International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:7, 2013

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

Ivan C. Mustakerov and Daniela I. Borissova

***Abstract*—**In recent years, new product development became more and more competitive and globalized, and the designing phase is critical for the product success. The concept of modularity can provide the necessary foundation for organizations to design products that can respond rapidly to market needs. The paper describes data structures and algorithms of intelligent Web-based system for modular design taking into account modules compatibility relationship and given design requirements. The system intelligence is realized by developed algorithms for choice of modules reflecting all system restrictions and requirements. The proposed data structure and algorithms are illustrated by case study of personal computer configuration. The applicability of the proposed approach is tested through a prototype of Web-based system.

***Keywords*—**Data structures, algorithms, intelligent web-based system, modular design.

I. Introduction

THE growing range of products and services from all sectors of economic activity is influenced from the technological progress. Today, 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 [1].

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 different functional capabilities. The modular design facilitates the design of modular product architectures and/or the creation of modular manufacturing processes. The concept of modularity is a useful foundation to design products that respond to rapidly changing market needs and allow a cost-effective way of changes in product design. Modular design allows having both the gains of standardization and the gains of customization [2]. Examples of modular design implementation can be referenced as vehicles, computers, and buildings, just to mention a few. The purpose of any engineering system design is to realize the requirement functions and to apply the essence of intelligent decision making. Development of products in many engineering disciplines is a challenging task and how to effectively support the innovative design has become a priority consideration [3]-[5]. In many application areas customer preferences change rapidly and this requires shorter product development cycles. Again, the principle of modularization can play a key role in achieving mass customization and this is crucial in today's competitive global market environments [6], [7].

Software has become an important tool of systems design. Currently, software is undergoing a fast technological progress. Object-oriented, service-oriented and modeling languages such as Java, XML, UML and SysML [8] tools have considerably influenced the software system development and design.

Mastering the complexity of software-intensive systems requires a combined effort for foundational research and new engineering techniques that are based on mathematically well-founded theories and approaches. The new methods should support the whole system life cycle including requirements, design, implementation, maintenance, reconfiguration and adaptation.

The software systems for information processing and decision support can be viewed as combination of information technology and people activities that support operations management and decision making. The bridges between industry and computer science using the theoretical foundations of information processing and computation could be realized via different architectural models and related algorithmic processes. In a very broad sense, the term information system is frequently used to refer to the interaction between people, processes, data and technology. In this sense, the term is used to refer not only to the information and communication technology but also to the way in which people interact with this technology in support of different industry and business processes [9]. The architecture of software system is an important field of software applications development, evolution, and maintenance [10]-[12]. Software architecture models the structure and behavior of a system on a high level view of a system, including the software elements and the relationships between them. Accordingly the IEEE 1471-2000 standard [13] software architecture is the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

When talking about the intelligence in modular system

I. C. Mustakerov is with the Institute of Information and Communication Technology at the Bulgarian Academy of Sciences, Sofia – 1113, Bulgaria (phone: 3952 9793241; e-mail: mustakerov@iit.bas.bg).

D. I. Borissova is with the Institute of Information and Communication Technology at the Bulgarian Academy of Sciences, Sofia – 1113, Bulgaria (phone: 3952 9792055; e-mail: dborissova@iit.bas.bg).

design it is recognized that intelligent software should be able to assist and advise the designer during the decision-making process. The developed software modules should react in near real-time to changing problem situations, propose alternative actions, and evaluate the merits of such proposals [14]. The recent developments in computer technology have made it possible to store vast amounts of data in electronic form. In reality, 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 with an emphasis on support of intelligent design decisions making.

II. Modular Design and Data Structure

The engineering design is as much art as science and there can be a variety of possible design alternatives. Any kind of assistance of the designer by automating some of his hard work – performing complex calculations, making some logical and reasonable choices, visualizing the results, etc., would be valuable. If the design process can be modeled and simulated in computer environment it will increase the design efficiency. Even simple software tools for design activities will be helpful in decreasing of design time and costs. The modular design does not involve the development of new components but require knowledge of the modules characteristics and their compatibility relationship with other modules. The intelligent decision making for any modular design is based on proper choices of modules. This choice should correspond to all of the given functional requirements for the designed system and in most cases should meet some design criteria. For example, one of the widely used design criterion is cost of the design. If the choice of modules is irrelevant and design goals are not met the whole design process should be repeated.

Concept of modular design is very helpful in providing variety of products to the customers. A typical example of using the advantages of modularization is personal computers (PC) configuration design. There exists large diversity of PC users. Most of them can be classified in different distinguished groups as business users, students, scientists, home users, gamers, etc. Each group of users has different requirements about the PC functional capabilities. On the other hand, there always exist individual users outside these groups with their own specific requirements about the PC functionality. In that sense the PC configuration design can be investigated as a case study that illustrates many of the features of modular system 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. For illustration of the proposed in the paper data structures and algorithms the following sets of basic modules are considered:

MB = $\{MB_i,\ i = 0, 1, \dots , 6\}$
CPU = $\{CPU_i,\ i = 0, 1, \dots , 6\}$
RAM = $\{RAM_i,\ i = 0, 1, \dots , 6\}$

These modules sets are defined on the basis of some real commercially available MB, CPU and RAM modules with parameters data shown in Tables I, II and III.

TABLE I
MB-Data [*i,j*]

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

TABLE II
CPU-Data [*j,l*]

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

TABLE III
RAM-Data [*j,l*]

j	RAM	RAM Size [GB]	RAM Frequency [MHz]	Price [BGL]
0	DDR A-DATA	1	400	55.00
1	DDR2 KINGSTON	1	667	41.50
2	DDR2 KINGSTON	1	1066	64.50
3	DDR3 KINGSTON	1	1066	42.50
4	DDR2 KINGSTON	2	800	77.50
5	DDR3 A-DATA	2	1333	77.50
6	DDR3 HYPER X KINGSTON	2	1600	83.00

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. The compatibility dependences of PC modules must be strictly met in order to have a working system configuration. For example, 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.

The compatibility dependences between modules are graphically illustrated in Fig. 1.

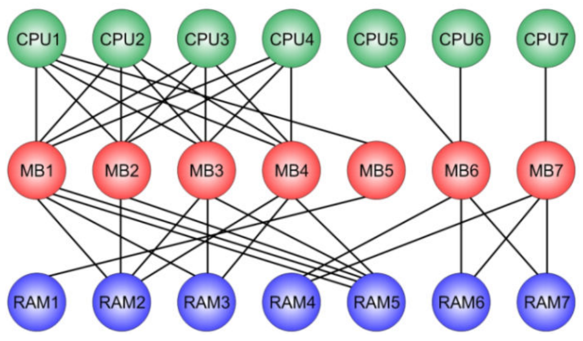


Fig. 1 Graphical illustration of PC modules relationship

When organizing data in decision support system, it is important to define proper data structures that will assist the development of efficient processing algorithms. Some experimentation with different types of data structures proved that the two-dimensional arrays, where values of 0 mean incompatible modules and nonzero values mean compatible modules are suitable for the goal. For example, compatibility dependencies between MB and CPU can be described by 2D array *MB-CPU*[*i,j*] as shown in Table IV.

TABLE IV
MB-CPU[*i,j*] – 2D ARRAY OF MB AND CPU COMPATIBILITY

	CPU_0	CPU_1	CPU_2	CPU_3	CPU_4	CPU_5	CPU_6
MB_0	1	1	1	1	0	0	0
MB_1	1	1	1	1	0	0	0
MB_2	1	1	1	1	0	0	0
MB_3	1	1	1	1	0	0	0
MB_4	1	0	0	0	0	0	0
MB_5	0	0	0	0	1	1	0
MB_6	0	0	0	0	0	0	1

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 and each RAM module has certain memory size and operating frequency (see Table III). For the cases like this, a possible solution is to use several 2D arrays i.e. three 2D arrays for MB and RAM compatibility – one for memory slots, one for max memory size and one for RAM frequency – Tables V, VI and VII.

TABLE V
MB-RAM-1[*i,j*] – 2D ARRAY OF MB RAM FREQUENCY COMPATIBILITY

	RAM_0	RAM_1	RAM_2	RAM_3	RAM_4	RAM_5	RAM_7
MB_0	0	667	1066	0	800	0	0
MB_1	0	667	0	0	800	0	0
MB_2	0	667	1066	0	800	0	0
MB_3	0	667	1066	0	800	0	0
MB_4	400	0	0	0	0	0	0
MB_5	0	0	0	1066	0	1333	1600
MB_6	0	0	0	1066	0	1333	1600

TABLE VI
MB-RAM-2[*i,j*] – 2D ARRAY OF MB RAM SLOTS NUMBER COMPATIBILITY

	RAM_0	RAM_1	RAM_2	RAM_3	RAM_4	RAM_5	RAM_7
MB_0	0	4	4	0	4	0	0
MB_1	0	2	0	0	2	0	0
MB_2	0	2	2	0	2	0	0
MB_3	0	4	4	0	4	0	0
MB_4	2	0	0	0	0	0	0
MB_5	0	0	0	4	0	4	4
MB_6	0	0	0	4	0	4	4

TABLE VII
MB-RAM-3[*i,j*] – 2D ARRAY OF MB MAXRAM SIZE COMPATIBILITY

	RAM_0	RAM_1	RAM_2	RAM_3	RAM_4	RAM_5	RAM_7
MB_0	0	16	16	0	16	0	0
MB_1	0	8	0	0	8	0	0
MB_2	0	4	4	0	4	0	0
MB_3	0	8	8	0	8	0	0
MB_4	2	0	0	0	0	0	0
MB_5	0	0	0	16	0	16	16
MB_6	0	0	0	8	0	8	8

If other modules of the designed system have to be considered similar data structures for those modules can be introduced.

III. DESIGN SPECIFIC AND ALGORITHMIC REALIZATION

Each system design has specific requirements that have to be reflected by developing of proper algorithms. This means the data structures and algorithmic realizations are closely related. One of the main questions in modular system design is decision making about the compatible modules choice. The developing of corresponding algorithm should follow as closely as possible the designer's logic for decision making. Using the described data structures for MB, CPU and RAM modules, a generalized algorithm for modular system design is shown on Fig. 2.

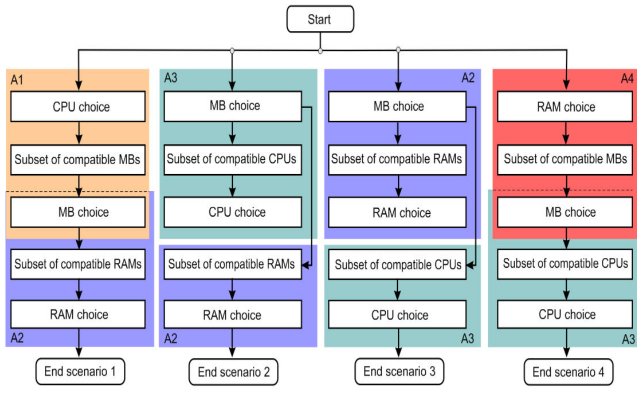


Fig. 2 Generalized algorithm for PC modular design

It describes 4 decision making scenarios distinguished as four algorithm branches. These branches differ from each other by the order of modules selection. First branch of algorithm starts with choice of CPU. The CPU choice defines a subset of compatible MBs that is used to make choice of a particular MB. The MB choice in turn defines a subset of compatible RAMs to choose from. Second branch of algorithm starts with choice of MB, goes through choice of CPU and ends with choice of RAM module. The third branch also starts with choice of MB but continues through RAM and then CPU modules choices. The fourth branch of algorithm describes the most unlikely but possible scenario of starting with RAM module choice. The RAM modules choice influences MB choice that in turn defines the CPU choice. There is no compatibility relationship between RAM and CPU modules to be considered.

The algorithmic realization of four scenarios/algorithms (A1, A2, A3, A4) shown in Fig. 2 is achieved by using of the proposed data structures as following algorithms:

A1: Algorithm realization for CPU => MB choice of compatible modules

```
1. Given the array CPU-Data[j] or CPUm[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
```

A2: Algorithm realization for MB => RAM choice of compatible modules

```
1. Given the array MB-Data[i] or MBc[i] set i = m;  ≺ choice of particular MB
2. for j = 0 to number of RAMs do  ≺ create list of compatible RAMs for "drop-down" menu
3.   if MB-RAM-1[m,j] > 0 then
4.     RAMm[j] = RAM-Data[i,0];  ≺ selectable item of "drop-down" list
5.   else RAMm[j] = "";  ≺ empty (not selectable) item of "drop-down" list
6. set j = r ;  ≺ choice of particular RAM with index r
7. get $N_{ram}$  ≺ get entered RAM modules number
8. if $N_{ram}$ > MB-RAM-2[m,j] then message goto 7
9. else if $N_{ram}$*RAM-Data[r,1] > MB-RAM-3[m,r] then message goto 7
10.    else end
```

A3: Algorithm realization for MB => CPU choice of compatible modules

```
1. Given the array MB-Data[i] or MBc[i] set i = m;  ≺ choice of particular MB
2. for j = 0 to number of CPUs do
3.   if MB-CPU[m,j] > 0 then  ≺ create list of compatible CPUs for "drop-down" menu
4.     CPUm[j] = CPU-Data[i,0];  ≺ selectable item of "drop-down" list
5.     else CPUm[j] = "";  ≺ empty (not selectable) item of "drop-down" list
6. set j = c;  ≺ choice of particular CPU with index c;
7. end
```

A4: Algorithm realization for RAM => MB choice of compatible modules

```
1. Given the array RAM-Data[j] set j = r;  ≺ choice of particular RAM,
2. for i = 0 to number of MBs do
3.   if MB-RAM-1[i,r] > 0 then  ≺ create list of compatible MBs for "drop-down" menu
4.     MBr[i] = MB-Data[i,0];  ≺ selectable item of "drop-down" list
5.   else MBr[i] = "";  ≺ empty (not selectable) item of "drop-down" list
6. set i = m;  ≺ choice of particular MB with index m
7. get $N_{ram}$  ≺ get entered RAM modules number
8. if $N_{ram}$ > MB-RAM-2[m,j] then message goto 7
9. else if $N_{ram}$*RAM-Data[r,1] >MB-RAM-3[m,r] then message goto 7
10.    else end
```

The goal of all algorithmic realizations of modules choices is to create lists of compatible modules to be used in "drop-down" menus. The proposed data structures allow this to happen with minimal algorithmic difficulties. For example, when algorithm A1 starts with selection of a concrete CPU module, 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*]. The cells corresponding to zero values are left empty as indication that these items are not selectable in "drop-down" menu.

When the choice of modules begins with MB module (A3 algorithm) a certain row of *MB-CPU*[*i,j*] is selected and the list of compatible processors is determined in the same manner by using data from the first column of the 2D array *CPU-Data*[*i,j*]. Creation of list of compatible RAMs and MBs follows the same logic but three arrays for compatibility description are used.

The array *MB-RAM-1*[*i,j*] is used for defining list of compatible RAMs (A2 algorithm) or compatible MBs (A4 algorithm) 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. If these limitations are not met proper messages are to be displayed and another number of RAM modules are expected to be entered.

These types of data structures support also algorithmic realization of intelligent choice of modules in design process. For example, if the user (DM, designer) needs a number of processor cores equal to N_{core} he could set this value in the corresponding menu position and the following algorithmic realization (algorithm A5a) has to be used:

A5a: Algorithmic realization of intelligent CPU choice

```
1. get $N_{core}$  ≺ 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-Data[j,1] $\geq N_{core}$ then
4.     CPUc[j] = CPU-Data [j,0];  ≺ selectable item of "drop-down" list
5.   else
       CPUc[j] = "";  ≺ empty (not selectable) item of "drop-down" list
       k = k +1;  ≺ counter of empty (not selectable) items
6. if k = number of CPUs then message goto 1
7. else set j = c;  ≺ choice of CPU with core number $N_{core}$;
8. continue  ≺ continuing with next part of modules choice algorithm;
```

If the given requirement cannot be satisfied i.e. there is no CPU with core number N_{core}, a proper message is displayed and another N_{core} could be entered.

Other example of intelligent choice is the choice of module by price. If the user (DM, designer) sets maximal price $Price_{mb}$ he is willing to pay for MB, the choice of proper MB is realized by following the algorithmic realization (A5b) as:

A5b: Algorithmic realization of intelligent CPU choice

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

The result of execution will be a list of MBs with price less or equal to the given $Price_{mb}$ 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 summary price of chosen modules is calculated by another easy to implement algorithm.

A6: Algorithmic realization of the design cost calculation

```
1.  get m, c, and r ≺ get indexes of chosen modules
2.  $Price_{sum} = MB\text{-}Data[m,3] + CPU\text{-}Data[c,3] + RAM\text{-}Data[r,3]$
3.  end
```

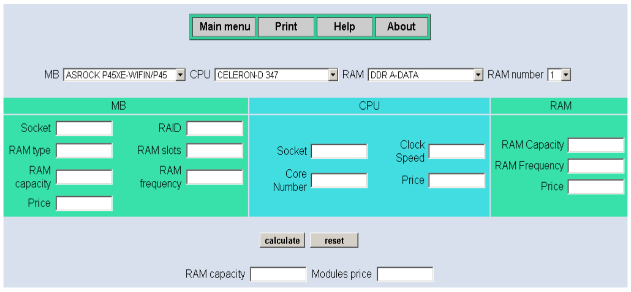
Following these algorithms reflecting different requirements for the designed system, a proper software system could be realized.

IV. GUI Design Description

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 there have been many advances in this area.

One of the important problems in software system building is designing of graphical user interface (GUI). The goal is to build a framework for a simple and intuitive GUI supporting the application of the processing algorithms and user acceptance of the system. The basic framework for developing of GUI should follow the same designer's logical functions that were followed in the development of algorithms. This means that there is also a close connection between the GUI and software modules to be developed.

The GUI shape depends on the specifics of the designed modular system. An intuitive and easy for interacting GUI for PC configuration, incorporating the developed above algorithms (A1, A2, A3, A4, A5a, A5b and A6) as a case study, is shown on Fig. 3.



MB ASROCK P45XE-WIFIN/P45 CPU CELERON-D 347 RAM DDR A-DATA RAM number 1

Fig. 3 GUI Screenshot

The main concept of GUI development is using of "drop-down" menus for choice of modules. At the start of operation of the system the lists of "drop-down" menus are initialized with all available modules. When a specific module is selected the lists of modules for other "drop-down" menus are updated to contain only compatible modules with the selected one. This system feature is reflection of implemented intelligence of the system. After choice of particular module the module's parameters are loaded and displayed in corresponding text areas. If there is some criterion for the parameters of the designed system, its value is calculated taking into account the parameters of chosen modules. For PC configuration case study the button "calculate" activates computing of total size of RAM and overall price of modules. Another intelligent option of the system is possibility to edit the text fields containing the modules' parameters. In other words, the user can enter some preferable value for module's parameter and after pressing button "calculate" the system will try to find the module that satisfies this requirement of the user. If this is not possible an appropriate message is displayed and user should take other decision about that parameter. This type of intelligence can also include the possibility to propose module with closest value to preferred one for this parameter. Other intelligent feature would be implementation of optimization according to given criterion or multiple criteria.

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. Testing of the prototype for PC configuration design using real data modules from Tables I and II, and Table III shows the practical applicability of the proposed data structures and algorithms. The concept of a modular realization of the algorithms supports the use of a modular architecture of the software system. The including of different processing software modules depends on user needs and the modular architecture allows easy carrying out of extensions and modifications to the system.

V. Conclusion

Engineering system design in modular form means designing a system based on modules. The advantages of modular design are flexible structure, easy to maintain, debug, upgrade and customize with respect to changing customer requirements. The automating some of designers hard work by 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 on the basis of personal computer configuration case study. The testing of a prototype of a Web-based system for PC configuration design using real modules data showed practical applicability of the proposed data structures and algorithms.

The described data structures and algorithms and the use of XML and related technologies will be the basis for developing a framework of software system for the design of modular engineering systems.

Acknowledgment

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).

References

[1] M. S. Sa'Ed, A. K. Kamrani. Modular Design. *Collaborative Engineering*, pp. 207-226, 2008.

[2] J. T. Dorsey, T. J. Collins, W. R. Doggett, R. V. Moe. Framework for Defining and Assessing Benefits of a Modular Assembly Design Approach For Exploration Systems. In: *Space Technology and Applications International Forum*, Albuquerque NM, 12-16 February 2006. AIP Conference Proceedings, vol. 813, Ed. Mohamed S. El-Genk.

[3] S. Gonnet, G. Henning, H. Leone. A model for capturing and representing the engineering design process. *Expert Systems with Applications,* no 33, pp. 881-902, 2007.

[4] K. Fujita. Product variety optimization under modular architecture. *Computer-Aided Design*, no 34, pp. 953-965, 2002.

[5] A. L. Chen, D. H. Martinez. A heuristic method based on genetic algorithm for the baseline-product design. *Expert Systems with Applications* no 39, pp. 5829-5837, 2012.

[6] W. Li, Y. Li, J. Wang, X. Liu. The process model to aid innovation of products conceptual design. *Expert Systems with Applications*, no 37, pp. 3574-3587, 2010.

[7] J. Yoo, S. R. T. Kumara. Implications of k-best modular product design solutions to global manufacturing. *CIRP Annals – Manufacturing Technology*, no 59, pp. 481-484, 2010.

[8] M. Hause, F. Thom. Building Bridges Between Systems and Software with SysML and UML. In: *INCOSE Intl. Symposium*, 2008.

[9] D. M. Kroenke. *Experiencing MIS*. Prentice-Hall, Upper Saddle River, NJ 2008.

[10] L. Bass, P. Clements, R. Kazman. *Software Architecture in Practice*. 2nd edition. Reading, MA: Addison-Wesley, 2003.

[11] G. Booch. The irrelevance of architecture. *IEEE Software*, no 24, pp.10-11, 2007.

[12] R. N. Taylor, N. Medvidovic, E. Dashofy. *Software Architecture: Foundations, Theory, and Practice*. New York, NY: Wiley Publishing, 2009.

[13] Software Engineering Standards Committee of the IEEE Computer Society, IEEE Recommended practice for architecture description of software-intensive systems, IEEE Std 1471-2000, Approved 21 September 2000, IEEESA Standards Board, Print: ISBN 0-7381-2518-0 SH94869, PDF: ISBN 0-7381-2519-9 SS94869, at (http://standards.ieee.org/).

[14] Decision Support Systems in Agent-Based Intelligent Environments. Knowledge-Based Intelligent Engineering Systems Series. Phillips-Wren G. and L. Jain (eds.), IOS Press, Amsterdam, The Netherlands, 2005.