Laboratórios de Informática IV

Licenciatura em Engenharia Informática

2010/2011

Proponente: Anabela Tereso ([anabelat@dps.uminho.pt](mailto:anabelat@dps.uminho.pt))

Projecto: Desenvolvimento de uma aplicação para apoiar a selecção de Software de Apoio à Decisão.

Descrição do Projecto: Este projecto tem como objectivo desenvolver uma aplicação que permita apoiar a selecção de Software de Apoio à Decisão. Já foi feito um levantamento dos softwares existentes nessa área (Seixedo, 2009; Tereso and Seixedo, 2010) e pretende-se agora desenvolver uma aplicação que permita a gestão de uma base de dados com informação sobre os vários softwares existentes no mercado, com registo de informações como o nome, a descrição, o preço e outras características importantes, incluindo também um link para a página do software, que possa ser aberta no próprio programa. Pretende-se também que a aplicação, usando técnicas multicritério, e com base em características seleccionadas pelo utilizador, permita ordenar, do melhor para o pior, o conjunto dos softwares existentes na base de dados ou um subconjunto destes.

A aplicação desenvolvida deverá ter uma interface apelativa, forma de armazenamento de dados/resultados, módulo de instalação (Plug and Play) e manual do utilizador, para além de ser convenientemente comentada. O software deverá ser testado e os resultados discutidos.

Detalhe das especificações:

* Linguagem a utilizar na interface: Inglês.
* A aplicação deve ter três módulos distintos:
  + Um módulo de Gestão das Bases de Dados.
  + Um módulo de Gestão dos Softwares (a inserir nas Bases de Dados).
  + Um módulo de Sistema de Apoio à Decisão (para seleccionar softwares dentro dos existentes).
  + Um módulo de Tutorials and Help
* Módulo de Gestão das Bases de Dados
  + Pretende-se ter a possibilidade de criar diferentes Bases de Dados com campos escolhidos pelo utilizador.
  + Pode haver uma Base de Dados (BD) simples (em inglês Basic DB), só com os campos “Software Name” e “WebPage Link”, e ser possível criar outras Bases de Dados com campos adicionais, escolhidos pelo utilizador e gravar separadamente essas BD.
  + O objectivo é que seja possível que o conjunto de informações a registar para caracterizar cada software possa ser alterado, ou seja, que possa ser dada a hipótese ao utilizador de seleccionar as características que pretende usar para descrever o conjunto dos softwares, isto é, pretende-se que seja possível alterar a estrutura da Base de Dados, ou seja, o nome, o número de campos e as características de cada campo. Pretende-se também que seja possível adicionar novos campos, remover campos, etc. Neste módulo, pode aparecer um menu com as seguintes opções: Data Base – New, Open, Save, Save As, Exit. Deve-se poder escolher se é uma “Basic DB” ou uma “Extended DB”. Deve ser possível carregar informação de uma Base de Dados já existente quando se cria uma nova.
  + Exemplo de informações (critérios de decisão) a registar para cada software:
    - Software name: 60 caracteres (Basic DB)
    - WebPage Link: 200 caracteres (Basic DB)
    - Compatibility between Operating Systems: Yes/No
    - Cost of a license: Valor real em Euros
    - Interaction with user: Bad/Fair/Good/Very Good/Excellent
    - User Manual: Yes/No
    - Tutorials: Yes/No
    - Application Examples: Yes/No
    - Online Help: Yes/No
    - Free Version: Yes/No
  + Para cada campo (excepto para os campos da “Basic DB”) é necessário pedir o nome e o tipo. Se for um valor numérico, pedir o limite inferior e o limite superior permitidos. Se for um conjunto de caracteres, pedir as diferentes possibilidades de conteúdo. Por exemplo, no caso do “Interaction with user” deve ser definida a lista de valores esperados, ou seja, [Bad, Fair, Good, Very Good, Excellent], bem como uma classificação. Nessa classificação deve-se definir a escala (valor mínimo e máximo), por exemplo, (2..10) e atribuir valores depois nessa escala aos atributos definido pelo utilizador, do tipo:

|  |  |
| --- | --- |
| Attribute value | Classification (2..10) |
| Bad | 2 |
| Fair | 4 |
| Good | 6 |
| Very Good | 8 |
| Excellent | 10 |

* Módulo de Gestão dos Softwares
  + Depois de aberta uma Base de Dados com, por exemplo, os critérios acima definidos, pretende-se poder adicionar novos softwares, visualizar os softwares existentes, eliminar softwares, etc.
  + Para testar o software, criar uma Base de Dados com os critérios acima e adicionar alguns Softwares (ver dissertação mestrado - tabela 6.1 e completar a lista com os softwares listados no artigo apresentado no EURO 2010). Pretende-se ter para os Softwares o seguinte menu: Edit Software List, View Software WebPage. Dentro do Edit Software List devem aparecer os softwares já existentes e ser possível eliminar algum, inserir novos ou simplesmente visualiza-los.
  + Quando se selecciona View Software WebPage, os softwares devem ser apresentados numa lista do lado esquerdo do ecran (só o nome) e a página a ele associada numa janela a ocupar o resto do ecran. Deve ser possível navegar nessa página e depois retornar o controlo à aplicação, quando pretendido.
* Módulo do Sistema de Apoio à Decisão
  + Permitir ao utilizador seleccionar os critérios a serem usados na decisão de seleccionar o melhor software, por exemplo “Interaction with user” e “Cost”.
  + Permitir ao utilizador seleccionar o conjunto dos softwares que pretende analisar, por exemplo, A, B e C.
  + Definição dos pesos - Etapa 1: Tem que se definir os pesos de cada critério na escolha do melhor software. Podem-se seleccionar dois métodos alternativos para atribuir esses pesos (SMART ou AHP):
    - SMART
      1. Assign 10 points to the least important attribute
      2. Give points (>10) to reflect the importance of the attribute relative to the least important attribute
    - AHP
* Para o método AHP, a atribuição dos pesos, para cada critério, é feito usando um método de comparação par-a-par, com uma escala de 1 a 9 proposta por Saaty (nota: também se podem usar valores intermédios):



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pair-wise comparison matrix | | |  |  |  |
|  |  |  |  |  |  |
|  | BEST SOFTWARE | Interaction with user | Cost |  |  |
|  | Interaction with user | 1 | 1/4 |  |  |
|  | Cost | 4 | 1 |  |  |
|  | Sum | 5 | 1.25 |  | Pesos dos |
|  |  |  |  |  | atributos |
| Normalized matrix | |  |  |  | ↓ |
|  | BEST SOFTWARE | Interaction with user | Cost | Sum | Mean |
|  | Interaction with user | 0.2 | 0.2 | 0.4 | 0.2 |
|  | Cost | 0.8 | 0.8 | 1.6 | 0.8 |
|  | Sum |  |  |  |  |

* Deve-se calcular a taxa de consistência, tentar melhora-la, e caso não se consiga uma taxa de consistência boa, avisar o utilizador que deve inserir outros valores na matriz de comparações, mas não obrigar (ver explicação do cálculo da taxa de consistência em anexo).
* A análise final (ver abaixo) é feita da mesma forma que se faria usando o método SMART, mas usando o peso dos atributos obtidos pelo AHP.
  + Definição dos pesos - Etapa 2: Depois é necessário definir as prioridades ou pesos para cada software dentro de cada critério. Para cada critério é necessário definir o método a usar para definir as prioridades (ValueFn ou AHP).
    - Se for usado o método ValueFn, é necessário perguntar se se pretende maximizar ou minimizar o critério.  
      1. No caso de maximização usa-se a seguinte fórmula para calcular as prioridades:
      2. No caso de minimização usa-se a seguinte fórmula para calcular as prioridades:
      3. Por exemplo, o critério “Interaction with user” devia ser maximizado. Supondo que temos três softwares A, B e C, com valores de “Interaction with user” iguais a [1, 3, 5], o valor das prioridades para este critério seriam [0, 0.5, 1].
      4. Por exemplo, o critério “Cost” devia ser minimizado. Para os três softwares A, B e C, com valores de “Cost” iguais a [100, 800, 1000], o valor das prioridades para este critério seriam [1, 0.222, 0].
      5. Depois é necessário normalizar os valores (de forma a que a soma seja igual a 1).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software | Interaction with user |  | Priorities | Normalized Priorities |
| A | 1 |  | 0 | 0.000 |
| B | 3 |  | 0.5 | 0.333 |
| C | 5 |  | 1 | 0.667 |
| Min | 1 | Sum | 1.5 | 1 |
| Max | 5 |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Software | Cost |  | Priorities | Normalized Priorities |
| A | 100 |  | 1 | 0.818 |
| B | 800 |  | 0.22 | 0.182 |
| C | 1000 |  | 0 | 0.000 |
| Min | 100 | Sum | 1.22 | 1 |
| Max | 1000 |  |  |  |

* + Usando o método AHP, a atribuição das prioridades às alternativas, para cada critério, é feito usando o método de comparação par-a-par, como visto acima:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Interaction with user | A | B | C |  |  |
|  | A | 1 | 1/2 | 1/4 |  |  |
|  | B | 2 | 1 | 1/2 |  |  |
|  | C | 4 | 2 | 1 |  |  |
|  | Sum | 7 | 3.5 | 1.75 |  |  |
|  |  |  |  |  |  | Pesos dos |
|  |  |  |  |  |  | atributos |
| Normalized matrix |  |  |  |  |  | ↓ |
|  | Interaction with user | A | B | C | Sum | Mean |
|  | A | 0.14 | 0.14 | 0.14 | 0.43 | 0.14 |
|  | B | 0.29 | 0.29 | 0.29 | 0.86 | 0.29 |
|  | C | 0.57 | 0.57 | 0.57 | 1.71 | 0.57 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Cost | A | B | C |  |  |
|  | A | 1 | 4 | 6 |  |  |
|  | B | 1/4 | 1 | 2 |  |  |
|  | C | 1/6 | 1/2 | 1 |  |  |
|  | Sum | 1.42 | 5.50 | 9.00 |  |  |
|  |  |  |  |  |  | Pesos dos |
|  |  |  |  |  |  | atributos |
| Normalized matrix |  |  |  |  |  | ↓ |
|  | Cost | A | B | C | Sum | Mean |
|  | A | 0.71 | 0.73 | 0.67 | 2.10 | 0.70 |
|  | B | 0.18 | 0.18 | 0.22 | 0.58 | 0.19 |
|  | C | 0.12 | 0.09 | 0.11 | 0.32 | 0.11 |

* + Análise Final (por níveis assumindo a utilização do AHP nas duas etapas):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Level |  |  |  |  |  |  |  |  |  |
| 0 | Best Software |  |  |  |  |  |  |  |  |
| 1 |  | Interaction with user | 0.2 |  |  | 0.2\*… |  |  |  |
| 2 |  |  | A | 0.143 |  | 0.029 |  |  |  |
| 2 |  |  | B | 0.286 |  | 0.057 |  | Final Priorities | Ranking |
| 2 |  |  | C | 0.571 |  | 0.114 | A | 0.589 | 1 |
| 1 |  | Cost | 0.8 |  |  | 0.8\*… | B | 0.212 | 2 |
| 2 |  |  | A | 0.700 |  | 0.560 | C | 0.200 | 3 |
| 2 |  |  | B | 0.194 |  | 0.155 |  |  |  |
| 2 |  |  | C | 0.107 |  | 0.085 |  |  |  |

* + Análise Final (por níveis assumindo a utilização do SMART na primeira etapa e do ValueFn na segunda etapa):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Level |  |  |  |  |  |  |  |  |  |
| 0 | Best Software |  |  |  |  |  |  |  |  |
| 1 |  | Interaction with user | 0.4 |  |  | 0.4\*… |  |  |  |
| 2 |  |  | A | 0.000 |  | 0.000 |  |  |  |
| 2 |  |  | B | 0.333 |  | 0.133 |  | Final Priorities | Ranking |
| 2 |  |  | C | 0.667 |  | 0.267 | A | 0.491 | 1 |
| 1 |  | Cost | 0.6 |  |  | 0.6\*… | B | 0.242 | 3 |
| 2 |  |  | A | 0.818 |  | 0.491 | C | 0.267 | 2 |
| 2 |  |  | B | 0.182 |  | 0.109 |  |  |  |
| 2 |  |  | C | 0.000 |  | 0.000 |  |  |  |

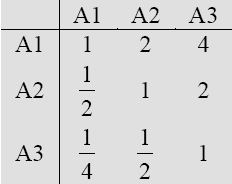
* Construir um módulo de Tutorials and Help

**ANEXO**

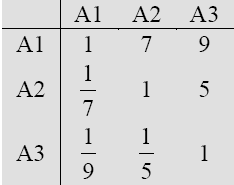
**Cálculo da taxa de consistência**

A taxa de consistência (*TC*) é um indicador matemático, que indica a consistência da comparação efectuada. Conforme sugerido por Saaty, esta taxa deve ser no máximo igual a 0.10.

Exemplo matriz de comparações consistente



Exemplo matriz de comparações inconsistente



A primeira matriz é consistente pois:

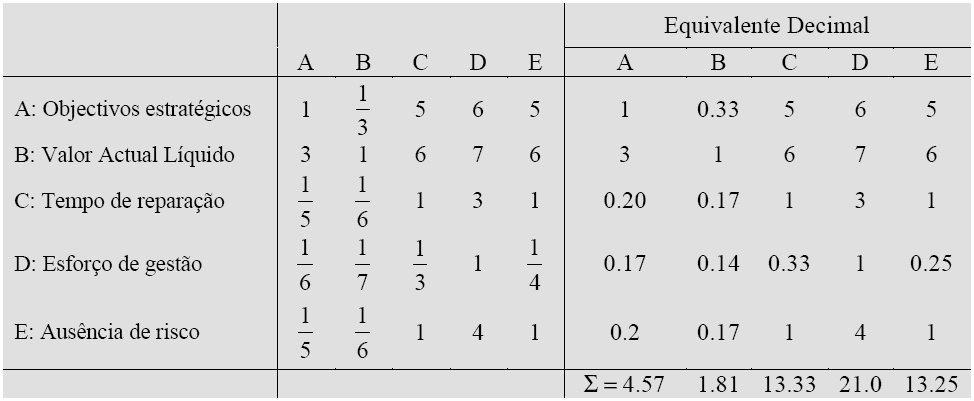
A1=2×A2 e A2=2×A3, logo A1 deveria ser 4×A3, o que realmente acontece.

Já, na segunda matriz, esta propriedade não se verifica. Assim:

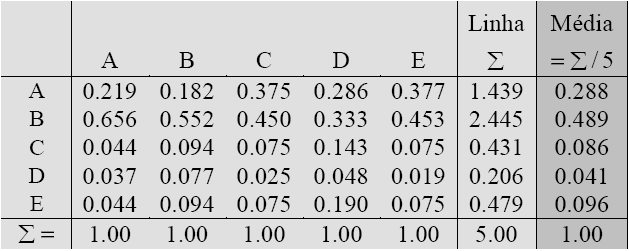
A1=7×A2 e A2=5×A3, logo A1 deveria ser 35×A3, o que não se verifica.

Considere-se o seguinte exemplo:

Matriz de comparações entre os atributos



Matriz normalizada de comparações entre os atributos e cálculo dos pesos de prioridade



Vamos exemplificar, para o exemplo em estudo, o cálculo da *TC*. Seja [A] a matriz de comparações e [B] o vector próprio calculado (Média). Multiplica-se então a matriz [A] pelo vector [B], para obter um novo vector [C].



De seguida, divide-se cada elemento do vector [C] pelos elementos correspondente do vector [B], obtendo-se um novo vector [D].



Depois calcula-se a média dos valores do vector [D], designado por λmax, que é uma aproximação ao maior valor próprio.



O índice de consistência (*IC*) de uma matriz de tamanho *N* é calculado pela seguinte fórmula:



Saaty, calculou por simulação, uma série de índices aleatórios (IA), para vários tamanhos de matrizes, conforme é apresentado a seguir:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | … |
| IA | 0.00 | 0.00 | 0.58 | 0.90 | 1.12 | 1.24 | 1.32 | 1.41 | 1.45 | 1.49 | 1.51 | … |

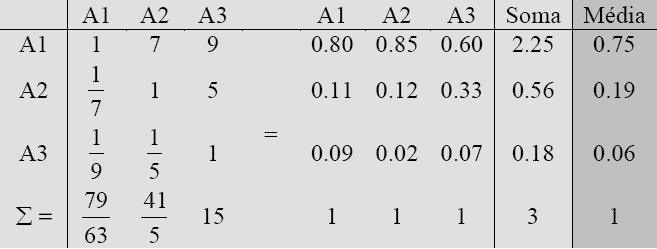
Para o exemplo acima, *IA=1.12*. A taxa de consistência (*TC*) pode ser então calculada usando a seguinte expressão:



Uma vez que o valor calculado está dentro dos padrões estabelecidos (≤0.10) podemos concluir que a comparação efectuada para obter os pesos dos atributos é consistente.

Se a taxa de consistência obtida for superior a 0.10, utiliza-se um método iterativo para melhorar a taxa de consistência. Vamos aplicar esse método à matriz inconsistente apresentada abaixo.

Começa-se por normalizar a matriz e calcular o peso dos atributos.





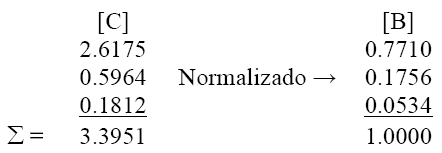
Como podemos ver, a taxa de consistência é superior a 0.10, logo a aproximação calculada não é aceitável.

Devemos então melhorar a aproximação ao vector próprio através do seguinte procedimento iterativo:

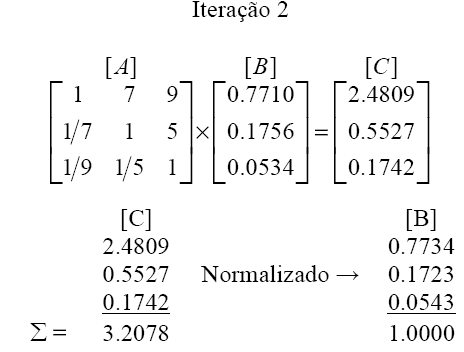
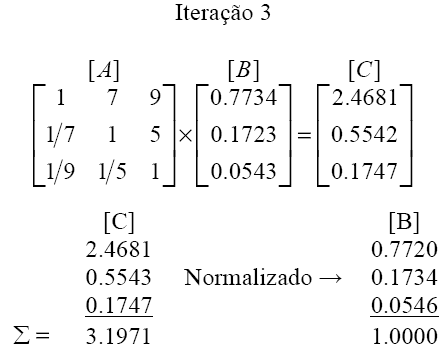
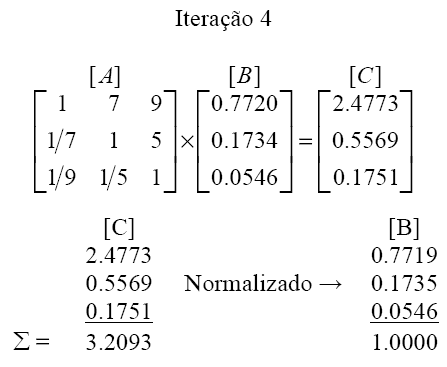
* + 1. Começamos por obter o conjunto inicial de pesos de prioridade, usando o método apresentado anteriormente.
    2. Multiplica-se a matriz de comparações inicial [A] pelo vector com os pesos de prioridade obtido [B], obtendo-se assim o vector [C]=[A]×[B].
    3. O novo vector [B] de pesos de prioridade resulta do vector [C] após normalização, para somar 1.
    4. Repetem-se os pontos 2 e 3 até obter a precisão pretendida, ou seja, até a diferença entre os valores de dois vectores sucessivos ser inferior a um determinado valor.

Vamos então aplicar este procedimento ao exemplo anterior. Os passos 1 e 2 já formam aplicados. Vamos então de seguida normalizar o vector [C] como indica o passo 3:

Iteração 1



Repetem-se os pontos 2 e 3, até se obter a precisão pretendida:

Podemos ver que a diferença decimal entre as prioridades da iteração 3 e da 4 é no máximo de 0.0001. Se essa for a precisão pretendida, podemos então parar o processo iterativo.

Vamos então calcular a taxa de consistência para esta aproximação:



A taxa de consistência não baixou o suficiente! Devemos rever a matriz inicial de comparações entre os atributos, e tentar de novo.