCS -> break
        od;
    }
    atomic {
        i = 0;
        sum = 0;
        do
            :: i < NUMPROCS ->
                sum = sum + array[i];
                i++;
            :: i >= NUMPROCS -> break
        od;
        assert(sum < NUMPROCS || count == NUMPROCS)
    }
}
```

Figure 3 - Validador

Neste programa temos três *threads* que vão partilhar um *array* e uma variável. Cada *thread* vai incrementar a variável *count* e o seu elemento correspondente no *array*. Depois de cada *thread* acabar será feita a verificação a testar se não foi encontrado um estado estranho.

O resultado da avaliação do modelo foi o seguinte (figura 4):


```

State-vector 56 byte, depth reached 29, errors: 0
  372 states, stored
  232 states, matched
  604 transitions (= stored+matched)
  431 atomic steps
hash conflicts:          0 (resolved)

Stats on memory usage (in Megabytes):
  0.030      equivalent memory usage for states (stored*(State-vector + overhead))
  0.290      actual memory usage for states
 128.000     memory used for hash table (-w24)
  0.534      memory used for DFS stack (-m10000)
 128.730     total actual memory usage

unreached in proctype incrementer
  (0 of 5 states)
unreached in init
  (0 of 24 states)

pan: elapsed time 0 seconds
MBP-de-Bruno-3:PML brunomadureira$ █

```

Figure 4 - Resultados da validação

Como é possível ver não foram encontrados erros.

V. CONCLUSÃO

Depois de termos implementado, validado e testado o programa pedido ficamos familiarizados com o desenvolvimento de programas usando o conceito de *N-version programming*. Pareceu-nos uma arquitetura simples conceptualmente e mostra-se tolerante a falhas tal como suposto, no entanto não nos foi possível deixar de reparar que num contexto do mundo real o desenvolvimento paralelo de várias replicas pode ser extremamente caro e moroso.

REFERÊNCIAS

- [1] http://en.wikipedia.org/wiki/N-version_programming
- [2] Material das aulas