Optimum Criteria for Developing Defined Structures

Ion IVAN, Eugen DUMITRAŞCU, Daniel MILODIN, Dragoş PALAGHIŢĂ
ionivan@ase.ro, eugen.dumitrascu@cs.ucv.ro, daniel.milodin@ase.ro, dpalaghita@gmail.com
Academy of Economic Studies, Bucharest

Basic aspects concerning distributed applications are presented: definition, particularities and importance. For distributed applications linear, arborescent, graph structures are defined with different versions and aggregation methods. Distributed applications have associated structures which through their characteristics influence the costs of the stages in the development cycle and the exploitation costs transferred to each user. The complexity of the defined structures is analyzed. The minimum and maximum criteria are enumerated for optimizing distributed application structures.
Keywords: *data structures, distributed applications, aggregation.*

1 Distributed applications

A distributed application or a global application is an application which includes access to data from several nodes of a computer network. The components are executed on different nodes, on different platforms which are connected to the network.
Distributed applications are those applications in which several beneficiaries or users in various locations in the territory, access resources defined for a computer network with a view to finding a solution to an issue. Modern perceptions of banking transactions, inter-bank transactions, electronic commerce activities, training, informing, knowledge evaluation activities, conclusion of contracts on-line, are only a few of the distributed application that should be features of the information society. E-learning, e-government, e-business, virtual organizations and new work types implementation development philosophies are all based on the principles of distributed applications. [IVAN06].
Each user has his own set of information which he has access to. There is information which all users have access to, through a username and a password.
The distinctive features of a distributed application are:

- powerful interfaces with can be used by a various number of people;
- an increased degree of generality which enables a very large number of people to solve their problems
- user-friendly interfaces which would enable the elimination of errors when data is filled out and of discontinued use
- security level which would ensure that the transaction system is functional;
- access levels which would provide a satisfactory solution to security and transparency issues
- an increased level of accuracy and reliability;
- it would ensure the recording of sufficient information to enable the retracing information tracks;
- the components of any distributed application contain at least two main parts: the application part and the communication part; some components also include a special part called administrative part which is a means to control and monitor the components;
- a high degree of modularity and the possibility of extension by adding or removing software or hardware components;
- secure operation.

There are three aspects to the term distributed application:

- application A, whose functionality is divided into *n* components A_1, A_2, …, A_n, $n \in N$, $n>1$ which interact and cooperate with each other; each component is an application or a process;
- A_i components are independent entities which run on different computers;
- A_i components change information

through the network.
A distributed application includes the $PROC_1$, $PROC_2$, … $PROC_{pr}$ procedures. Each procedure is characterized by entry points and exit points. The procedure is requested from one or more locations. The procedure with more exits points has more *return* instructions.
There are programming languages which define more entry points – *entry* - for procedures. All situations in which a procedure with the entry points, exit points and calls between procedures occur are presented.
A procedure with *w* parameters has the following format $PROC(p_1, p_2, p_3, \ldots, p_w)$.
The figures below indicate various methods for procedure calls.

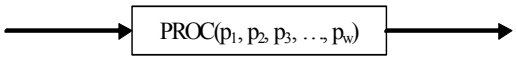

Fig.1. Procedure with one entry point and one exit point

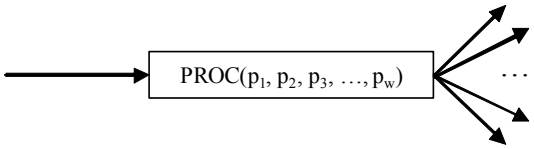

Fig.2. Procedure with one entry point and several exit points

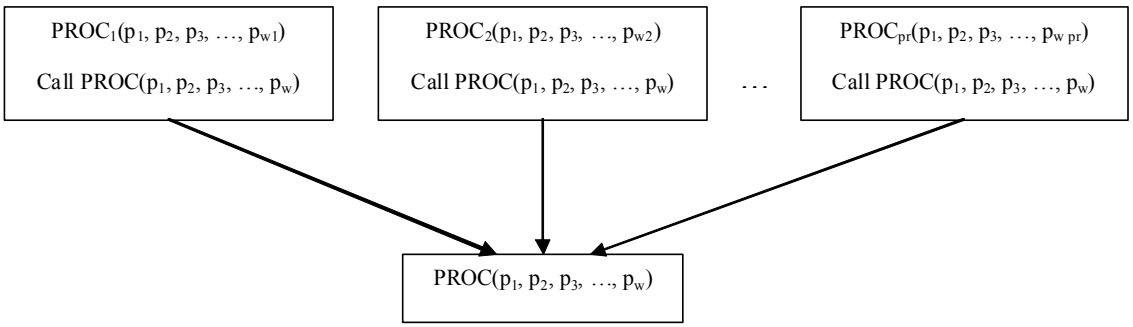

Fig.3. Procedures which calls the same procedure

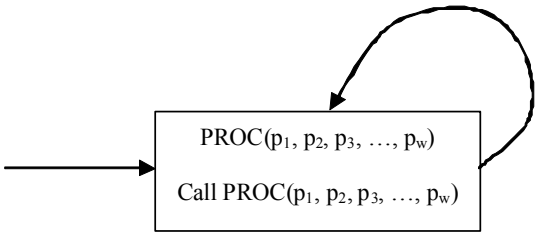

Fig.4. Procedure with one entry point and self-call – recursive procedure

Starting from the ideas of procedure and call, procedures are nodes in a graph and calls generate arcs between the nodes. Considering two procedures $PROC_i$ and $PROC_j$, the structure of the call graph of the $PROC_j$ procedure by $PROC_i$ is shown in figure 5.

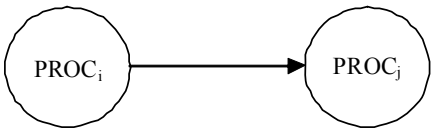

Fig.5. Call of procedure $PROC_j$ by procedure $PROC_i$

Each structure has the following features:
- homogeneity – all structures have the same number of incident arcs to the inside and the same number of incident arcs on the outside;
- node complexity – each node has his own complexity;
- hierarchy – organization of the structure on levels;
- flows between the nodes are characterized by intensity or reference frequency

The quality features of the structure are:
- accessibility – all nodes are accessed by means of an access procedure which complies with a certain rule, in the same way you access an arborescent structure: in disorder: left – root – right, in pre-order: root – left – right, in post-order: left – right – root;
- testability – the ability of observing the behavior of the element in the structure on different entries;
- interchangeability – the ability to replace a node with another node which has at least an equivalent performance to the replaced node;
- finitude – the number of references in an accepted closed interval, limitation of the du-

ration of the execution;
- correctness – the ability to arrive at the expected results for the specified entry data;
- generality – the ability to process any combination of entry data of an interval. If, n $\in$ [10, 20] and $x_i \in$ [1, 100], the average calculation program must provide accurate results for series of no matter how many numbers between 10 and 20 and for any values between 1 and 100;
- determinism – the same results are arrived at, each time the same entry data is processes;
- stability – the structure's ability to react: low entry variations result in low exit variations are obtained, high entry variations result in high exit variations; the ability of the structure to maintain elements on the same levels and to maintain the connections between the nodes through: structural stability – the position of the elements and of the connections, and behavior stability – low entry, low exit;
- accuracy – focuses on receiving no errors or on keeping them within acceptable limits set out by the quality standards;
- consistency – the extent to which entry or exit data comply with a series of conditions pertaining to the absence of variation and contradiction;
- correlation – which characterizes entry or exit data which can be compared to other similar data;
- relevancy – the nodes are kept in a condition which enables the user to always find the necessary information;
- opportunity – the ability of the nodes as applications to meets the requirements of the user;
- validity – the ability of nodes applications to generate the desired results.

2. Defined structures

The defined structure corresponds to the concept according to which the structure of an application does not change once it was set. The structure assumes interdependent components and elements.
The defined structure types are:
- linear structures;
- tree structures;
- graph structures.

The defined structure streams are:
- un-oriented ;
- one-dimensional oriented;
- two-dimensional oriented.

A *linear structure* with *n* nodes X_1, X_2, …X_n, is defined like in figure 6. Each node X_i is connected with an oriented arc to node X_{i+1}, where X_i represents procedures or modules of the distributed linear structured application. The procedures or modules are called in series or cascade. In case a procedure or module doesn't work the whole application becomes nonfunctional.

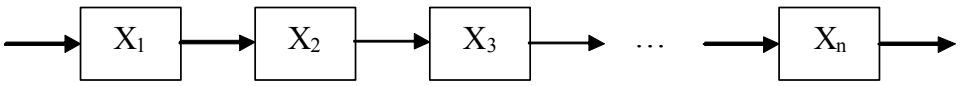

Fig.6. Linear defined structure

An *arborescent structure* with *n* nodes X_1, X_2, …X_n, like in figure 7 is defined. Each node X_i with i>1 has a parent and it is connected with an oriented arc to node X_j, on the inferior level. The procedures or modules are called in hierarchical order. In case a procedure or module does not work then all its descendent procedures will not work.
A *graph structure* with *n* nodes X_1, X_2, …X_n, like in figure 8 is defined. Each node X_i is connected with an oriented arc to any structure node X_j. In case a procedure or module doesn't work the application continues to work if the node that failed to work does not determine the isolation of other nodes, meaning that the paths between nodes remain valid.

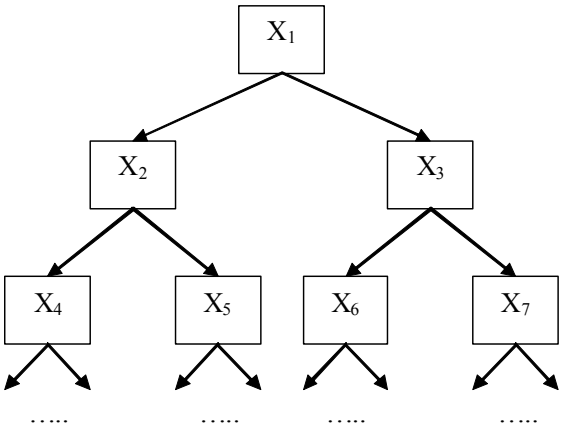

Fig.7. Arborscent defined structure.

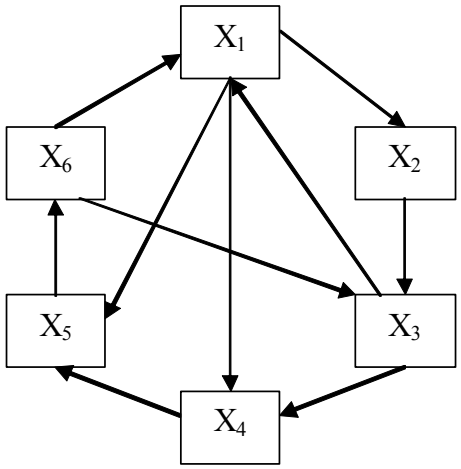

Fig.8. Graph defined structure

Defining structures for distributed applications means:

- identifying the modules;
- defining the relationships between modules;
- defining the structure type;
- defining operations between nodes.

The structure is considered defined when:

- all processing functions have been defined;
- all the requirements have been placed in correspondence with the modules;
- all the variables have been specified;
- all the modules have been built;
- all the algorithms have been established.

Redefining distributed application structures means:

- adding new nodes;
- eliminating nodes;
- changing node position from level *k* to level *k+1* or from level *k* to level *k-1*;
- modifying the content of a node;
- changing the node position in the level.

The probability to modify the structure in the development cycle is under a level that ensures the stability of the structure. The defined structure is a structure with a low modification probability.

Premises must be created in order to develop well defined structures, meaning those structures that don't suffer modifications while the development cycle unfolds. The comebacks to previous stages are only because of incorrect or incomplete operations which reduce the quality of realized components.

The well defined structure generates detailed specifications and comebacks appear because the differences between the growth of detailed specifications and the way in which the stage in question is realized. A comeback means working on a module. If the structure is not defined and it is an ordinary construction, the work is done based on the specifications and on the module with high expenses.

The stability of the structure is demonstrated using an arborescent structure. The complexity of the structure is computed based in the number *NRNp* of weighted nodes and the number *NRAp* of weighted arcs. For a binary arborescent structure the number of nodes and arcs is computed based on the number of levels, *niv,* of the tree.

The number of nodes on a level *k* is 2^{k-1}. The weight is given by the level on which the node is placed, *k.* So the next formula is deduced used for computing the number of weighted nodes of a binary tree:

$$NRNp = \sum_{k=1}^{niv} k \cdot 2^{k-1}$$

The number of incident arcs in the nodes on level *k* is given by the number of nodes on that level, the only exception being level 1 where there is no incident side. The formula used to compute the number of weighted arcs is deduced:

$$NRAp = \sum_{k=2}^{niv} k \cdot 2^{k-1}$$

The Mc Cabe complexity for a binary weighted arborescent structure is:

$$C=NRAp - NRNp + 2= \sum_{k=2}^{niv} k \cdot 2^{k-1} - \sum_{k=1}^{niv} k \cdot 2^{k-1} + 2 = 1$$

where:
NRNp – number of weighted nodes;
NRAp – number of weighted arcs;
niv – number of levels.
It is observed that for a binary weighted arborescent structure which has a maximum number of nodes on each level the complexity is constant and equal to 1.
A binary arborescent structure with 3 levels like in figure 9 is considered.

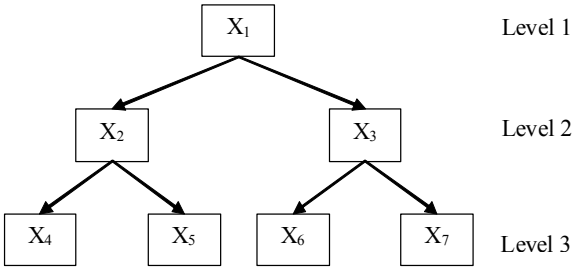

Fig.9. Arborescent structure with 3 levels.

For this structure:
$NRNp = 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 4 = 17$
$NRAp = 2 \cdot 2 + 3 \cdot 4 = 16$
C = NRAp-NRNp+2=16-17+2=1
If in this structure another node is added the number of levels is increased by 1, but the structure is not complete, meaning it does not have a maximum number of nodes on level 4, and so:
$NRNp = 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 4 + 4 \cdot 1 = 21$
$NRAp = 2 \cdot 2 + 3 \cdot 4 + 4 \cdot 1 = 20$
C = NRAp-NRNp+2=20-21+2=1

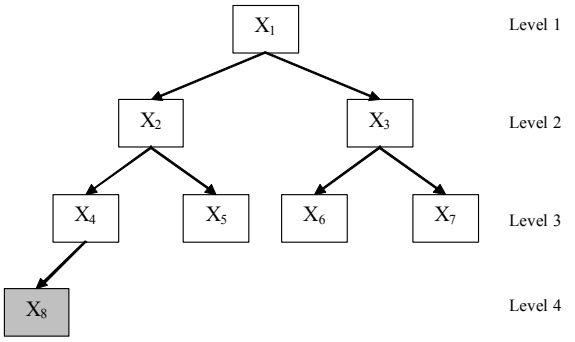

Fig.10. Adding a new node in a binary arborescent structure with 3 levels.

It is observed that in this case the complexity is still 1.
This complexity remains constant only if the structure is homogenous and the maximum number of descendents of each node is the same.
The stability of distributed applications defined structures is a very important characteristic. The distributed technology becomes day by day more robust but still immature, because of increasingly focused research on optimal technologies, secure and which make applications easier to implement. As a consequence it is necessary to know the weak points of technologies to correct problems in order to obtain stable structures.
The application simplifying process is meant to find the redundant methods, to account the reusable components, to pick the best algorithms used to find solutions to the problem that needs solving, establishing patterns for interfaces and processes, identifying the dynamical data stream.
The main objective is to establish the defined structure through allocated processes to solving the problems. Often, memory or resource exceeding undermine stability and viability of the defined structure. The stability of the algorithm that forms the base of the applications in the structure must be validated in the same time.
The stability characteristic assumes:

- a correctly defined structure;
- a homogenous structure;
- a low complexity structure, or constant even if the structure is modified;
- a viable and robust structure;
- a structure in which the modules have a high degree of security;
- a defect tolerant structure;
- a structure easy to maintain in time an to which future modifications are very small or inexistent;
- a portable structure.

The methods to increase the stability of a defined structure are:

- ensuring the homogeneity of the connections between modules;
- giving nodes the same specific quality characteristic levels will result in a good aggregated product;
- the creation of structure versions and selecting the one that corresponds to a criterion;
- testing the structure behavior and improving it;
- the complete defining of the problem and ensuring a high generality degree including all situations such that new elements will not appear in the problem;
- the computation of complexity for the structures and selecting the structure with the lowest complexity;
- aggregating structures that are stable to obtain a stable structure.

The optimization of defined distributed application structures represents a debating problem in the software field, by minimizing or maximizing several criteria which form the base of optimization.

3. The minimization problem

The minimization problem corresponds to the situation in which the performance criterion leads to selecting the solution which has the smallest criterion value.

An optimization process influenced by the F_1, F_2, ...F_n factors dynamics is considered.

The optimum criterion pursue the implementation cost resulting variable level minimization, the transaction duration minimization, or the simultaneous user number resulting variable, maximizing the information security level.

The analytical optimum criterion form is:

$$\begin{Bmatrix}\min\\ \max\end{Bmatrix} f(x_1, x_2, ..., x_n)$$

where:

n – number of independent variables included in the optimization model;

x_i – the independent variable that corresponds to a factor F_i which influences the optimization process.

The value of n depends on:

- the capacity to measure the levels of the x_i variables associated to F_i factors;
- the importance given to the variables, knowing that numerous independent variables influence insignificantly the optimization process, fact for which are not included in models.

The analytical forms associated to optimum criterion are:

- ***the linear form***

$$\begin{Bmatrix}\min\\ \max\end{Bmatrix} \sum_{i=1}^{n} a_i \cdot x_i$$

where:

a_i – real importance coefficient with $a_i \in [0, 1]$ and $\sum_{i=1}^{n} a_i = 1$;

n – number of variables considered significant in the optimization process;

x_i – the measured level associated to the F_i influence factor.

- ***the quadratic form***

$$\begin{Bmatrix}\min\\ \max\end{Bmatrix} \sum_{i=1}^{n} a_i \cdot x_i^2$$

where:

a_i – real importance coefficient with $a_i \in [0, 1]$ and $\sum_{i=1}^{n} a_i = 1$;

n – number of variables considered significant in the optimization process;

x_i – the measured level associated to the F_i influence factor.

- ***the polynomial form***

$$\begin{Bmatrix}\min\\ \max\end{Bmatrix} \sum_{i=0}^{n} a_i \cdot x_i^i$$

where:

a_i – real importance coefficient with $a_i \in [0, 1]$ and $\sum_{i=1}^{n} a_i = 1$;

n – number of variables considered significant in the optimization process;

x_i – the measured level associated to the F_i influence factor.

The models associated with distributed applications must consider:

- k number of users;
- the x_{ij} level associated to factor F_i recorded for user j.

The analytical forms associated to optimum criterion for distributed applications are:

- ***the linear form***

$$\begin{Bmatrix}\min\\ \max\end{Bmatrix}\sum_{i=1}^{n}\sum_{j=1}^{k} a_{ij}\cdot x_{ij}$$

where:
n – number of variables considered significant in the optimization process;
k – user number;
a_{ij} – real importance coefficient for user j with $a_{ij}\in[0, 1]$ and $\sum_{j=1}^{k} a_j = 1$;
x_{ij} – measured level associated to the F_i registered for user j.

- ***the quadratic form***

$$\begin{Bmatrix}\min\\ \max\end{Bmatrix}\sum_{i=1}^{n}\sum_{j=1}^{k} a_{ij}\cdot x_{ij}^2$$, where:

n – number of variables considered significant in the optimization process;
k – user number;
a_{ij} – real importance coefficient with $a_{ij}\in[0, 1]$ and $\sum_{j=1}^{k} a_j = 1$;
x_{ij} – measured level associated to the F_i registered for user j.

- ***the polynomial form***

$$\begin{Bmatrix}\min\\ \max\end{Bmatrix}\sum_{i=0}^{n}\sum_{j=1}^{k} a_{ij}\cdot x_{ij}^{i}$$, where:

n – number of variables considered significant in the optimization process;
k – user number;
a_{ij} – real importance coefficient with $a_{ij}\in[0, 1]$ and $\sum_{j=1}^{k} a_j = 1$;
x_{ij} – measured level associated to the F_i registered for user j.

From one period to another depending on the pursued objectives the criterion is changing because of this it is needed to restart the optimization process.
Considering the initial structure Str_0 and the Str_i structure which results after the modifications required in stage E_i of the development cycle of the distributed application.
If:
niv – the number of levels in the structure;
$Comp_i$ – the number of modules on level i;
Sup_i – the number of components that move from level i to the upper one;
Inf_i – the number of components that move from level i to the inferior one,
The linear optimum criterion for a defined structure which includes these variables is:

$$\{\min\}\, a_1\cdot niv + a_2\cdot\sum_{i=1}^{niv} Comp_i + a_3\cdot\sum_{i=1}^{niv} Sup_i + a_4\cdot\sum_{i=1}^{niv} Inf_i$$

The non linear optimum criterion for a defined structure which includes these variables is:

$$\{\min\}\, a_1\cdot niv + a_2\cdot\sum_{i=1}^{niv}(Comp_i - Sup_i)^2 + a_3\cdot\sum_{i=1}^{niv}(Comp_i - Inf_i)^2$$

For structure Str_0 in figure 11, where niv = 3
$Comp_i = \{1, 2, 6\}$, with $i=\overline{1..3}$
After the modification the system becomes like in figure 12, where niv = 5, $Comp_i = \{1,1,3,1,3\}$, with $i=\overline{1..5}$.
The models are calculated:

- linear optimum for a defined structure which includes these variables is:

$$\{\min\}\, a_1\cdot niv + a_2\cdot\sum_{i=1}^{niv} Comp_i + a_3\cdot\sum_{i=1}^{niv} Sup_i + a_4\cdot\sum_{i=1}^{niv} Inf_i = 5\cdot a_1 + a_2\cdot(1+1+3+1+3) +$$
$$+ a_3\cdot(0+0+0+0+0) + a_4\cdot(0+1+3+0+0) = 5\cdot a_1 + 9\cdot a_2 + 4\cdot a_4$$

- nonlinear optimum for a defined structure which includes these variables is:

$$\{\min\}\, a_1\cdot niv + a_2\cdot\sum_{i=1}^{niv}(Comp_i - Sup_i)^2 + a_3\cdot\sum_{i=1}^{niv}(Comp_i - Inf_i)^2 =$$
$$= 5\cdot a_1 + a_2\cdot[(1-0)^2 + (1-0)^2 + (3-0)^2 + (1-0)^2 + (3-0)^2] +$$
$$+ a_3\cdot[(1-0)^2 + (1-1)^2 + (3-3)^2 + (1-0)^2 + (3-0)^2] = 5\cdot a_1 + 21\cdot a_2 + 11\cdot a_3$$

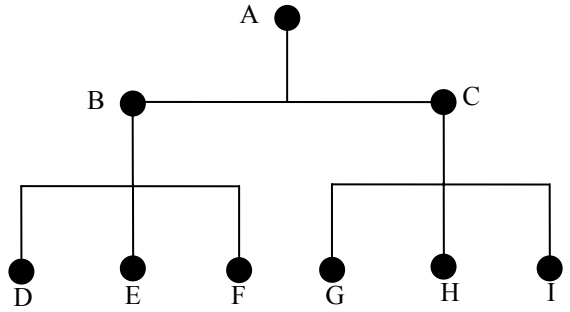

Fig.11. Arborescent defined structure

These models give the same importance to the moving process indifferently of the level on which it is executed.

In reality the movement from an inferior level to a superior one or the other way around requires programming efforts, supplementary testing and implementation costs. This is why it is necessary to make a differentiation using coefficients.

The optimum linear criterion for a defined structure includes these variables with the importance level movement differentiation is:

$$\{\min\}\, a \cdot niv + \sum_{i=1}^{niv} b_i \cdot Comp_i + \sum_{i=1}^{niv} c_i \cdot Sup_i + \sum_{i=1}^{niv} d_i \cdot Inf_i$$

The optimum nonlinear criterion for a defined structure includes these variables with the importance level movement differentiation is:

$$\{\min\}\, a \cdot niv + \sum_{i=1}^{niv} b_i \cdot (Comp_i - Sup_i)^2 + \sum_{i=1}^{niv} c_i \cdot (Comp_i - Inf_i)^2$$

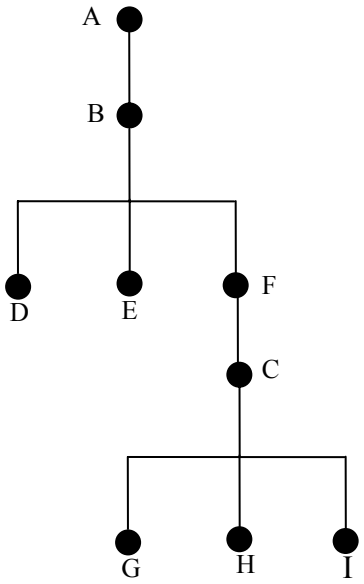

Fig.12. Arborescent modified defined structure

The optimum criteria for minimizing distributed application structures are:

- the minimization of program development costs;
- the minimization of transaction duration for applications which realize transactions on databases;
- the minimization of the access duration representing the minimization of web pages loading time in the client browser;
- the minimization of storage place occupied by the application;
- the minimization of host information storage space level, by file archiving, like files with documents, by transforming Word or PDF into HTML files with reduced size, by transforming files that contain images from a large dimension format in a format that takes up a smaller amount of storage space on the disk;
- the minimization of the traveled path in order to arrive at the requested resource, by rearranging the page structure and the tree links or graph of the web application; the length of the path as number of selections must be minimal.

A minimization process of the traversed path in a web application to arrive at the desired resource is presented, such that the number of selections in this type of tree structured application will be kept to a minimum.

A web application is considered to be used for presenting and commercializing IT products. The information page structure and the links between them given by the traveled path are reproduced in figure 13. For each node the accessing frequency is written down for the specific information contained within it.

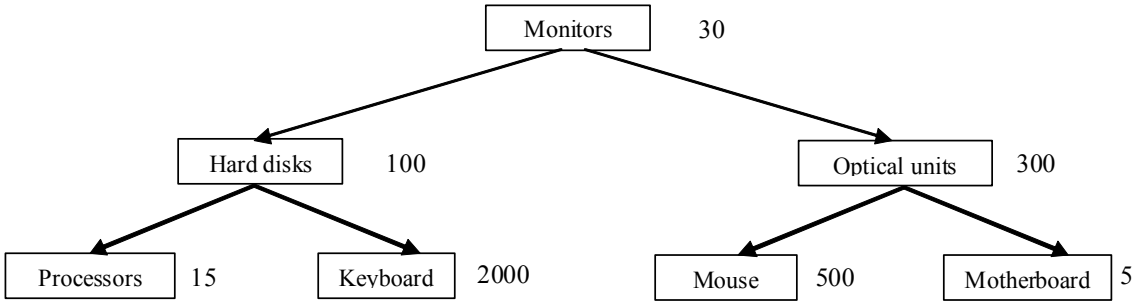

Fig.13. The initial structure of the web application

The average length of the path from the root to the leaf nodes is given by the relation:

$$\overline{L} = \frac{\sum_{i=1}^{NRO} frecv_i \cdot nivel_i}{\sum_{i=1}^{NRO} frecv_i}$$

where:

NRO – number of tree nodes attached to the application;

$frecv_i$ – the access frequency for node i;

$nivel_i$ – the level on which node i is placed.

For the given application structure the average length of the path is:

$$\overline{L} = \frac{30 \cdot 1 + 100 \cdot 2 + 300 \cdot 2 + 15 \cdot 3 + 2000 \cdot 3 + 500 \cdot 3 + 5 \cdot 3}{30 + 100 + 300 + 15 + 2000 + 500 + 5} = \frac{8390}{2950} = 2.844$$

Rearranging the pages in the structure is made considering the frequency, the pages with higher frequency will be paced closer to the root and the ones with lower access frequency will be placed closer to the leaf nodes.

After the rearrangement is completed the application structure is presented in figure 14:

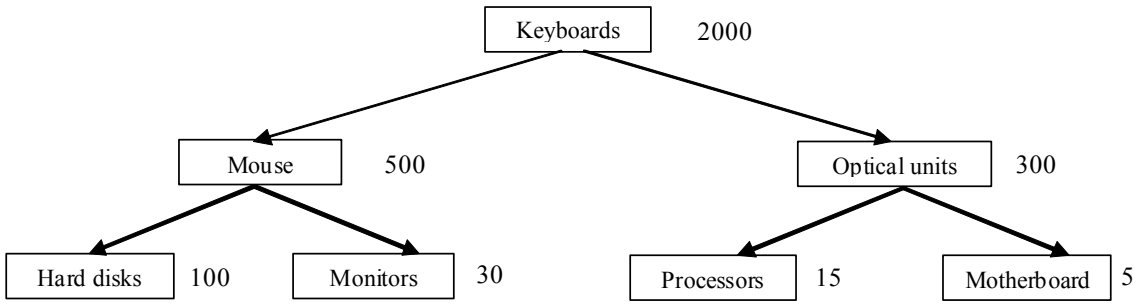

Fig.14. The new structure of the web application

The average path length for the new structure is:

$$\overline{L} = \frac{2000 \cdot 1 + 500 \cdot 2 + 300 \cdot 2 + 100 \cdot 3 + 30 \cdot 3 + 15 \cdot 3 + 5 \cdot 3}{30 + 100 + 300 + 15 + 2000 + 500 + 5} = \frac{4050}{2950} = 1.373$$

It is observed the average length of the path for the rearranged structure is smaller.

The files storage space required on the host or application server is given by the formula:

$$L(FI) = \sum_{i=1}^{NRF} L(FI_i)$$

where:

NRF – number of files present on the server;

FI_i – file i;

FI – the set of files on the host $FI = \bigcup_{i=1}^{NRF} FI_i$;

$L(FI_i)$ – the length or size of file FI_i;

L(FI) – the total length or size of the files.

In order to minimize the total file storage space it is proceeded to the transformation of files to other formats or archiving existing documents. In this manner .doc or .xsl files

are compressed using .zip or .rar archiving formats, .pdf documents are transformed in .html pages. Image files with high resolution are transformed din files with smaller resolutions in the order of kilo bytes.
So, the total space decreases and the size of the set of transformed files FI' is a lot smaller then the initial space taken by the untransformed set of files FI:

$$L(FI') << L(FI)$$

The optimization criteria presented for minimization are applied at developer level but at host level as well.
Developer level optimization is a stage which must be processed after the application implementation stage or even in the implementation stage with regard to the experience of the programmer or application developer. In application development teams there are experienced individuals in programming and code optimization, they are responsible for suggesting optimization methods or even optimized reimplementation of application modules.

4. The maximization problem

The maximum problem corresponds to the situation in which the performance criterion leads to the selection from the possible solutions of the one with the highest value.
The optimum criteria for maximizing the distributed application structures are:

- the maximization of the degree of satisfaction of the user by receiving the requested information from the application;
- the maximization of the application modules degree of reuse;
- the maximization of the diversity of client request coverage;
- the maximization of application stability.

The maximization of user satisfaction degree is given by the report:

$$GS = \frac{NRUS}{NRUT}$$

where:
NRUS – the number of users who had successfully finished an operation or an application transition;
NRUT – total number of users.

For an e-commerce application there are users who successfully find the desired product and buy it online, there are users that leave the application without finishing their tasks or before even starting to use the application. In these cases the degree of user satisfaction is computed with the formula:

$$GS = \frac{\sum_{i=1}^{NRUS} AS_i}{NRUII + NRUIP + NRUS}$$

where:
AS_i – the number of actions completed successfully by user i;
NRUII – the number of users that quit the application at the beginning;
NRUIP – the number of users that quit the applications while it is ongoing;
NRUS – the number of users that successfully finish their tasks and then leave the application;

The web application structure optimization to increase the degree of user satisfaction is realized by reducing the number of levels in a tree like structured application.
This is done by counting the nodes from the root to the leaves, more precisely the accessing frequencies of the pages in the application.
This optimization method by maximizing the degree of user satisfaction is reduced to minimizing the length of the path followed in the application, like the web application with a tree like structure. This method was presented in the minimization problem.

5. Aggregated optimization criteria for defined structures

To posses a complete image of a defined structure belonging to a distributed application, it is needed to develop an aggregation process.
For *ncrit* optimum criteria denoted by K_1, K_2, ... K_{ncrit}, building KA aggregated criterion is done by:

- simple sum of the criteria:

$$KA = \sum_{i=1}^{ncrit} K_i$$

- the weighted average of the criteria, where each criterion has importance coefficients H_i, with $H_i \in [0, 1]$ and $\sum_{i=1}^{ncrit} H_i = 1$:

$$KA = \sum_{i=1}^{ncrit} H_i \cdot K_i$$

- simple arithmetical average of the criteria:

$$KA = \frac{\sum_{i=1}^{ncrit} K_i}{ncrit}, \text{ with } ncrit \neq 0$$

- arithmetic average of the weighted criteria:

$$KA = \frac{\sum_{i=1}^{ncrit} H_i \cdot K_i}{ncrit}, \text{ with } ncrit \neq 0$$

- simple geometrical average of the criteria:

$$KA = \sqrt[ncrit]{\prod_{i=1}^{ncrit} K_i}, \text{ with } \prod_{i=1}^{ncrit} K_i > 0$$

- weighted geometrical average:

$$KA = \sqrt[ncrit]{\prod_{i=1}^{ncrit} K_i^{H_i}}, \text{ with } \prod_{i=1}^{ncrit} K_i^{H_i} > 0$$

The distributed applications assume developers, users and hosts. For each of the individuals the defined structure criteria is different in the optimization process. The developer criterion, the user criterion, and the host criterion are considered. The global optimization idea converges to aggregating the three criteria.

It is observed that the objective function associated with the optimization process oversees obtaining a structure that corresponds to specific requirements:

- the developer: programming effort, reutilization degree, costs;
- the user: transaction duration, use cost, maintenance costs, operation costs;
- the host: the server storing resource, channel occupation degree (access frequency function).

By consulting the specialists' importance coefficients are obtained for the three criteria H_1 – the developer criterion, H_2 – the user criterion, H_3 – the host criterion.

The aggregated criterion *KAGR* is given by the relation:

$$KAGR = H_1 \cdot KE + H_2 \cdot KU + H_3 \cdot KG$$

where:

KE – the analytical expression for the developer criterion;

KU – the analytical expression for the user criterion;

KG – the analytical expression for the host criterion.

The multi-criterion optimization algorithm implies:

- defining the initial solution
- evaluating the three criteria;
- evaluating the global criterion;
- making modifications at structure level according to the requirements of the developer;
- making modifications at structure level according to the requirements of the user;
- making modifications at structure level according to the requirements of the host;
- guaranteeing the process finality by making limited modifications to the three criteria.

In the end of the optimization process an application is obtained with improved characteristics, independent of the selected software criteria set. By all of this the producer aims the effect maximization, concentrating only on the most important characteristics. The reason is given by limited resource quantities and the value of the quality cost report [IVAN07].

The effective optimization is just a stage in this complex process, as it is preceded by the identification of the areas that improved lead to developing an application with a quality level that corresponds to the user's requirements [IVAN07].

At the developer's or programmers level the effort optimization is completed by:

- reusing the authentication modules or procedures
- reusing the database access procedures;
- avoiding the transmission of large information quantities for processing through specific GET or POST methods;
- shrinking the HTML sources by using XML, XSLT, XSL, CSS files;

- using dynamical components for pages like JSP, ASP or ASP.NET;
- writing code using optimal algorithms;
- using image compression and file archiving algorithms.

At user level optimization is realized by:

- minimizing the transaction duration;
- minimizing the use cost;
- minimizing the maintenance costs;
- minimizing the operation costs.

At host level the optimization is made by minimizing the required server storage space.
These three criteria (developer, user, host) aggregated, form the base of the global optimization criterion.

6. Conclusions

The importance of distributed applications is grater and grater in the informatics society. Currently almost every application has a distributed architecture.
The enumeration of the structures, properties, advantages and disadvantages of distributed applications facilitate selecting the best structure in a certain problem. Only after this stage the optimization process comes up which reefers to the number of nodes, links between nodes, node loading and sizing the data stream.
The optimization process is iterative and complex. It is applied to a working application from which obtaining high levels of performance is required according to certain optimum criteria. These optimum criteria form the base of optimization methods which assume the minimization or maximization of certain problems. Optimization is realized by measuring certain quality evaluation indicators and by applying the optimization methods that lead to improved values for the indicators.
Optimizing distributed applications means defining several application structures with personal characteristics, to which different optimization criteria, and determining the minimum or maximum values for certain metrics or certain quality indicators.
The continuous increase of the application in the distributed domain and their growing diversity assumes a careful and continuous selection in order to ensure customer satisfaction in any domain.
The developed informational society operates with distributed applications which use a complex structure this enforces continuing concentrating the research efforts to finding ways for ensuring the quality growth of these applications.

Bibliography

[BOIA00] Florin Mircea BOIAN - *Programarea distribuită în Internet-metode şi aplicaţii*, Editura Albastră - Grupul microInformatica, Cluj-Napoca, 2000

[IVAN06] Ion IVAN , Eugen DUMITRAŞCU, Marius POPA, *Evaluating the Effects of the Optimization on the Quality of Distributed Applications,* Economic Computation and Economic Cybernetics Studies and Research, vol. 40, nr. 3-4, 2006, pg. 73 - 85, ISSN 0424-267X.

[IVAN07] Ion IVAN, Cătălin BOJA, *Practica optimizării aplicaţiilor informatice*, Editura ASE, Bucureşti, 2007, ISBN 978-973-594-932-7

[IVBO05] Ion IVAN, Catalin BOJA, *Managementul calităţii proiectelor TIC*, Editura ASE , Bucureşti, 2005, ISBN 973-594-558-4.

[IVBO06] Ion IVAN, Cătalin BOJA - *Analiza Metricilor Software*, Studii şi Cercetări de Calcul Economic şi Cibernetică Economică, vol. 40, nr. 1, 2006, pg.65 - 78, ISSN 0585-7511

[IVPO05] Ion IVAN, Marius POPA, Entităţi text, dezvoltare, evaluare, analiză, Editura ASE, Bucureşti, 2005, ISBN 973-594-663-7