
Assignment 1: N-Version programming web services

Qualidade e Confiabilidade de Software
Universidade de Coimbra



UNIVERSIDADE DE COIMBRA

Autores:

José Pedro Marques, nº 2011144548

Paulo Pereira, nº 2011164879

Pedro Ferreira, nº 2011154743

17 de Maio de 2015

1 Sumário executivo

A programação *N-Version* consiste no desenvolvimento de N versões independentes de *Software*, todas elas baseadas nas mesmas especificações. Esta técnica tem como objetivo o desenvolvimento de software altamente fiável uma vez que o resultado final é uma votação dos resultados produzidos pelas N versões.

Neste trabalho, pretende-se aplicar a técnica referida a uma situação real que passa pelo cálculo de doses de insulina para diabéticos do tipo 2. Para isso, desenvolveu-se um *Web Service* que disponibiliza os métodos para efetuar o referido cálculo e um votador que, com base nos resultados devolvidos não só pelo *Web Service* que desenvolvemos mas também por mais dois de um conjunto de *Web Services* desenvolvidos por outros grupos, produz um resultado final. Estas funcionalidades estão acessíveis através de uma interface Web produzida para o efeito.

O presente documento, que serve como relatório do trabalho realizado, contém uma descrição geral dos módulos desenvolvidos e uma análise técnica do votador, que funciona sobre os dados devolvidos pelos serviços Web, e é a parte que diz respeito ao tema que dá origem a este trabalho, *N-Version Programming*.

Conteúdo

1	Sumário executivo	2
2	Introdução	4
3	Arquitetura do Software	5
3.1	Web Service	6
3.2	Votador	6
3.3	Interface Web	7
4	Análise técnica do votador	9
4.1	Resolução dos problemas	9
4.2	Verificação formal do votador	10
5	Descrição da divisão das tarefas	12
6	Conclusão	13

2 Introdução

Quando se desenvolve uma aplicação na área da saúde, assim como em áreas que envolvem a segurança da vida humana, é necessário assegurarmos-nos de que essa aplicação é fiável. N-version programming é um método que tem como finalidade criar um sistema altamente fiável e seguro, sendo usado em sistemas críticos onde erros/falhas têm de ser evitadas ao máximo. Este método consiste no uso de N versões diferentes de programas que cumpram os mesmos requisitos, aumentando a redundância e tolerância a possíveis falhas.

Neste trabalho, iremos aplicar a técnica de N-version programming a uma calculadora de insulina para pacientes de diabetes do tipo 2. Esta é uma doença bastante comum nos dias de hoje, cujo tratamento pode estar dependente da injeção de insulina na corrente sanguínea. A quantidade de insulina administrada é fulcral, pois quantidades erradas podem ter efeitos muito negativos na vida de um paciente. Assim sendo, utilizar N-version programming numa calculadora de insulina trás vantagens, pois sendo a quantidade administrada de insulina crítica, esta técnica irá aumentar a probabilidade do valor calculado estar correto.

A calculadora desenvolvida tem como principais funcionalidades o processamento de três valores: a quantidade de insulina a ser administrada depois de uma refeição, a sensibilidade de um paciente à insulina e a quantidade de insulina a ser administrada entre refeições. A calculadora estará disponível numa página web, em que o paciente introduz os valores requeridos e espera por uma resposta. Esta resposta vai variar consoante a decisão de um votador encarregue de decidir o fazer com os resultados obtidos dos diferentes serviços web (desenvolvidos por alunos da disciplina de QCS, diferentes na sua implementação e linguagem).

3 Arquitetura do Software

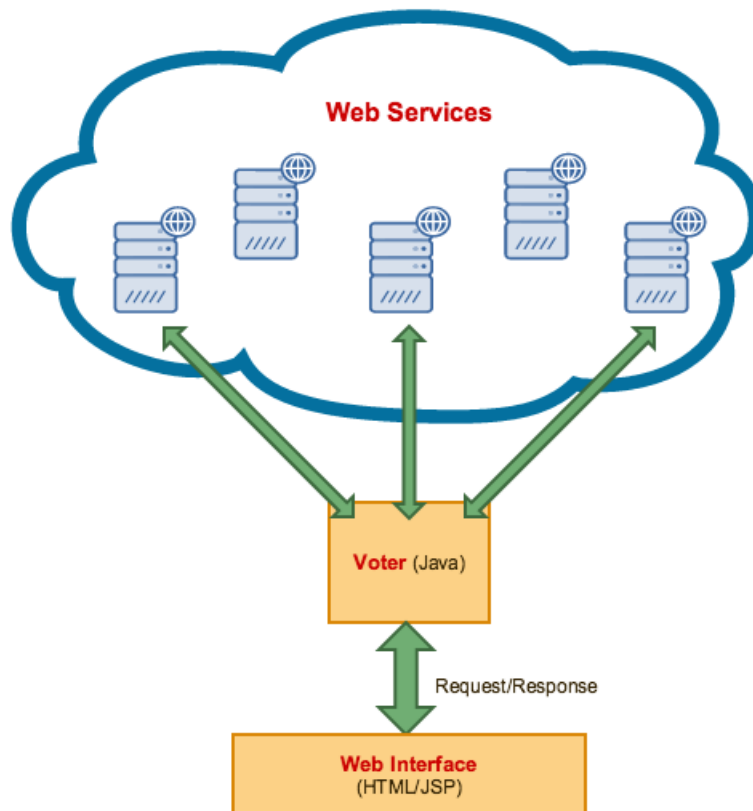


Figura 1: Arquitetura do sistema desenvolvido

A arquitetura do sistema consiste em três aplicações distintas: uma interface web, um votador e um conjunto de web services.

A interface web será uma página web onde o utilizador introduz os dados e aguarda por uma resposta, sendo a única interface com qual tem contacto direto. O votador é a aplicação responsável por toda a logística de decisão de escolha e manuseamento dos resultados retornados pelos web services. Os serviços web são responsáveis por efetuar os cálculos críticos do problema.

Cada uma das aplicações irá ser explicada em pormenor nas secções seguintes.

3.1 Web Service

Relativamente à implementação do web service, decidimos utilizar a linguagem Java porque é uma linguagem que facilita bastante a criação de serviços e métodos web. O nosso serviço web possui três métodos, cada um usado para calcular um valor diferente. O wsdl gerado está disponível em <http://qcs12.dei.uc.pt:8080/insulin?wsdl>.

Em todos os métodos é feita a verificação aos valores introduzidos pelo utilizador, e caso algum destes valores esteja errado ou fora dos limites definidos pelos requisitos funcionais, o serviço web irá retornar 0 sem efetuar nenhum calculo. Caso todos os valores estejam corretos são criadas variáveis temporárias do tipo `BigDecimal` com os mesmos valores recebidos do votador. Usamos o tipo `BigDecimal` para que as operações realizadas entre as variáveis tenham a maior precisão possível (são utilizados os métodos `add`, `subtract`, `divide` e `multiply` da sua classe).

No fim de todas as operações, o valor a ser retornado é convertido para um inteiro através de um cast para o tipo `int`. É feito um arredondamento do valor anteriormente ao cast com o método `round` da classe `Math`.

3.2 Votador

O votador foi elaborado em Java que, por semelhança ao web service, foi escolhido devido à sua facilidade e simplicidade de implementação. Foram criadas 4 classes, sendo 3 delas relativas aos métodos específicos da calculadora e uma geral do votador.

Cada classe contém, para além das variáveis enviadas pela interface web, um array com os links wsdl de todos os web services. Quando o método `getResult` é invocado são criadas `N` threads e cada thread inicializa uma ligação a um web service com um timeout de 1 segundo. Caso o web service retorne um valor válido, o resultado é atualizado e no caso do tempo de execução exceder 1 segundo, o resultado ficará com o valor -999. Depois das `N` threads terem executado, é invocado um método da classe `Voter` que vai verificar os resultados devolvidos de cada web service. Caso não seja encontrado um resultado maioritário (ou caso seja maioritário mas com valor -999), todo este processo é repetido novamente até ao máximo de mais duas vezes (de modo a fazer os 3 segundos máximos).

A classe votador é geral a todas as restantes classes e é a responsável pela decisão do resultado final. O método `voter` irá receber 2 parâmetros, um array de resultados e o número de web services utilizados. Este método vai analisar a frequência dos resultados obtidos, tendo em conta que valores separados por uma unidade serão também incluídos no cálculo. Caso se verifique uma maioria de um valor (mais ou igual a metade de `N`), então é retornado o valor mais baixo, e em caso de não haver maioria é retornado o valor -999, este valor mais tarde é convertido para uma mensagem de erro. É escolhido o valor mais baixo porque caso o valor retornado esteja errado, é menos grave o paciente administrar uma quantidade mais pequena de insulina do que uma quantidade em excesso, que pode provocar danos graves.

3.3 Interface Web

A interface Web é suportada por uma página HTML que disponibiliza ao utilizador acesso às funcionalidades presentes nos Web Services. A página que foi desenvolvida tem três secções, uma para cada tipo de cálculo de insulina propostos neste trabalho. Como é possível reparar durante a utilização, a página não é recarregada a cada mudança de secção, isto porque, todas as secções estão na mesma página e a sua visibilidade difere dependendo da secção que o utilizador selecciona. Para isto, foi usada uma framework em Javascript, o AngularJS, cujo principal objectivo é facilitar o desenvolvimento de *single-page applications*.

Mealtime Insuline Dose - Standard Insulin Sensitivity | Mealtime Insuline Dose - Personal Insulin Sensitivity | Backgroug Insuline Dose

Total grams of carbohydrates in the meal g
Should be an integer between 60 and 120.

Total grams of carbohydrates processed by 1 unit of rapid acting insulin g/unit

Actual blood sugar level measured before the meal mg/dl
Should be an integer between 120 and 250.

Target blood sugar before the meal mg/dl
Should be an integer between 80 and 120.

Today's physical activity level
Should be an integer between 0 and 10.

Add Physical Activity and Blood Sugar Samples:

Physical Activity Level

1.
Should be an integer between 0 and 10.

2.
Should be an integer between 0 and 10.

Drops in Blood Sugar

1.
Should be an integer between 15 and 100.

2.
Should be an integer between 15 and 100.

+Add xRemove

Calculate insulin dose

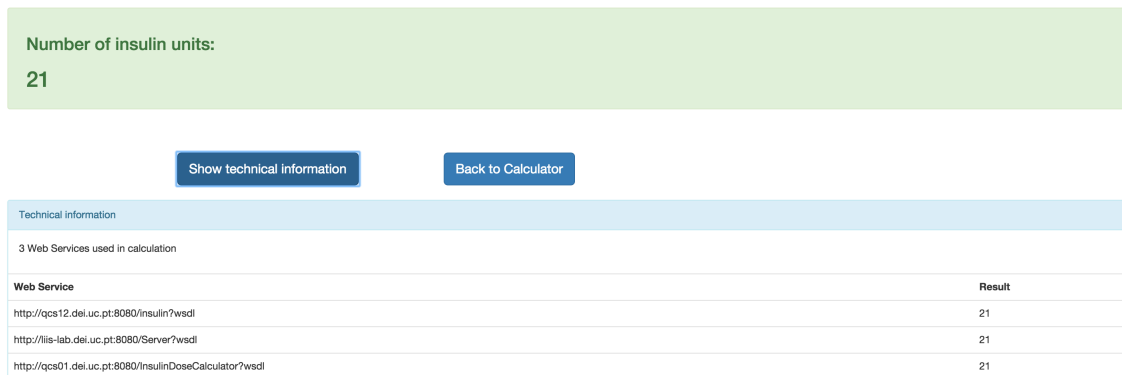
Figura 2: Página principal da interface Web

Dependendo da secção seleccionada o formulário a preencher vai diferir. Todos os campos dos formulários estão protegidos, ou seja, o botão para submeter os dados e efectuar o cálculo só fica disponível (*clicável*) quando todos os campos estiverem não só preenchidos como também correctamente preenchidos. Junto a cada campo existe um mensagem de erro que indica ao utilizador que o campo não está ainda preenchido e/ou que os valores inseridos não são adequados. Foi utilizado de novo o AngularJS para apresentar dinamicamente estas mensagens de erro.

No segundo formulário, *Personal Insulin Sensitivity*, existe adição dinâmica de campos de preenchimento. Para tal recorremos mais uma vez à mesma *framework* para adicionar dinamicamente

código ao HTML. Acharmos aqui que seria importante o utilizador poder remover os campos que adicionou uma vez que se poderá enganar e acrescentar campos além dos pretendidos, assim, adicionámos um botão que invoca uma função em JavaScript e remove a última linha do formulário. Para ambas as funcionalidades, adicionar e remover, foi necessário criar uma variável para manter atualizado o número de linhas existentes no formulário, impedindo que o utilizador adicione mais de 10 ou remova demasiadas linhas sendo que duas são obrigatórias.

Relativamente à apresentação de resultados, estes são apresentados numa página que dá destaque ao resultado final e que contém dois botões. O primeiro permite ao utilizador visualizar os detalhes técnicos relativos ao resultado calculado. O segundo botão permite ao utilizador voltar á página principal e assim efetuar novo cálculo. O botão que permite visualizar os detalhes técnicos chama uma função em JavaScript que altera a visibilidade de uma *div* (elemento HTML) que contém estes dados.



Number of insulin units:
21

Show technical information Back to Calculator

Technical information	
3 Web Services used in calculation	
Web Service	Result
http://qcs12.dei.uc.pt:8080/insulin?wsdl	21
http://iis-lab.dei.uc.pt:8080/Server?wsdl	21
http://qcs01.dei.uc.pt:8080/InsulinDoseCalculator?wsdl	21

Figura 3: Página de apresentação de resultados

Acrescentar ainda que o estilo de todas páginas foi baseado na *framework* Bootstrap que faz uso de HTML, CSS e JS para embelezar e ainda, se quisermos, criar páginas responsivas.

O *deployment* e execução deste projeto, que inclui a interface e o votador, está de momento a ser feita através de um *Web Server* - *Tomcat*.

4 Análise técnica do votador

Um problema conhecido das técnicas de votação de resultados assenta no votador que é um ponto único de falhas. Nesta secção será descrita a forma como foram resolvidos os vários problemas que surgem num votador, em particular o sincronismo e a votação. Depois de descrito o votador será explicada a verificação formal que foi realizada ao votador.

4.1 Resolução dos problemas

Para este trabalho prático foram construídos dois votadores primeiro um em Promela que depois de verificado foi construído outro semelhante desta vez em Java. Os problemas que surgiram nos votadores foram o sincronismo e o método utilizado na votação dos resultados que foram resolvidos da seguinte forma.

- **Sincronização** - cada chamada a um web service é realizada num processo/thread em separado isto para o programa não ficar à espera da resposta em cada ligação aos web services. Além disso de seguida o programa principal só avança para a votação dos resultados obtidos por cada web service quando eles concluírem a ligação ou derem timeout.
- **Votação** - a votação é realizada através da análise das frequências dos valores obtidos, ou seja é escolhido o elemento com maior frequência isto se a frequência dele for maior ou igual a metade do número de valores obtidos, por exemplo para um votador de 3 versões é escolhido um resultado se a frequência dele for igual ou superior a 2. Uma vez que podem surgir problemas entre os vários resultados obtidos devido ao arredondamento no cálculo das frequências foi considerado que dois valores são equivalentes se a sua subtração for menor ou igual a 1. No caso de dois valores terem exatamente a mesma frequência é escolhido o menor valor, por exemplo dois valores separados por uma unidade vão ter a mesma frequência, e portanto visto que se trata de uma aplicação crítica em que escolhido um valor superior ao correto pode ser grave decidimos escolher o menor visto que mesmo em caso de engano é possível compensar numa próxima medida. Além disso no caso de nenhum valor ter a frequência maior ou igual a metade do número total de valores ou no caso de a maioria dos resultados dar timeout é mostrada uma mensagem de erro.
- **Repetição** - sabendo que o sistema não deve demorar mais que 4 segundos a devolver um resultado e que cada processo têm um timeout de 1 segundo o votador foi construído para no caso de não obter um resultado maioritário ele voltar a consultar os web services até ao máximo de mais duas vezes. Assim no máximo ele irá demorar 3 segundos, no timeout mais o tempo de processamento dos dados. Ao voltara ligar-se ao web services

ele irá ligar-se a outros web services, isto para no caso dos web services estarem off-line ou estarem a dar respostas incorretas o votador poder consultar outros web services que poderão retornar uma resposta maioritária.

4.2 Verificação formal do votador

Nesta subsecção será realizada a verificação formal ao votador construído em Promela. Para simular o votador em Promela escolhemos os seguintes resultados simulados dos web services.

- **0** - valor por defeito
- **2, 3** - valores corretos
- **5** - valor incorreto

Além disso para simular o bloqueio de um processo e o timeout que irá ocorrer utilizamos as instruções (`false`) para bloquear o processo e no processo principal a instrução (`timeout`) que desbloquear o processo que esta à espera de respostas isto se não for possível prosseguir de outra forma.

A verificação formal de um programa faz-se em três passos. Estes passos são utilizados para criar um verificador a partir de um modelo especificado. Este verificador irá analisar todo o espaço de busca do modelo. No final retorna o número de erros encontrados. Adicionalmente ainda se pode usar um quarto passo para no caso de existirem erros mostrar os caminhos que levaram ao erros.

```
spin -a modelo.pml
cc pan.c -o pan
./pan
spin -t -p modelo.pml
```

Para a verificação formal foram realizados dois testes diferentes, num primeiro (1.) em que cada web service pode devolver um valor qualquer dentro dos descritos em cima e outro (2.) que vai devolver como resultado final sempre um resultado correto (2 ou 3).

1. Neste teste o resultado final pode ser qualquer um dos valores descritos em cima ou pode

ainda não conseguir encontrar um resultado maioritário, no caso de desacordo na votação ou no caso de timeout como resultado maioritário, e assim o resultado será 999 e no final irá apresentar uma mensagem de erro. Desta forma foram colocadas asserções no final do programa e no final de cada uma das repetições isto para assegurar que o resultado final e os resultados intermédios que se mostraram errados ou não maioritários são iguais aos resultados possíveis do votador. Ao correr o verificador este encontrou 0 erros.

2. No segundo teste, primeiro escolhemos aleatoriamente um dos 3 web services em que este pode dar uma resposta certa, uma resposta errada ou bloquear. Os dois outros web services devolvem sempre um valor correto (2 ou 3). Desta forma sabe-se à prior que o resultado final será 2 ou será 3. No final foi colocada uma asserção que verifica se o resultado final é um correto. Ao correr o verificador para este modelo ele retornou 0 erros.

5 Descrição da divisão das tarefas

No que diz respeito à divisão das tarefas o grupo organizou-se da seguinte forma: o Web Service foi desenvolvido pelo Paulo Pereira; o Votador, Java e Promela, foi um trabalho conjunto entre Paulo Pereira e o Pedro Ferreira, a interface ficou a cargo do José Pedro.

A escrita do relatório foi dividida entre todos, sendo que cada de aplicou mais na escrita da secção relativa à parte do projeto que desenvolveu.

6 Conclusão

Neste trabalho prático pretendeu-se aplicar a programação de N-versões a uma calculadora de insulina para pacientes de diabetes do tipo 2. Esta calculadora foi desenvolvida numa interface web e comunica-se a um votador que por sua vez se comunica aos web services, em que um desses web services foi construído por nós e os outros por outros grupos de alunos. O objetivo deste trabalho foi testar o votador construído por nós com os diferentes web services.

Um dos problemas que surge no votador é o facto de ele ser um *bottleneck*. A técnica escolhida para o votador foi a escolha do elemento com maior frequência, isto se a frequência for igual ou superior a metade de N em que foram considerados que dois valores são equivalentes se a sua subtração for menor ou igual a 1. Outra técnica possível para a construção do votador passava pela mediana ou pela média, no entanto nós consideramos que estas técnicas apresentavam um grande número de erros.

Outro aspeto que foi referido neste relatório foi a resolução de problemas que surgem num votador, tal como a sincronização, a votação e a repetição. De seguida foi realizada uma verificação formal ao votador para isso foi primeiro construído um votador em Promela que depois de verificado foi então construído o votador final desta vez construído em Java que faz a interligação entre os web services e a interface web.