Data Structures and Algorithms of Intelligent Web-based System for Modular Design

Ivan Mustakerov, Daniela Borissova

Introduction

Recently, new product development became more and more 
competitive and globalized, and the design phase is critical for the 
product success.

It is recognized that modular design is more agile and competitive in 
rapidly changing market environments. Modular design offers some 
essential advantages that can be summarized as:

• stabilized design of modular item reduces time for development of 
  final product;
• extended variety of products;
• high speed market response;
• considerable flexibility in products design;
• easy service, fast diagnosis of fault and replacements;
• simplified material planning;
• less inventory due to easily available modular sub-assemblies
Introduction

The main principle of modular design is to break systems into discrete interchangeable modules with well-defined interfaces to ensure functional compatibility and to provide required functional capabilities.

Examples of modular design implementation are everywhere around us:

- cars and vehicles;
- computers;
- buildings;
- and much, much more..........

In many application areas customer preferences change rapidly and this requires shorter product development cycles. The modularization can play a key role in achieving mass customization and this is crucial in today’s competitive global market environments.
Intelligence in modular system design is recognized as intelligent software that should be able to assist and advise the designer during the decision-making process. The recent computer technology have made it possible to store vast amounts of data in electronic form. In reality, more emphasis was placed on storage efficiency rather than processing effectiveness.

That defines existence of a data-processing bottleneck and is one of the primary reasons for evolution of software intelligence. One way to overcome the data-processing bottleneck is to define proper data structures that assist developing of efficient data processing algorithms.

In the current paper, data structures and algorithms for Web-based system for modular design are proposed for support of intelligent design decisions making.
The decision making for any modular design is based on proper choices of modules. This choice should correspond to all of the given functional requirements including compatibility restrictions for the designed system and should meet some design criteria.

A typical example of using the advantages of modularization is personal computers (PC) configuration design. It is well known example and is used as case study to illustrate data structures and algorithms that could be applied to modular systems design.

In this case study, motherboard (MB), processor (CPU) and memory (RAM) can be regarded as basic modules defining most of the system functional capabilities.

- MB = \{Mb_i, i = 0, 1, \ldots, 6\}
- CPU = \{CPU_i, i = 0, 1, \ldots, 6\}
- RAM = \{RAM_i, i = 0, 1, \ldots, 6\}
## TABLE I. MB-DATA \([i,j]\)

<table>
<thead>
<tr>
<th>(i)</th>
<th>MB</th>
<th>MB RAM, slots</th>
<th>MB Max RAM [GB]</th>
<th>Price BGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ASROCK P45XE-WIFI/VP45</td>
<td>4</td>
<td>16</td>
<td>178.50</td>
</tr>
<tr>
<td>1</td>
<td>ASROCK G31M-S/G31</td>
<td>2</td>
<td>8</td>
<td>69.00</td>
</tr>
<tr>
<td>2</td>
<td>Gigabyte G31M-ES2C/G31</td>
<td>2</td>
<td>4</td>
<td>74.50</td>
</tr>
<tr>
<td>3</td>
<td>Gigabyte X48-DS6X/X48</td>
<td>4</td>
<td>8</td>
<td>227.50</td>
</tr>
<tr>
<td>4</td>
<td>ASUS P5S800-VM/SIS661FX</td>
<td>2</td>
<td>2</td>
<td>36.00</td>
</tr>
<tr>
<td>5</td>
<td>ASUS P7P55D/PRO/P55</td>
<td>4</td>
<td>16</td>
<td>315.50</td>
</tr>
<tr>
<td>6</td>
<td>INTEL DX58SO/X58/BOX</td>
<td>4</td>
<td>8</td>
<td>421.50</td>
</tr>
</tbody>
</table>

## TABLE II. CPU-DATA \([j,l]\)

<table>
<thead>
<tr>
<th>(j)</th>
<th>CPU</th>
<th>CPU Core number</th>
<th>CPU Clock [GHz]</th>
<th>Price [BGL]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CELERON-D 347</td>
<td>1</td>
<td>3.06</td>
<td>59.50</td>
</tr>
<tr>
<td>1</td>
<td>Core DUO E5200 /800/2M BOX</td>
<td>2</td>
<td>2.50</td>
<td>109.00</td>
</tr>
<tr>
<td>2</td>
<td>Core2 DUO E7600 /1066/3M BOX</td>
<td>2</td>
<td>3.06</td>
<td>251.00</td>
</tr>
<tr>
<td>3</td>
<td>Core2 QUAD Q8200S/1333 BOX</td>
<td>4</td>
<td>2.33</td>
<td>381.00</td>
</tr>
<tr>
<td>4</td>
<td>C i5-750 /8M/BOX/</td>
<td>4</td>
<td>2.66</td>
<td>371.00</td>
</tr>
<tr>
<td>5</td>
<td>C i7-860 /8M/BOX/</td>
<td>4</td>
<td>2.80</td>
<td>531.00</td>
</tr>
<tr>
<td>6</td>
<td>C i7-940 /8M/BOX/</td>
<td>4</td>
<td>2.93</td>
<td>979.50</td>
</tr>
</tbody>
</table>

## TABLE III. RAM-DATA \([j,l]\)

<table>
<thead>
<tr>
<th>(j)</th>
<th>RAM</th>
<th>RAM Size [GB]</th>
<th>RAM Frequency [MHz]</th>
<th>Price [BGL]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DDR A-DATA</td>
<td>1</td>
<td>400</td>
<td>55.00</td>
</tr>
<tr>
<td>1</td>
<td>DDR2 KINGSTON</td>
<td>1</td>
<td>667</td>
<td>41.50</td>
</tr>
<tr>
<td>2</td>
<td>DDR2 KINGSTON</td>
<td>1</td>
<td>1066</td>
<td>64.50</td>
</tr>
<tr>
<td>3</td>
<td>DDR3 KINGSTON</td>
<td>1</td>
<td>1066</td>
<td>42.50</td>
</tr>
<tr>
<td>4</td>
<td>DDR2 KINGSTON</td>
<td>2</td>
<td>800</td>
<td>77.50</td>
</tr>
<tr>
<td>5</td>
<td>DDR3 A-DATA</td>
<td>2</td>
<td>1333</td>
<td>77.50</td>
</tr>
<tr>
<td>6</td>
<td>DDR3 HYPER X KINGSTON</td>
<td>2</td>
<td>1600</td>
<td>83.00</td>
</tr>
</tbody>
</table>
One of the major problems in modular design is meeting the **compatibility dependences** between modules.

This is well illustrated by the PC configuration case study - each **MB has specific requirements about CPU and RAM modules** it can support. The CPU and RAM modules have also corresponding requirements for MBs they can be installed on.

![Graphical illustration of PC modules relationship](image)

**Fig. 1.** Graphical illustration of PC modules relationship
The compatibility dependencies between MB and CPU can be described as 2D array:

<p>| TABLE IV. MB-CPU[i,j] – 2D ARRAY OF MB AND CPU COMPATIBILITY |
|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|</p>
<table>
<thead>
<tr>
<th>CPU0</th>
<th>CPU1</th>
<th>CPU2</th>
<th>CPU3</th>
<th>CPU4</th>
<th>CPU5</th>
<th>CPU6</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MB6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
In most cases of modular design the compatibility dependencies are more complex to be described by single 2D array. PC configuration case study demonstrates this by existence of additional compatibility requirements for RAM modules choice.

Each particular MB has a certain number of memory slots, maximal memory size and RAM modules frequency it can support. On the other side, each RAM module has certain memory size and operating frequency.

For cases like this, a possible solution is to use several 2D arrays, for example, three 2D arrays for MB and RAM compatibility – one for memory slots, one for max memory size and one for RAM frequency – Table V, Table VI and Table VII.
### TABLE V. MB-RAM-1[i,j] – 2D Array of MB RAM Frequency Compatibility

<table>
<thead>
<tr>
<th></th>
<th>RAM0</th>
<th>RAM1</th>
<th>RAM2</th>
<th>RAM3</th>
<th>RAM4</th>
<th>RAM5</th>
<th>RAM6</th>
<th>RAM7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB0</td>
<td>0</td>
<td>667</td>
<td>1066</td>
<td>0</td>
<td>800</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB1</td>
<td>0</td>
<td>667</td>
<td>0</td>
<td>0</td>
<td>800</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB2</td>
<td>0</td>
<td>667</td>
<td>1066</td>
<td>0</td>
<td>800</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB3</td>
<td>0</td>
<td>667</td>
<td>1066</td>
<td>0</td>
<td>800</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB4</td>
<td>400</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1066</td>
<td>0</td>
<td>1333</td>
<td>1600</td>
<td>1600</td>
</tr>
<tr>
<td>MB6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1066</td>
<td>0</td>
<td>1333</td>
<td>1600</td>
<td>1600</td>
</tr>
</tbody>
</table>

### TABLE VI. MB-RAM-2[i,j] – 2D Array of MB RAM Slots Number Compatibility

<table>
<thead>
<tr>
<th></th>
<th>RAM0</th>
<th>RAM1</th>
<th>RAM2</th>
<th>RAM3</th>
<th>RAM4</th>
<th>RAM5</th>
<th>RAM6</th>
<th>RAM7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB3</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>MB5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>MB6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### TABLE VII. MB-RAM-3[i,j] – 2D Array of MB maxRAM Size Compatibility

<table>
<thead>
<tr>
<th></th>
<th>RAM0</th>
<th>RAM1</th>
<th>RAM2</th>
<th>RAM3</th>
<th>RAM4</th>
<th>RAM5</th>
<th>RAM6</th>
<th>RAM7</th>
</tr>
</thead>
<tbody>
<tr>
<td>MB0</td>
<td>0</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB1</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB2</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB3</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>MB5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>MB6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>
Using the described data structures for MB, CPU and RAM modules, a **generalized algorithm** for modular system design is shown. It describes **4 decision making scenarios** as 4 algorithm branches – A1, A2, A3, A4.

![Diagram](image)

Fig. 2. **Generalized algorithm for PC modular design**
Algorithm A1 starts with selection of a concrete CPU module and this selection defines certain column in 2D array MB-CPU[i,j]. The list of compatible MBs with this CPU is created by replacing all non zero values of the chosen CPU's column with names of MBs stored in the first column of 2D array MB-Data[i,j].

A1: Algorithm realization for CPU => MB choice of compatible modules
1. Given the array CPU-Data[j] set j=c;  \(<\) choice of particular CPU
2. for i = 0 to number of MBs do \(<\) create list of compatible MBs for "drop-down" menu
   3. if MB-CPU[i,c] > 0 then
   4. \[ MBc[i] = MB-Data[i,0]; \(<\) selectable item of "drop-down" list
   5. else MBc[i] = ""; \(<\) empty (not selectable) item of "drop-down" list
   6. set i = m; \(<\) choice of particular MB with index m;
   7. else end
For example, if \( CPU_2 \) is chosen

<table>
<thead>
<tr>
<th>( MB )</th>
<th>( CPU_0 )</th>
<th>( CPU_1 )</th>
<th>( CPU_2 )</th>
<th>( CPU_3 )</th>
<th>( CPU_4 )</th>
<th>( CPU_5 )</th>
<th>( CPU_6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( MB_0 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( MB_1 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( MB_2 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( MB_3 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( MB_4 )</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( MB_5 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( MB_6 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
For example, if $MB_3$ is chosen
### TABLE V. $MB_{RAM-1}[i,j]$ – 2D Array of MB RAM Frequency Compatibility

<table>
<thead>
<tr>
<th>$RAM_0$</th>
<th>$RAM_1$</th>
<th>$RAM_2$</th>
<th>$RAM_3$</th>
<th>$RAM_4$</th>
<th>$RAM_5$</th>
<th>$RAM_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MB_0$</td>
<td>0</td>
<td>667</td>
<td>1066</td>
<td>0</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>$MB_1$</td>
<td>0</td>
<td>667</td>
<td>0</td>
<td>0</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>$MB_2$</td>
<td>0</td>
<td>667</td>
<td>1066</td>
<td>0</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>$MB_3$</td>
<td>0</td>
<td>667</td>
<td>1066</td>
<td>0</td>
<td>800</td>
<td>0</td>
</tr>
<tr>
<td>$MB_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$MB_5$</td>
<td>0</td>
<td>0</td>
<td>1066</td>
<td>0</td>
<td>1333</td>
<td>1600</td>
</tr>
<tr>
<td>$MB_6$</td>
<td>0</td>
<td>0</td>
<td>1066</td>
<td>0</td>
<td>1333</td>
<td>1600</td>
</tr>
</tbody>
</table>

### TABLE VI. $MB_{RAM-2}[i,j]$ – 2D Array of MB RAM Slots Number Compatibility

<table>
<thead>
<tr>
<th>$RAM_0$</th>
<th>$RAM_1$</th>
<th>$RAM_2$</th>
<th>$RAM_3$</th>
<th>$RAM_4$</th>
<th>$RAM_5$</th>
<th>$RAM_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MB_0$</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>$MB_1$</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$MB_2$</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>$MB_3$</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>$MB_4$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$MB_5$</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$MB_6$</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

### TABLE VII. $MB_{RAM-3}[i,j]$ – 2D Array of MB maxRAM Size Compatibility

<table>
<thead>
<tr>
<th>$RAM_0$</th>
<th>$RAM_1$</th>
<th>$RAM_2$</th>
<th>$RAM_3$</th>
<th>$RAM_4$</th>
<th>$RAM_5$</th>
<th>$RAM_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$MB_0$</td>
<td>0</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>$MB_1$</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>$MB_2$</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>$MB_3$</td>
<td>0</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>$MB_4$</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$MB_5$</td>
<td>0</td>
<td>0</td>
<td>16</td>
<td>0</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>$MB_6$</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

AComIn: Advanced Computing for Innovation

http://www.iict.bas.bg
A3: Algorithm realization for MB $\Rightarrow$ CPU choice of compatible modules

1. Given the array $MB-Data[i]$ set $i = m$; \textless choice of particular MB
2. \textbf{for} $j = 0$ \textbf{to} number of CPUs \textbf{do}
3. \textbf{if} $MB-CPU[m, j] > 0$ \textbf{then} \textless create list of compatible CPUs for
   "drop-down" menu
   \hspace{1cm} \hspace{1cm} $CPUm[j] = CPU-Data[i, 0]$; \textless selectable item of "drop-down" list
4. \hspace{1cm} \textbf{else} $CPUm[j] =$ ""; \textless empty (not selectable) item of "drop-down" list
5. \hspace{1cm} \textbf{end}
6. \hspace{1cm} \textbf{set} $j = c$; \textless choice of particular CPU with index $c$
7. \hspace{1cm} \textbf{end}
If **MB\textsubscript{3}** is chosen

<table>
<thead>
<tr>
<th>MB\textsubscript{0}</th>
<th>MB\textsubscript{1}</th>
<th>MB\textsubscript{2}</th>
<th>MB\textsubscript{3}</th>
<th>MB\textsubscript{4}</th>
<th>MB\textsubscript{5}</th>
<th>MB\textsubscript{6}</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU\textsubscript{0}</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CPU\textsubscript{1}</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CPU\textsubscript{2}</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CPU\textsubscript{3}</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CPU\textsubscript{4}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CPU\textsubscript{5}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>CPU\textsubscript{6}</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
In A4 algorithm the array $MB-RAM-1[i,j]$ is used for defining list of MBs compatible with chosen RAM while arrays $MB-RAM-2[i,j]$ and $MB-RAM-3[i,j]$ are used to check if the entered number of RAM modules correspond to MB's slots number and to maximal supported RAM size:

<table>
<thead>
<tr>
<th>A4: Algorithm realization for RAM =&gt; MB choice of compatible modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Given the array $RAM-Data[j]$ set $j = r$; $&lt;$ choice of particular RAM,</td>
</tr>
<tr>
<td>2. for $i = 0$ to number of MBs do</td>
</tr>
<tr>
<td>3. $&lt;$ create list of compatible MBs for &quot;drop-down&quot; menu</td>
</tr>
<tr>
<td>4. $MBBr[i] = MB-Data[i,0]$; $&lt;$ selectable item of &quot;drop-down&quot; list</td>
</tr>
<tr>
<td>5. else $MBBr[i] = &quot;&quot;$; $&lt;$ empty (not selectable) item of &quot;drop-down&quot; list</td>
</tr>
<tr>
<td>6. set $i = m$; $&lt;$ choice of particular MB with index m</td>
</tr>
<tr>
<td>7. get $N_{ram}$ $&lt;$ get entered RAM modules number</td>
</tr>
<tr>
<td>8. if $N_{ram} &gt; MB-RAM-2[m,j]$ then message goto 7</td>
</tr>
<tr>
<td>9. else if $N_{ram} * RAM-Data[r,1] &gt; MB-RAM-3[m,r]$ then message goto 7</td>
</tr>
<tr>
<td>10. else end</td>
</tr>
</tbody>
</table>
The described data structures can be used also for algorithmic realization of intelligent choice of modules in design process. For example, if the DM needs processor with number of processor cores equal to $N_{\text{core}}$ the following algorithm will be used:

A5a: Algorithmic realization of intelligent CPU choice

1. get $N_{\text{core}}$ \(<\) get entered CPU core number
2. for $i = 0$, $k = 0$ to number of CPUs do \(<\) create list of CPUs fitting to this requirement
3. if $CPU\text{-Data}[j, 1] \geq N_{\text{core}}$ then
4. \hspace{1cm} $CPUc[j] = CPU\text{-Data}[j, 0]$; \(<\) selectable item of "drop-down" list
5. else
6. \hspace{1cm} $CPUc[j] = \"\"$; \(<\) empty (not selectable) item of "drop-down" list
7. \hspace{1cm} $k = k + 1$; \(<\) counter of empty (not selectable) items
8. if $k = \text{number of CPUs}$ then message goto 1
9. else set $j = c$; \(<\) choice of CPU with core number $N_{\text{core}}$;
10. continue \(<\) continuing with next part of modules choice algorithm;
For example, if $N_{\text{core}} = 4$

<table>
<thead>
<tr>
<th>$j$</th>
<th>CPU</th>
<th>CPU Core number</th>
<th>CPU Clock [GHz]</th>
<th>Price [BGL]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CELERON-D 347</td>
<td>1</td>
<td>3.06</td>
<td>59.50</td>
</tr>
<tr>
<td>1</td>
<td>Core DUO E5200 /800/2M BOX</td>
<td>2</td>
<td>2.50</td>
<td>109.00</td>
</tr>
<tr>
<td>2</td>
<td>Core2 DUO E7600 /1066/ 3M BOX</td>
<td>2</td>
<td>3.06</td>
<td>251.00</td>
</tr>
<tr>
<td>3</td>
<td>Core2 QUAD Q8200S/1333/ BOX</td>
<td>4</td>
<td>2.33</td>
<td>381.00</td>
</tr>
<tr>
<td>4</td>
<td>C i5-750 /8M/BOX/</td>
<td>4</td>
<td>2.66</td>
<td>371.00</td>
</tr>
<tr>
<td>5</td>
<td>C i7-860 /8M/BOX/</td>
<td>4</td>
<td>2.80</td>
<td>531.00</td>
</tr>
<tr>
<td>6</td>
<td>C i7-940 /8M/BOX/</td>
<td>4</td>
<td>2.93</td>
<td>979.50</td>
</tr>
</tbody>
</table>

If the given requirement cannot be satisfied i.e. there is no CPU with this core number, a proper message is displayed and another $N_{\text{core}}$ number could be looked for.
Other example of intelligent choice is the choice of module by price. If the DM sets maximal price $\text{Price}_{mb}$ he is willing to pay for MB, the MB choice can be realized by the following algorithm:

A5b: Algorithmic realization of intelligent MB choice

1. get $\text{Price}_{mb}$ $\leftarrow$ get entered maximal value for MB price
2. for $i = 0, k = 0$ to number of MBs do $\leftarrow$ create list of MBs fitting to this requirement
3. if $MB\text{-Data}[i,3] \leq \text{Price}_{mb}$ then
4.   $MBc[i] = MB\text{-Data}[i,0]$; $\leftarrow$ selectable item of "drop-down" list
5.   else
6.     $MBc[i] = "$\text{"}$"; $\leftarrow$ empty (not selectable) item of "drop-down" list
7.     $k = k + 1$; $\leftarrow$ counter of empty (not selectable) items
6. if $k = \text{number of MBs}$ then message goto 1
7. else set $i = m$; $\leftarrow$ choice of MB with price less or equal to $\text{Price}_{mb}$; 
8. continue $\leftarrow$ continuing with next part of modules choice algorithm;
For example, if $\text{Price}_{mb} \leq 300$

<table>
<thead>
<tr>
<th>$i$</th>
<th>MB</th>
<th>MB RAM, slots</th>
<th>MB Max RAM [GB]</th>
<th>Price BGL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ASROCK P45XE-WIFIN/P45</td>
<td>4</td>
<td>16</td>
<td>178.50</td>
</tr>
<tr>
<td>1</td>
<td>ASROCK G31M-S/G31</td>
<td>2</td>
<td>8</td>
<td>69.00</td>
</tr>
<tr>
<td>2</td>
<td>Gigabyte G31M-ES2C/G31</td>
<td>2</td>
<td>4</td>
<td>74.50</td>
</tr>
<tr>
<td>3</td>
<td>Gigabyte X48-DQ6/X48</td>
<td>4</td>
<td>8</td>
<td>227.50</td>
</tr>
<tr>
<td>4</td>
<td>ASUS P5S800-VM/SIS661FX</td>
<td>2</td>
<td>2</td>
<td>36.00</td>
</tr>
<tr>
<td>5</td>
<td>ASUS P7P55D/PRO/P55</td>
<td>4</td>
<td>16</td>
<td>315.50</td>
</tr>
<tr>
<td>6</td>
<td>INTEL DX58SO/X58/BOX</td>
<td>4</td>
<td>8</td>
<td>421.50</td>
</tr>
</tbody>
</table>

The result of execution will be a list of MBs with price less or equal to the given $\text{Price}_{mb}$ or, if all of the available MBs are more expensive, a message will be displayed and other requirement from the user will be expected.
The price is always worth to consider when designing a system. The summary price of chosen modules can be calculated by another easy to implement algorithm.

<table>
<thead>
<tr>
<th>A6: Algorithmic realization of the design cost calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. get m, c, and r \gets get indexes of chosen modules</td>
</tr>
<tr>
<td>2. [ Price_{\text{sum}} = MB-Data[m,3] + CPU-Data[c,3] + RAM-Data[r,3] ]</td>
</tr>
<tr>
<td>3. end</td>
</tr>
</tbody>
</table>

Following these approach other algorithms could be developed to reflect different requirements to designed system.
A software system is constructed to serve a **specific purpose**. In order to achieve the desired outcome, the software code needs to complete different tasks leading to the final solution. The tasks can be grouped by **logical functions**. Building web applications makes no exception and one of the important problems in software system building is designing of **graphical user interface** (GUI).

The basic framework for developing of GUI should **follow the same designer's logical functions** that were followed in the development of algorithms. On the other hand, there is a close connection between the **GUI and software modules** to be developed.

A GUI prototype for PC configuration example is shown on the next slide.
## GUI Design Description

### Main Menu
- Main menu
- Print
- Help
- About

### System Configuration
- **MB**: ASROCK P45XE-WIFIN/P45
- **CPU**: CELERON-D 347
- **RAM**: DDR A-DATA
- **RAM number**: 1

### Table: System Specifications

<table>
<thead>
<tr>
<th>MB</th>
<th>CPU</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socket</td>
<td>RAID</td>
<td>RAM Capacity</td>
</tr>
<tr>
<td>RAM type</td>
<td>RAID slots</td>
<td>RAM Frequency</td>
</tr>
<tr>
<td>RAM capacity</td>
<td>RAM frequency</td>
<td>Price</td>
</tr>
<tr>
<td>Price</td>
<td>Core Number</td>
<td>Clock Speed</td>
</tr>
<tr>
<td></td>
<td>Socket</td>
<td>Price</td>
</tr>
</tbody>
</table>

### Control Buttons
- **Calculate**
- **Reset**

### Additional Input Fields
- RAM capacity
- Modules price
The final goal of data structures and algorithms developing is to create a **software tool to assist the** DM in making intelligent choice of modules when designing a modular system. The GUI shape depends on the **specifics of the designed modular system**. The main concept of the developed GUI is using of **intelligent “drop-down” menus for choice of modules**, i.e **lists of compatible modules** to be used for choice. The proposed data structures allow to do this with minimal algorithmic difficulties.

The developed GUI was implemented in prototype of a Web-based system for PC configuration design used to test the proposed data structures and algorithms. The prototype has been developed as **client-side** application by means of HTML and JavaScript.
Conclusion

Engineering system design in **modular form** has some **advantages** as: *flexible structure, easy to maintain, debug, upgrade and customize with respect to changing customer requirements*. The automating some of designers hard work by **software tools** making some logical and reasonable choices, visualizing the results, etc., is valuable.

The current paper describes **data structures** that support developing of easy to realize and implement **algorithms** for **intelligent Web-based system for modular design**. Web-based format is chosen because it offers the advantages of **instant access, automatic upgrades, and cross-platform compatibility**. The **intelligence** of the system is focused on choice of compatible modules reflecting different design requirements. The proposed data structures and algorithms are explained and tested on the basis of **personal computer configuration case study**. Testing shows the practical applicability of the proposed data structures and algorithms.
• The research work reported in the paper is partly supported by the project **AComIn – Advanced Computing for Innovation**, grant 316087, funded by the FP7 Capacity Programme (Research Potential of Convergence Regions).