Relatório de Sistemas Críticos

Faculdade de Engenharia da Universidade do Porto

Universidade do Porto

Faculdade de Engenharia

FEUP

Daniel Moreira João Loureiro Ricardo Neves

25 de Maio de 2015

Conteúdo

Introdução

Atualmente com o crescimento e popularização dos computadores, vários sistemas que anteriormente eram controlados por humanos, tornaram-se automatizados, ou seja controlados por *software*, isto permitiu criar sistemas mais fiáveis e eficientes, na medida em que foi reduzida a intervenção humana, e consequentemente os erros associados a esta.

Contudo existem vários tipos de sistemas dependem inteiramente do *software* que os suporta e que têm a necessidade de continuarem operacionais mesmo quando existe algum evento de falha. Caso o sistema não consiga tratar e recuperar alguma falha ou contenha algum defeito, podem resultar consequências graves como por exemplo: um desastre ambiental, estragos em equipamentos dispendiosos ou mesmo a morte de pessoas, estes sistemas são designados de sistemas criticos.

Uma forma melhorar as hipóteses de um sistema critico recuperar de falhas, é ter diversidade no design e na arquitetura, tal facto vai permiter obter alguma redundância o que torna o software mais fiável. Esta técnica é designada *n-version programming*, de uma forma muito sintética são criadas várias variantes do mesmo software criadas geralmente por equipas diferentes que, quando operacionais, vão partilhar os *inputs* e gerar cada uma os seus *outputs*. De seguida e como esses outputs podem conter resultados diferentes, é utilizado um outro módulo de *software* que tem como função receber todos os *outputs* das variantes e decidir um *output* baseado nas inputs das variantes.

É de salientar que o votador é um *single point of failure*, porque á semelhança das variantes não contem diversidade, contudo este deve ser o mais robusto possível e o mais *fault free* possível. Para preencher estes requisitos o votador deve ter um implementação simples e minimalista de modo a evitar erros de programação.

É importante salientar que esta técnica apesar de esta técnica poder evitar erros semelhantes entre as implementações, porém a realização de cada versão aumenta bastante os custo de produção de software nunca garantindo resultados correctos.

1. Solução

1.1 Arquitetura

A arquitetura usada para este projeto foi a arquitetura sugerida no enunciado denominada de *N Version Programming*.

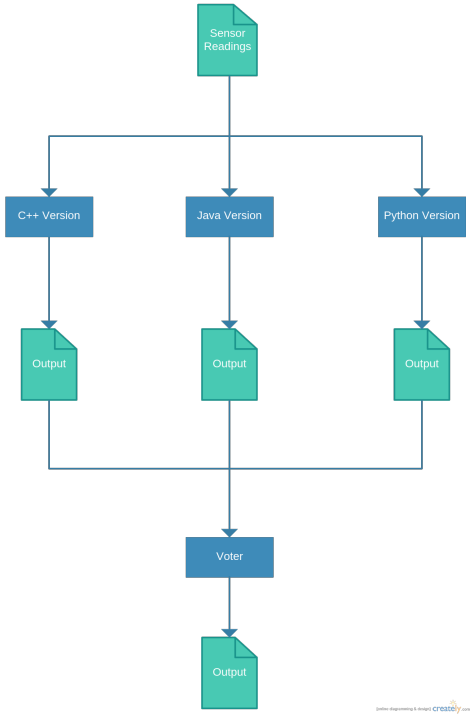

Figura 1.1: Arquitetura

Como demonstra a figura 1.1 existem três versões de implementação da lógica de cálculo da dosagem de insulina a ser administrada ao paciente. Estas implementações vão ler um ficheiro de texto com as leituras da glucose no sangue provenientes dos sensores, calcular as dosagens a injetar a cada três minutos e

escrever essas dosagens num ficheiro de texto.

Tendo os três ficheiros com os resultados das dosagens calculadas por cada versão, foi implementado um votador que lê os três ficheiros e escreve para um ficheiro de saída o valor combinado das três versões.

1.2 Diversidade

Para garantir a diversidade e a unicidade de cada versão, cada uma delas foi distribuida por um dos elementos do grupo. Cada elemento escolheu uma linguagem para implementar a sua versão, as restrições e validações aplicadas ao ficheiro de leituras sem que houvesse qualquer tipo de partilha de informação entre os membros do grupo nesta fase da conceção, à excepção do formato do ficheiro com as dosagens a serem administradas ao paciente.

Depois de implementadas as três versões, os membros do grupo juntaram-se para desenvolver o votador.

As linguagens usadas para implementar as versões foram: *C++, Java, Python* e o votador foi desenvolvido em *Python*

2. Tolerância a Falhas

2.1 Versão 1: C++

A variante de *c++*, possui vários mecanismos de tolerância a falhas, tanto na leitura do ficheiro dos dados de input, como no tratamento e processamento desses mesmos dados.

Em primeiro lugar todos os inputs fora dos limites da função fornecida para o cálculo da glucose são automaticamente descartados. Posteriormente é verificado o estado de cada sensor (operacional ou inoperacional), para tal existem vários parâmetros a ter em conta nomeadamente se o sensor está preso *stuck*, este estado é determinado comparando a leitura atual com a leitura anterior e a leitura anterior é comparada com a imediatamente antes, se a diferença entre estas duas comparações for inferior a 0.01, então o sensor é considerado *stuck*.

Um outro parâmetro a ter em conta é se a sucessão dos valores lidos pelos sensores são díspares, ou seja se existe uma grande variação entre eles, para tal a cada leitura, é verificado se a diferença entre a leitura atual e a anterior é superior ao limite estabelecido de duas unidades. Foi estabelecido este limite porque considera-se que é fisicamente impossível haver tal variação durante 1 minuto. Quando existe uma falha de comunicação, ou seja recebe um input da forma '–', o valor do sensor nesse minuto é automaticamente descartado.

Depois de fazer todas estas validações, o objetivo é juntar as leituras dos dois sensores numa só leitura. Se os dois sensores forem considerados operacionais é usada a média das leituras, se apenas existir um sensor operacional é usada a leitura deste sensor, por fim se nenhum sensor estiver operacional a leitura deste minuto é descartada.

Posteriormente para calcular de 3 em 3 minutos a dose de insulina a ser injetada, se o valor da glucose no sangue em mmol/litro não possuir um valor válido no minuto coicidente com o terceiro minuto, são verificados os valores calculados para os dois minutos anteriores, se o valor da glucose imediatamente anterior for válido é usado esse valor para calcular a dose, senão é usado o anterior a esse. Quando os dois valores anteriores forem válidos, é calculada a média aritmética entre eles, caso nenhum destes seja válido então o programa retorna *FAIL*. No extremo oposto se o valor da glucose o sangue possuir um valor válido no minuto coicidente com o terceiro minuto é utilizado este valor para calcular a dose a injetar no paciente.

2.2 Versão 2: Java

Esta versão do programa começa por ler o ficheiro onde se encontram os dados obtidos pelos sensores. Imediatamente após esta operação, o programa faz um processamento das leituras obtidas pelos sensores rejeitando imediatamente todas as que não se encontrem em confirmidade com o domínio da função de calcúlo da glucose, ou seja valores não pertencentes ao domínio de [1,5].

Uma vez terminado este processamento os dados recolhidos pelos sensores serão todos analisados numa tentativa de validar os resultados recolhidos, tentando que o programa seja o mais tolerante possível a eventuais falhas nos sensores.

Para que tal seja possível, diversas verificações são efectuadas. Em primeiro lugar se para um determinado minuto ambos os sensores forem incapazes de obter uma leitura, seja por uma falha de comuniação, ou não, para esse minuto os valores são automaticamente discartados.

Para as proximas verificações é necessário explicar que os sensores podem-se encontrar em vários estados. Os sensores podem estar presos sempre no mesmo valor, podem estar a receber valores aleatórios.

Seguidamente a esta verificação, é verificado se algum dos sensores está com problemas de comunicação, nao produzindo desta forma leituras, se isto se verificar para algum sensor é verificado se o outro sensor está a produzir uma leitura válida, se sim é considerado o valor deste sensor, caso o segundo sensor esteja preso ou a receber valores aleatórios ambos os valores medidos são rejeitados.

Para o caso em que ambos os sensores se encontrem a receber leituras, são verificados alguns casos especiais, para ambos os sensores, se algum se encontra preso ou a receber leituras aleatórias. Se tal se verificar para ambos os sensores é utilizado ambas as leituras são automaticamente discartadas, se por outro lado tal se verificar apenas para um dos sensores, é utilizada a leitura do outro sensor.

De seguida, se ambos os sensores se encontrarem a produzir valores aparentemente válidos, ou seja que não se enquadrem em nenhum dos casos especiais já mencionados, são ainda assim realizadas algumas validações antes de aceitar algum valor. Para este caso, se os valores recebidos pelos sensores forem muitos distantes (diferença de pelo menos 40%) um do outro, é analisado qual o mais próximo da média das ultimas medições e é considerado esse valor, caso contario, ou seja no caso em que ambos os valores recebidos são validos é considerado o valor médio das duas medições.

Posto isto termina a fase de validação dos valores obtidos pelos sensores. Assim para os valores obtidos de minuto a minuto é calculado o valor da glucose. Completo este passo apenas resta de 3 em 3 minutos calcular a dose de insulina a injetar no paciente. Para tal, são também realizadas algumas verificações. Apesar de só ser injentada insulina de 3 em 3 minutos o processamento da quantidade de insulina a injetar é realizado em blocos de 3 minutos. Ou seja, se para o minuto em que deveria ser injetada insulina existe uma leitura de glucose é calculada a dose a injetar com base nesse mesmo valor. Contudo para

o caso em que por falha dos sensores não existe um valor de glucose para esse minuto, o programa calcula o valor da dose a injetar para o primeiro e segundo valores do bloco de 3 minutos e calcula a média entre eles obtendo assim a dose a injetar, caso tal não seja possível por nalgum dos minutos não haver um valor válido de glucose, o programa tenta calcular a dose tendo por base o valor da glucose do segundo minuto e caso tal não seja possível usando o do primeiro minuto. Contudo pode dar-se o caso de nenhuma destas hipóteses ser possível, nesse caso o programa retorna *FAIL*.

2.3 Versão 3: Python

A variante em *Python* tem vários mecanismos de tolerância a falhas, tanto na leitura do ficheiro com os dados de glucose no sangue como no tratamento desses dados e na escrita do ficheiro com as doses a administrar ao paciente.

Na leitura do ficheiro o programa falha se o mesmo não existir ou se o ficheiro não for passado ao programa.

Depois de obter os dados do ficheiro o programa vai tratar desses mesmos dados seguindo um conjunto de restrições. A primeira restrição aplicada ao conjunto de dados tem a ver com o domínio, só são válidos valores entre 1 e 5, exclusive. A segunda restrição verifica se os valores lidos num dado minuto têm uma diferença superior a 30%, se tiverem o programa incrementa um contador e se o valor desse contador for superior à tolerância de sincronização definida, neste caso 3, ambas as entradas dos sensores são descartadas se a diferença for inferior aos 30% esse contador é decrementado, esta restrição garante que os valores dados por ambos os sensores não são muito diferentes. A terceira restrição verifica se o sensor ficou preso num certo valor, para isso ele verifica se o valor que está a ler de um certo sensor é igual às duas últimas leituras válidas desse mesmo sensor, se sim esse valor é descartado. A última restrição aplicada aos dados de entrada é uma tentativa de minimizar valores aleatórios, esta restrição consiste em descartar valores que variem em mais de 30% da última leitura válida.

Depois dos valores de entrada serem tratados o programa combina os valores dos dois sensores num único valor, esta combinação é feita da seguinte forma: caso as duas leituras sejam inválidas, ou seja, o sensor não leu ou a leitura não era válida segundo as restrições aplicadas, o estado retornado nesse minuto é de *FAIL*; caso uma das leituras seja inválida o estado retornado nesse minuto é a leitura válida; caso ambas sejam válidas é retornada a média entre os dois valores.

Para o cálculo da dosagem a administrar ao cliente o programa apenas utiliza valores de três em três minutos, se o valor do terceiro minuto não for válido é utilizado o valor do segundo minuto e se esse não for válido é utilizado o valor do primeiro minuto, caso nenhum deles seja válido é retornado *FAIL* nessa dosagem. Se houver um valor válido nesses três minutos e se esse valor for maior que 6 é calculada a tendência tendo em conta os últimos valores, até um máximo de 5. Se a têndencia for de descida e maior que 0.4 por minuto é

calculada a dose usando as fórmulas fornecidas no enunciado do trabalho. Ainda é feita uma verificação com as fórmulas da glucose e das suas derivadas com um valor conhecido para verificar se a memória está corrompida.

Calculadas as doses a administrar estas são escritas no ficheiro que foi passado ao programa.

Foi usado uma variação máxima de 30% entre os valores nas restrições, para não permitir variações fora do que é fisicamente possível. Para o cálculo da tendência foi usado um histórico de no máximo 5 leituras, pois os níveis de glucose no sangue tendem a ter uma variação uniforme e lenta e este histórico permite calcular uma tendência muito próxima da realidade.

2.4 Votador

O votador é um mecanismo simples, desenvolvido em *Python*, de escolha das doses a administrar ao paciente, tendo em conta o resultado dos cálculos das três versões acima descritas.

A escolha da dose é feita tendo em conta os seguintes passos: se as todas as doses forem inválidas, o votador retorna *FAIL*; se alguma das doses for válida, ele retorna essa dose; se houver mais do que uma dose válida, o votador verifica se a diferença entre as doses é superior a 2, caso nenhuma das doses cumpra este requisito ele retorna *FAIL*, caso contrário ele retorna a média das duas ou três doses que cumprem esta restrição. A diferença máxima entre as doses de cada versão é de dois, porque permite garantir que as doses não se afastem muito do valor correto, o que é reforçado pelo uso da média entre os valores válidos.

Conclusão

Em retrospetiva, após a realização do trabalho prático proposto, o grupo considera que o mesmo se revelou essencial de modo a permiter uma melhor assimilação dos conhecimentos e conceitos abordados ao longo das aulas teóricas.

Para que tal fosse possível, todos os elementos do grupo tiveram a oportunidade de ter um contacto mais próximo com um sistema crítico, sistema esse em que em caso de falha poderia levar à morte de pessoas, o que se revelou bastante interessante uma vez que permitiu, através da sua implementação, perceber a importância de um sistemas crítico bem como os riscos de uma má implementação.

Por fim é importante salientar a elevada importância que este trabalho teve para uma melhor compreenção das noções e técnicas que permitem em sistemas como estes garantir a tolerância à falha.