JOURNAL OF APPLIED QUANTITATIVE METHODS

Reliability and Quality Control – Practice and Experience

OPTIMAL REDUNDANCY ALLOCATION FOR INFORMATION MANAGEMENT SYSTEMS

Cezar VASILESCU[1]
PhD, Associate Professor
National Defense University, Bucharest, Romania

E-mail: caesarv@crmra.ro

Abstract: *Reliability allocation requires defining reliability objectives for individual subsystems in order to meet the ultimate goal of reliability. Individual reliability objectives set for software development must lead to an adequate ratio of time-length, level of difficulty and risks, as well as decrease development process total cost.*
Thus, redundancy ensures meeting the reliability request by introducing a sufficient quantity of spare equipment. But in the same time, this solution leads to an increase in weight, size and cost.
The aim of this paper is the investigation of reliability allocation to specific sets of software applications (AST) under the circumstances of minimizing development and implementation costs by using the Rome Research Laboratory methodology and by complying with the conditions of costs minimization triggered by the introduction of redundancies [GHITA 00].
The paper analyses the ways in which the software reliability allocation gradual methodology can be extended. It also analyses the issue of optimal system design in terms of reliability allocation by using instruments of mathematical programming and approaches the variation of reliability and system cost by taking into account the redundancy introduced in the system.
This paper is also going to provide an example of calculus which uses a representative software system and illustrates the methodology of optimal allocation of specific sets of software applications reliability.

Key words: *Reliability allocation; Optimal redundancy; Increase of software applications reliability; Application software tools*

Introduction

Reliability allocation requires defining reliability objectives for individual subsystems in order to meet the ultimate goal of reliability. Individual reliability objectives set for software development must lead to an adequate ratio of time-length, level of difficulty and risks, as well as decrease development process total cost.

Thus, redundancy ensures meeting the reliability request by introducing a sufficient quantity of spare equipment. But in the same time, this solution leads to an increase in

weight, size and cost. In this respect, software reliability allocation gradual methodology [ROME 97][2] can be extended to include the approach used in [GHITA 96]. The latter analyses the issue of optimal system design in terms of reliability allocation by using instruments of mathematical programming and approaches the variation of reliability and system cost by taking into account the redundancy introduced in the system.

Consequently, the aim of this paper is the investigation of reliability allocation to specific sets of software applications (AST) under the circumstances of minimizing development and implementation costs by using the Rome Research Laboratory methodology and by complying with the conditions of costs minimization triggered by the introduction of redundancies [GHITA 00].

Before proceeding any further some theoretical clarifications are needed. Firstly, reliability allocation as viewed by [ROME 97] refers to allotting reliability specifications at system level to software module level (be there a non-redundant configuration). Reliability allocation as viewed by [GHITA 00] refers to the optimal allocation of redundancy in order to reach the reliability level set through reliability specifications. In conclusions, the complementarity of the two approaches is worth mentioning.

Secondly, within the context of information management systems, the term redundancy refers both to the existence of several specific sets of software applications (AST) developed and designed independently and which have the same functions, and to testing and upgrading these sets.

All this considered, this paper is also going to provide an example of calculus which uses a representative software system and which illustrates the idea of the possibility of merging the two methodologies. Moreover, the conclusion that is to be drawn is that the modeling of the AST reliability increase by technological means (i.e. by testing and upgrading the software) and by redundancy is a necessity.

The Increase of Software Applications Reliability through Redundancy

The hypothesis underlying the analysis of the software reliability increase of the information management systems is that these systems are part of those systems that are fault-tolerant. In this respect, 'redundancy' (viewed as the use within a system of more elements than necessary for its functioning in order to have the system run flawlessly even in the presence of breakdowns/failures [SERB 96]) is the basic element that assures the reliability of these systems. Other elements may concern hardware or software subsystems and can be traced at any level, starting from individual components up to the whole system (hardware and/ software).

With regard to reliability, the information management systems software has a hierarchical functional partition, beginning with the Mission Specific Tools Set (MSTS), Software Applications sets (AST) and the software modules within them, all of which including redundant components and mechanisms to reestablish the functioning.

The basic methods from the fault tolerance theory for the hardware field can be adapted and applied to the software of the information management systems. Thus, in order to assure its tolerance to failures, encoding logical functions by using redundant codes, error recognition and error removal by screening faults with the help of multiple (redundant) software modules installed in different system equipments or functional reconfiguration of

the system by activating a spare software element that is to replace the failed element can be used.

These methods underlie the suggestions made by the information management systems designers to use the following basic forms of redundant software architectures (forms that assure an increase in reliability regardless of the hierarchical functional level - MSTS, AST, software module):

- The triple modular redundancy. It includes three identical functional modules that carry out similar tasks. Their results are subject to the process known as 'voting' that screens a possible erroneous functioning of one of the modules.
- Duplication with comparison. It is based on two functional modules that assure carrying out similar tasks. If due to their parallel functioning results (outputs) differ, diagnosis procedures are carried out to identify the faulty module.
- Dynamic redundancy. It contains several modules with similar functions. However, only a part of the functions are operational, whereas the other is on stand-by. When a failure is identified the ones on stand-by become operational and take over the tasks of the faulty ones.

All these three basic forms of redundant software architectures are to be found in the implementation of the specific sets of software applications (AST).

In order to evaluate the latter's reliability performance this paper starts from the hypothesis that screening faults is instantaneous and that the faults of the individual copies of ASTs are independent. Moreover, I am to employ reliability logical models conventionally represented in a way similar to those specific to the evaluation of the reliability functions for redundant hardware structures.

The following examples display the evaluation of AST reliability performance using as bibliography the evaluation of reliability functions of redundant structures [SERB 96].

Example 1

The triple modular redundancy made up of identical ASTs

The triple modular redundancy is made up of three identical ASTs where $R_{AST}(t)$ is their reliability function and a voter where $R_V(t)$ is its reliability function. The reliability function of the triple modular redundancy can be modeled by starting from the logical reliability model (fig. 1)

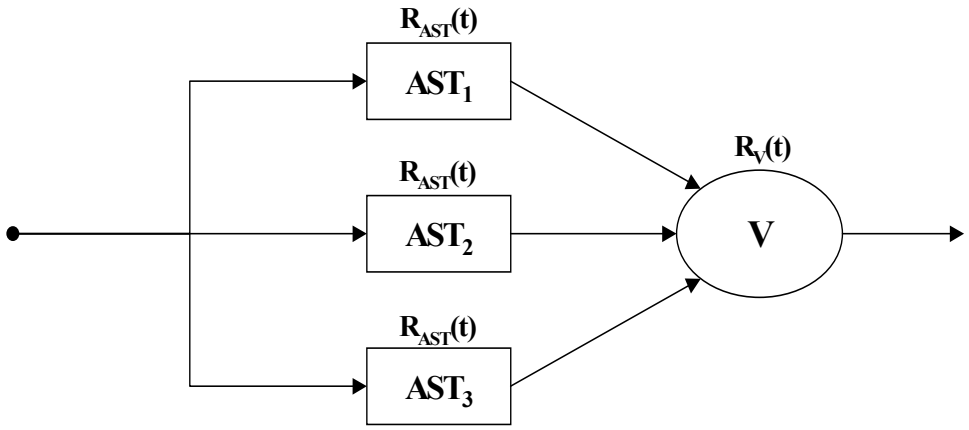

Figure 1. The reliability logical model for the triple modular redundancy

For a good functioning of the software system, at least 2 ASTs and the V voter must function correctly.

The functioning probability of the redundant system under discussion is given by the general formula [SERB 96] for k-out-of-n systems:

$$R = R_V(t)\left[\sum_{i=k}^{n} C_n^i \times R(t)^i \left(1-R(t)\right)^{n-i}\right]$$

which is

$$R = R_V(t)\times\left(3R_{AST}(t)^2 - 2R_{AST}(t)^3\right)$$

Example 2

The triple modular redundancy made up of non- identical ASTs

The three ASTs perform the same functions but they are different in terms of design and implementation. The reliability logical model is similar to the one in fig. 1, with the observation that the ASTs have different reliabilities which are given the notation $R_{AST_i}(t)$.

In order to calculate the reliability function the method of exhaustive enumeration of system states is used. In table 1 the probabilities of correct functioning of the system and the probabilities associated to these events are presented.

Table 1. The probabilities of good functioning of the system with non-identical ASTs

Seq.	The events assuring the good functioning	The probability of the event
1.	$AST_1 \cap AST_2 \cap AST_3$	$R_{AST_1}(t)\times R_{AST_2}(t)\times R_{AST_3}(t)$
2.	$AST_1 \cap AST_2 \cap \bar{AST}_3$	$R_{AST_1}(t)\times R_{AST_2}(t)\times\left(1-R_{AST_3}(t)\right)$
3.	$AST_1 \cap \bar{AST}_2 \cap AST_2$	$R_{AST_1}(t)\times\left(1-R_{AST_2}(t)\right)\times R_{AST_3}(t)$
4.	$\bar{AST}_1 \cap AST_2 \cap AST_3$	$\left(1-R_{AST_1}(t)\right)\times R_{AST_2}(t)\times R_{AST_3}(t)$

The good functioning of the system is assured by joining all four events. They are incompatible with one another and thus the probability of the good functioning of the triple modular redundancy is:

$$\begin{aligned}R(t) &= R_V(t)\left[R_{AST_1}(t)\times R_{AST_2}(t)\times R_{AST_3}(t) + R_{AST_1}(t)\times R_{AST_2}(t)\times\left(1-R_{AST_3}(t)\right)+\right.\\ &\left.+ R_{AST_1}(t)\times\left(1-R_{AST_2}(t)\right)\times R_{AST_3}(t) + \left(1-R_{AST_1}(t)\right)\times R_{AST_2}(t)\times R_{AST_3}(t)\right]=\\ &= R_V(t)\left[R_{AST_1}(t)\times R_{AST_2}(t) + R_{AST_1}(t)\times R_{AST_3}(t) + R_{AST_2}(t)\times R_{AST_3}(t) -\right.\\ &\left.- 2R_{AST_1}(t)\times R_{AST_2}(t)\times R_{AST_3}(t)\right]\end{aligned}$$

In the two examples, the modeling of the reliability function of the AST redundancies does not take into account the instances of error-compensation. Consequently,

the probability of the event to have m failures in AST_1, n < m failures in AST_2, r < n failures in AST_3 and the three ASTs to function:

$$P = R_{AST}(t)\frac{(\lambda t)^m}{m!} \times R_{AST}(t)\frac{(\lambda t)^n}{n!} \times R_{AST}(t)\frac{(\lambda t)^r}{r!}$$

where (λ) is the failure rate of an AST.

There is a number of P_{mnr} permutations for the triple (m, n, r) with a view to identifying the errors of the three ASTs:

$$P_{mnr} = \begin{cases} 1, m = n = r \\ 3, m = n \text{ sau n} = \text{r} \\ 6, \text{m} > \text{n} > \text{r} \end{cases}$$

Each triple is associated with a $\Pr_{\text{m,n,r}}$ conditioned probability defined as:

"The AST system functions correctly if it contains m, n or r errors', where m can be set to any value, n < m and r < n.

Consequently, the reliability function of the AST is calculated according to the relation:

$$R(t) = \sum_{m=0}^{\infty}\sum_{n=0}^{m}\sum_{r=0}^{n} P_{mnr}\,\Pr_{m,n,r}\,R_{AST}(t)\frac{(\lambda t)^m}{m!}R_{AST}(t)\frac{(\lambda t)^n}{n!}R_{AST}(t)\frac{(\lambda t)^r}{r!} =$$

$$= R_{AST}(t)^3\sum_{m=0}^{\infty}\sum_{n=0}^{m}\sum_{r=0}^{n} P_{mnr}\,\Pr_{m,n,r}\frac{(\lambda t)^{m+n+r}}{m!\times n!\times r!}$$

$$R(t) = P_{000}\,\Pr_{0,0,0}\,R_{AST}(t)^3 + R_{AST}(t)^3\sum_{m=1}^{\infty} P_{m00}\,\Pr_{m,0,0}\frac{(\lambda t)^m}{m!} +$$

$$+\sum_{m=1}^{\infty}\sum_{n=1}^{m}\sum_{r=0}^{n} P_{mnr}\,\Pr_{m,n,r}\frac{(\lambda t)^{m+n+r}}{m!\times n!\times r!}$$

By acknowledging that for software systems there is an exponential repartition for run time, for which $R_{AST}(t) = e^{-\lambda t}$, it results:

$$\frac{1}{R_{AST}(t)} = \sum_{m=0}^{\infty}\frac{\lambda t}{m!} = 1 + \sum_{m=1}^{\infty}\frac{(\lambda t)^m}{m!}$$

or

$$\sum_{m=1}^{\infty}\frac{(\lambda t)^m}{m!} = \frac{1 - R_{AST}(t)}{R_{AST}(t)}$$

By replacing, it results:

$$R(t) = R_{AST}(t)^3 + 3R_{AST}(t)^2\left(1 - R_{AST}(t)\right) + R_{AST}(t)^3\sum_{m=1}^{\infty}\sum_{n=1}^{m}\sum_{r=0}^{n} P_{mnr}\,\Pr_{m,n,r}\frac{(\lambda t)^{m+n+r}}{m!\times n!\times r!}$$

Example 3

Dynamic redundancy

Be there a dynamic redundancy made up of two ASTs, a basic (functional) one - AST1 and a spare one - AST2. The spare AST can be functioning or on stand-by and can be identical (or not) with the functional one.

The following notations are to be used:

- $R_{AST\,1}(t)$ - the reliability function of the basic AST;
- $R_{AST\,2}(t)$ - the reliability function of the spare functioning AST;
- $R_{AST\,2r}(t)$ - the reliability function of the spare standby AST.

The logical model of the dynamic redundancy is presented in fig. 2.

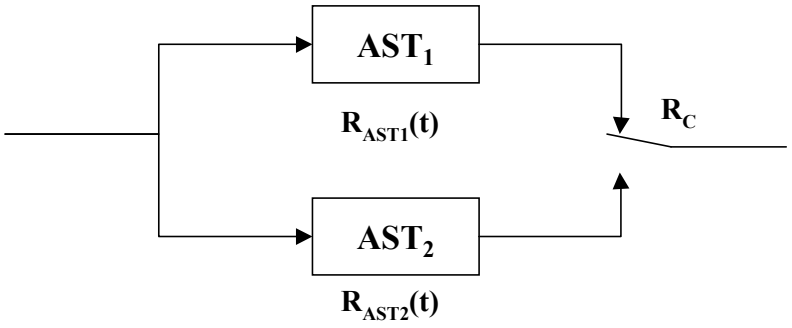

Figure 2. The logical reliability model of the dynamic redundancy

The dynamic redundancy can successfully function on long-term if the following events take place:

1. AST1 (basic AST) functions well for the (0, t) time duration; for the probability of this event we give the notation $\Pr_1 = R_{AST_1}(t)$;
2. AST2 fails at time moment τ, where $\tau < t$; AST (spare AST) is in proper functioning condition and it works well for the time interval (τ, t).

The probability of AST failure within the infinite small time interval $(\tau, \tau + d\tau)$ is $f(\tau)d\tau$, and the probability of AST1 at the τ moment and of the AST2 functioning from the τ moment until the t moment, with AST2 in functioning condition at the τ moment is:

$$f(\tau)R_{AST_{2R}}(\tau)R_{AST_2}(t-\tau)d\tau$$

If $0 < \tau < t$, the probability of the $\Pr_2$ composed event is:

$$\Pr_2 = \int_0^t f(\tau)R_{AST_{2R}}(\tau)R_{AST_2}(t-\tau)d\tau$$

The two events are incompatible. Thus, the probability of a good functioning of the dynamic redundancy is:

$$R(t) = R_{AST_1}(t) + \int_0^t f(\tau)R_{AST_{2R}}(\tau)R_{AST_2}(t-\tau)d\tau$$

The Optimal Allocation of Application Software Redundancies

The problem of reliability allocation issues during the stage of provisional reliability evaluation. The paper [GHITA 00] offers solutions for the optimal allocation of reliability for general situations by tackling the topic of "objects made up component equipments" and puts forward a way of choosing the type of redundancy that best meets the reliability requirement.

In what follows I would like to deal with the issue of adapting the methodology of reliability allocation to the reliability of specific sets of software applications and to present a methodology- adequate calculation program that would enable solving case studies.

The first thing under consideration is the problem of availability allocation (adapted after [SERB 96]) if the IT system is designed as a serial connection of (parallel) redundancies of subsystems.

Usually, system design starts by introducing a minimum number of functionally necessary equipments in its structure. The resulting structure is, from the reliability point of view, a serial one. Since serial structures have the lowest reliability, they may not meet reliability requirements and, consequently, the designer is to increase system reliability starting from redundancy in the number of elements.

By giving the notation of $D_i(m_i)$ to the availability of equipment number "i", equipment which has "m_i" redundant (same type of) equipments and the notation of $m=(m_1, m_2, ..., m_n)$ to redundancy at product level, where n is the number of equipments, it results that D(m) is expressed as:

$$D = \prod_{i=1}^{n} D_i$$

Availability calculation $D_i(m_i)$ depends on the type of redundancy practiced (redundancy through the design of parallel systems, "r out of n", or by using spare equipment).

In the first two alternatives, redundant equipments work under the same conditions as basic equipment does. On the one hand, that assures a technically easier solution. However, the issuing reliability is less good compared to the last alternative.

For this alternative of "parallel" redundancy

$$D_i = 1 - (1 - d_i)^{m_i+1}$$

for $m_i = 0$, it results $D_i = d_i$

$$d_i = \frac{\lambda_i}{\lambda_i + \mu_i}$$

where D_i is the availability of an equipment of type "i".

Through redundancy, the reliability requirements can be met by introducing enough spare equipment. Nonetheless, weight, size and cost increase.

If we give the notation of C (m) to the cost of redundant equipments within the system, the latter is calculated as follows:

$$C(m) = \sum_{i=1}^{n} (c_i * m_i)$$

where c_i is the cost of an equipment of type "i".

From the cost relation it results that the function increases monotonously as against any component m_i. Of all solutions, the one that meets the reliability condition at the lowest cost must be chosen.

In conclusion, the problem of the optimal design of the system is formulated as follows: "of all m redundant solutions, one must find the solution that minimizes the cost C(m), be there a restriction, in which D= D(m) is calculated in accordance with the relations above".

The reliability requests for AST can be expressed as follows:

$$P_{AST} \geq P^* \text{ or}$$

$$K_{D_{AST}} \geq K_D^*$$

where

- P_{AST} is the probability of good functioning;
- $K_{D\,AST}$ is the availability coefficient;
- P^* and K_D^* are the minimum values of reliability indicators.

The reliability requirement can be thus met by [GHITA 96] [GHITA 00]:

a) increasing system's components reliability;
b) increasing (improving) system's reparability;
c) using some reliability redundancies.

The third alternative is going to be discussed in more details in the following paragraphs.

Usually software design starts from the basic principle of a minimum and functionally necessary number of modules within the system. Reliability analysis points out that the latter is a serial structure of low reliability. Reliability increase during the design stage is done by having a redundancy introduced as far as the number of modules is concerned.

If $P_{AST\,i}(m_i)$ is the notation for the probability of good functioning of an AST_i that has m_i redundant modules of the same type and the redundancy at the level of the whole set of ASTs is given the notation $m = (m_1, m_2, ..., m_n)$, where n represents the number of ASTs, it results that $P_{AST}(m)$ is expressed through the relation:

$$P_{AST}(m) = \prod_{i=1}^{n} P_{AST\,i}(m_i).$$

Calculating probabilities $P_{AST\,i}(m_i)$ depends on the type of redundancy employed:

- redundancy by designing systems of parallel software modules;
- redundancy by designing systems of "r out of n" software modules;
- the use of spare software.

The last alternative has the advantage of assuring a reliability increase superior to the other two for which the systems of redundant software modules work in the same manner as the basic ones.

The probability of an AST_i good functioning for the first two alternatives of redundancy is:

$$P_{AST\,i}(m_i) = \sum_{k=r}^{n} C_{m_i+1}^{k} P_i^k (1-P_i)^{m_i+1-k}$$

where

P_i - probability of good functioning of a model of type i;

$P_i = e^{-\lambda_i t}$ (an exponential repartition law follows);

with λ_i - failure intensity of the module type i and t - mission duration.

If r=1 the system is of a parallel type and if r>1 the system is of an r-out- of- n type. As for the redundancy through spare software modules, each module together with the m_i redundant modules forms a kit that fails when the m_i +1 modules fail.

In this case, the probability to have an exact number of k failures is calculated as follows:

$$P_i(k) = P(\nu_i = k) = (\lambda_i t)^k e^{-\lambda_i t},$$

where

λ_i - failure intensity of modules;

ν_i - number of type i failed modules.

But $P_{AST\,i}(m_i) = P(\nu_i \le m_i)$, thus resulting the relation:

$$P_{AST\,i}(m_i) = \sum_{k=0}^{m_i} (\lambda_i t)^k \left(e^{-\lambda_i t}\right)/k!$$

The previous formula is valid if we are to accept the hypothesis according to which failure and module replacement is instantaneously done through a spare module and that probability equals 1. By having the calculus formulas the good functioning of AST_i analyzed it results that they are monotonously increasing functions. In conclusion, regardless of the P* level, there is the possibility of reaching the desired reliability level by including enough redundant software modules.

$$\lim_{m_i \to \infty} P_{AST\,i}(m_i) = 1 \text{ si}$$

$$\lim_{m \to \infty} P_{AST}(m) = 1$$

However, one observation must be made in this respect: by introducing any number of redundant software modules within the structure of an AST, its complexity and cost automatically increase. In all software systems, total cost reduction is an efficiency criterion unanimously accepted. Consequently, an optimal equilibrium between the desired reliability for an AST, the number of redundant software modules and the cost of this activity needs to be reached. In the general concept of "cost" we include the design/ development costs, software maintenance/ exploitation costs and downtime costs.

By giving the cost of introducing within AST the redundant modules the notation $C_{AST}(m)$, its value can be estimated as follows:

$$C_{AST}(m) = \sum_{i=1}^{n} c_i m_i \ ,$$

where c_i is the cost of a module of type i. From the cost relation it results that the function increases monotonously as compared with any component.

Figure 3 presents a qualitative graph of reliability and AST cost variation as compared with the redundancy employed. Each dot on the graph corresponds to a vector m, and by increasing the number of m_i components the dots move up and to the right.

The optimal design of the AST is done by selecting the dots that go over P* (redundant dots that meet the reliability requirement) of the most cost-effective alternative. Thus, it results that the problem of finding an optimal design solution pertains to mathematical programming (optimum with restrictions) and that it displays certain particularities [GHITA 00]:

- it is a problem of non-linear programming - $P_{AST}(m)$ is not a linear function as compared with argument m_i;
- it is a problem of whole numbers programming- arguments m_i are whole numbers.

Consequently, the problem of AST reliability allocation is a problem of whole numbers non- linear programming that is to be worked out by using specific methods.

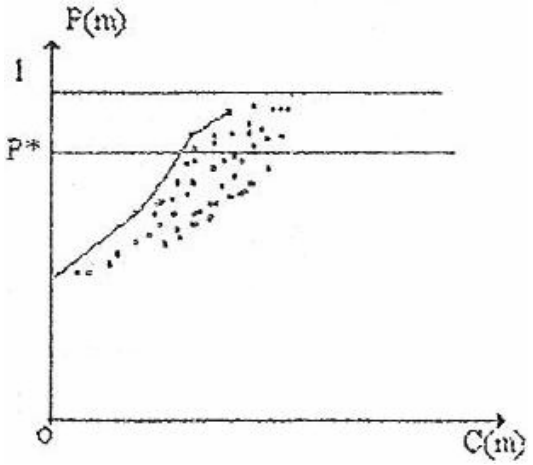

Figure 3. Variation of reliability and cost in accordance with the redundancy

In the graph displayed in figure 3 there is a line of dots on the upper side called dominant vectors and they are optimal solutions as compared with the other dots. Thus the optimal solution is the first vector from the line of dominant vectors that go over level P*. It results that identifying the optimal solution is a matter of using the appropriate methods by which some dominant vectors are obtained.

A vector m′ is called dominant if the following conditions are met:

1. $P(m) > P(m') \Rightarrow C(m) > C(m')$
2. $P(m) = P(m') \Rightarrow C(m) \geq C(m')$

The identification of the dominant vectors is done by using the functions:

$$\varphi_i(k) = \frac{1}{c_i} \ln \frac{P_i(k+1)}{P(k)}$$

which evaluate the probability increase per unit of cost for a component with k redundancies. All procedures are workable if the functions $\varphi_i(k)$ are convex and for the previously mentioned structures (r- out- of- n systems or spare ones) $\varphi_i(k)$ are convex.

The **procedure** below supplies a line of dominant vectors $m(k)=(m_1(k), m_2(k), ..., m_n(k))$, k=1, 2, …, N in which

1. m(1)= (0, 0, …, 0)
2. m(k+1) is recurrently deduced as follows

   $m_i(k+1)=m_i(k)+1$ daca i = I

   $m_i(k+1)=m_i(k)$ daca i ≠ I

   where I is the index number that maximizes function $\varphi_i(k)$; (if there are more indices I, in order to obtain maximum possible one of them is selected as index I);
3. Algorithm stall results from:

   $N=\min\{k : P(m(k))\geq P^*\}$

In conclusion, the procedure leads to a line of dominant vectors deduced one from the other by adding one unit for each argument that reaches the greatest increase in probability per unit of cost.

The chain begins with the identical null vector and ends with the first vector that meets the reliability condition (C). This procedure supplies a chain of dominant vectors that does not necessarily include all possible dominant vectors between the identical invalid vector and vector $m(k)$. Consequently, it does not always supply an optimal solution, but a quasi-optimal one. The procedure has the advantage of completely taking algorithm form and of being easy to implement on a computer. Its main disadvantage resides in the fact that it starts from vector (0, 0, ..., 0) and thus a number of steps must be taken towards finding the first dominant vector that meets the reliability and cost requests.

A more direct method (with fewer steps) towards obtaining a chain of dominant vectors is the one recommended in [BARLOW 92] and which involves using one of the following procedures.

Procedure 1

It is similar to the procedure previously described and it helps determine the whole chain of dominant vectors by starting from vector (0, 0, ..., 0) and successively introducing redundancies in accordance with the increase criterion.

Procedure 2

It is an operational alternative that helps determine one dominant vector $n^*=(n_1^*,...,n_m^*)$ that corresponds to the imposed level of probability P^*. It is based on the particularity that lets probability P^* be, there is a constant value so that all the components of the dominant vector n^* meet the condition:

$$n_i^*=\min\{k : \varphi_i(k)<\delta(P^*)\}.$$

Since functions $\varphi_i(k)$ are positive and monotonously decreasing and $\delta(P^*)>0$, the previous relation always assures finding components n_i^*. The advantage of this procedure consists in directly supplying vector n^* that corresponds to the imposed

probability level P^*. The disadvantage lies not in offering any clue as to the manner of choosing the constant value $\delta(P^*)$, which is done through successive trials.

Procedure 3

It consists in joining previous procedures by using their advantages. If a level of probability P^* is imposed, an estimate value for the constant $\delta(P^*)$ and its corresponding dominant vector $n(\delta)$ are established through successive trials, so that $P(t,n(\delta)) < P^*$ by using procedure 2.

Once $n(\delta)$ is established, by using procedure 1 the chain of dominant vectors is established in its turn until the dominant vector n^* that meets condition $P(t,n^*) \geq P^*$ is obtained.

Case Study: The Methodology of Optimal Allocation of AST Reliability

In this sub-chapter we give an example that illustrates the methodology of optimal allocation of AST reliability [VASILESCU 05] by using the optimized method of dominant vectors calculation that was explained in detail in the previous paragraphs (procedures 1-3).

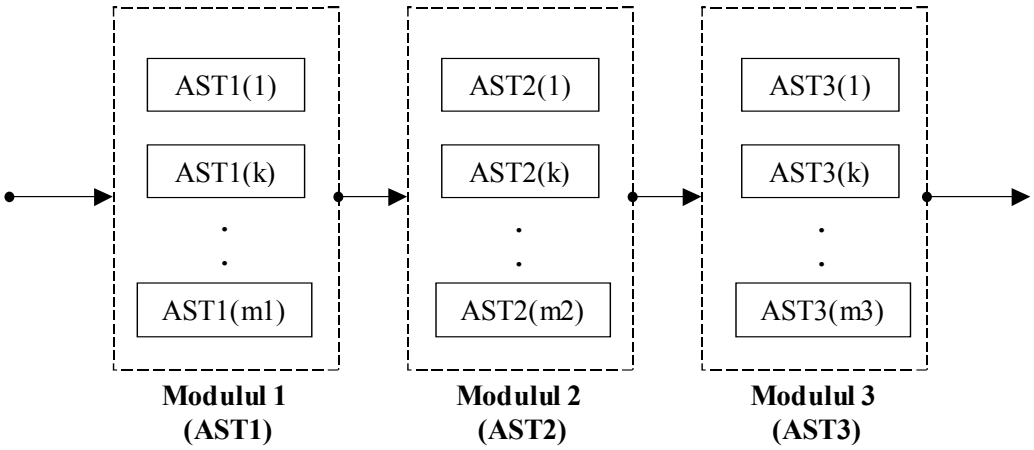

Figure 4. Specific set of software applications (AST) with redundancies at the software modules level

In order to set the basis of this calculus, here are the initial data of the problem. We analyze an IT system, in which a command and control activity is supported through a specific set of software applications (AST) consisting of three software modules (figure 4).

Table 3 depicts the failure intensities and their specific costs.

Table 3. Specific set of software applications - initial data

l	λ_{ASTi} (hour^{-1})	c_{ASTi} (u.c.)
1	0,0008	200
2	0,0005	300
3	0,0003	250

For maximum generality we preferred expressing the c_{AST} values in unit costs (u.c.). For an effective analysis of this case study and in order to obtain relevant results we analyzed the functioning of the three ASTs for the time length of t=3000 hours.

The reliability of the software applications set is increased by introducing a redundancy within its modules. The type of redundancy chosen is the one based on introducing some spare software modules.

The reliability requirement is $P^*_{AST} = 0,95$.

Problem: The optimal design of the AST by choosing the redundancy alternative that meets the reliability requirement and that involves the lowest cost for the redundant modules.

The solution to this problem is given by procedure 3, for $P^*_{AST} = 0,95$.

In order to establish an orientative value for the constant $\delta(P^*)$ we take $\hat{P} = 0,75 < P^*_{AST} = 0,95$ as a probability and assume that all its components $\hat{P}_{AST1}, \hat{P}_{AST2}, \hat{P}_{AST3}$ are equal. Consequently, $\hat{P}_{AST1} = \sqrt[3]{\hat{P}} = \sqrt[3]{0,75} = 0,91$.

The intermediary results are provided in table 4.

As we notice from the table, $P_{AST1}(7) = 0,8867$ is the closest in value to $\hat{P}_{AST1} = 0,91$ and consequently/ it results that,

$$\hat{n}_{AST1} = \max\left\{k : P_{AST1}(k) < \hat{P}_{AST1}\right\} = 7.$$

Table 4. Establishing the orientative value of the constant $\delta(P^*_{AST1})$ - intermediary results

k	$P_{AST1}(k)$	$R_{AST1}(k)$
0	0,0082	0,0082
1	0,0395	0,0477
2	0,0948	0,1425
3	0,1517	0,2942
4	0,1820	0,4763
5	0,1747	0,6510
6	0,1398	0,7908
7	0,0959	0,8867
8	0,0575	0,9442

If $n^*_i = \min\left\{k : \varphi_i(k) < \delta(P^*)\right\}$, the orientative value is:

$$\hat{\delta} = \varphi_{AST1}\left(\hat{n}_{AST1}\right) = \varphi_{AST1}(7) = \frac{1}{c_1}\ln\frac{P_{AST1}(8)}{P_{AST1}(7)} = 0,00314$$

By using procedure 2 we obtain the components of the dominant vector n^* that have to meet the condition:

$$n^*_{ASTi} = \min\left\{k : \varphi_{ASTi}(k) < \delta(P^*)\right\},$$

As a result, the first φ_{AST2} that meets the previous condition is $\varphi_{AST2}(8)=0{,}000260$, namely the first φ_{AST3} that meets the previous condition is $\varphi_{AST3}(6)=0{,}000179$. It results the dominant vector $\hat{n}=(7,8,6)$ with its corresponding probability $\hat{P}=0{,}63<0{,}95$.

Moreover, by employing the rule of increase from procedure 1 the result is the chain of dominant vectors presented in table 5 and their corresponding values $P(n)$ and $C(n)$.

Table 5. Chain of dominant vectors

n_{AST}			$P(n)$	$C(n)$
n_{AST1}	n_{AST2}	n_{AST3}		
7	8	6	0,639351	5300
7	9	6	0,790611	5600
7	10	6	0,889451	5900
7	11	6	0,946202	6200
8	11	6	0,975635	6400

We notice that the first dominant vector that meets the condition $P(n^*)\geq 0{,}95$ at the lowest cost is $n_{AST}=(8,11,6)$, whereas its corresponding probability is $P(n)=0{,}975$ at the cost of $C(n)=6400$ u.c.

We also observe that by using procedure 3 we needed 5 steps to obtain the result, whereas for the procedure 1 we would have needed 21 extra steps.

In order to implement formulas and do the calculations we used Microsoft Excel due to the possibility it offers to introduce initial data in a rapid manner, and also because of the elegant and explicit layout for the results provided by its spreadsheets.

Excel spreadsheet cells explanation:

- D4:D6 - intensity of failures in the modules that were given the notations AST_1, AST_2 and AST_3;
- E4:E6 - modules specific cost;
- H3 - mission duration in hours;
- B9:B21 - number of redundant modules introduced;
- D9:D21 (H9:H21, L9:L21) - values of modules reliability; for example, cell D10 contains the formula
  =D9+POWER(F4*D4*H3;B10)/FACT(B10)*EXP(-(F4*D4*H3));
- E9:E21 (I9:I21, M9:M21) - values of the reliability increase for modules per unit cost; for example, cell E10 contains the formula *=(1/E4)*LN((D11/D10))*;
- C25:E25 (C29:E29) - chains of dominant vectors; for example, cell C26 contains the formula *=IF(MAX(E17;I17;M17)=E17;C25+1;C25)*;
- F25:F29 - values of overall AST reliability; for example, cell F25 contains the formula *=D16*H16*L16*;

G25:G29 - values of overall AST costs; for example, cell G25 contains the formula *=C25*E4+D25*E5+E25*E6*.

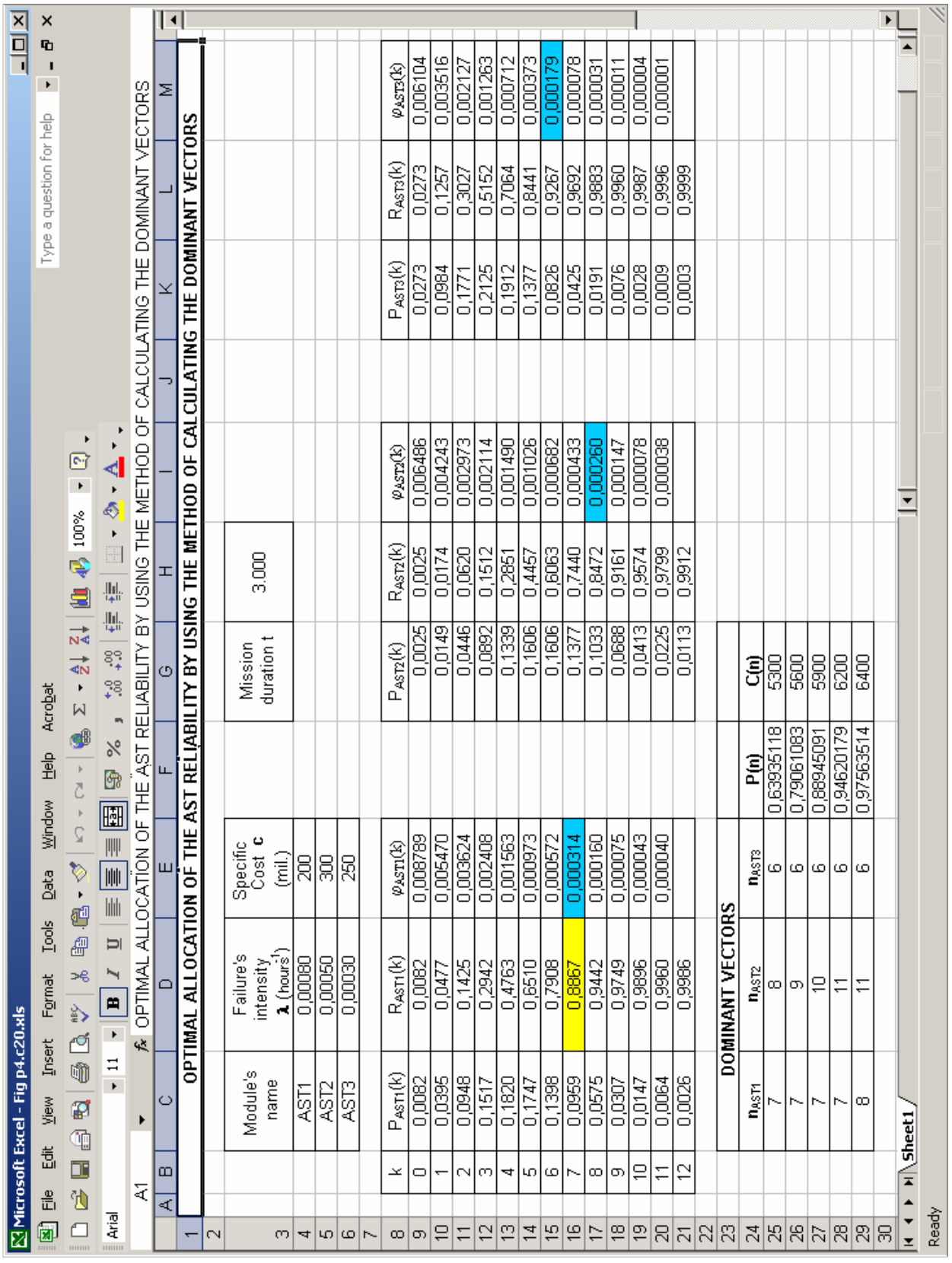

Figure 6. Optimal allocation of the AST reliability by using the method of calculating the dominant vectors- results

In conclusion, in order to meet the imposed reliability requirement a specific set of software applications tools is needed. The latter includes: eight modules type one, eleven modules type two and six modules type three. The cost of the AST is 6400 unit cost.

References

1. BARLOW, R., PROSCHAN, F. **Mathematical Theory of Reliability,** John Wiley, New York, 1998
2. GHITA, A., IONESCU, V. **Metode de calcul în fiabilitate,** Editura Academiei Tehnice Militare, Bucharest, 1996
3. GHITA, A., IONESCU, V., BICA, M. **Metode de calcul în mentenabilitate,** Editura Academiei Tehnice Militare, Bucharest, 2000
4. SERB, A. **Sisteme de calcul tolerante la defectari** Editura Academiei Tehnice Militare, Bucharest, 1996
5. VASILESCU, C. **Alocarea optima a fiabilitatii seturilor specifice de aplicatii software din sistemele C4ISR,** The 10th International Scientific Conference, Academia Fortelor Terestre, Sibiu, November 24-26, 2005
6. *** **System and Software Reliability Assurance Notebook,** produced for Rome Laboratory, New York, 1997
7. *** **TR-92-52 - Software Reliability Measurement and Test Integration Techniques,** produced for Rome Laboratory, New York, 1992

[1] Cezar Vasilescu has graduated the Faculty of Electronics and Information Science within the Military Technical Academy - Bucharest in 1997. He holds a PhD diploma in Computer Science from 2006.
He has graduated in 2003 the Advanced Management Program organized by National Defense University of Washington D.C., USA. Also, he has received the US Department of Defense Chief Information Officer (CIO) certification from the Information Resources Management College of Washington D.C.
Currently he is the head of the IT&C Office within the Regional Department of Defense Resources Management Studies - Brasov and associate professor at the National Defense University - Bucharest. He is the author of more than 30 journal articles and scientific presentations at conferences in the fields of hardware/software reliability, command and control systems and information resources management.
He has coordinated as program manager the activity of establishing in Romania of an international educational program in the field of information resources management, in collaboration with universities from USA. Beside his research activity, he has coordinated "Train the Trainers" and "Educate the Educators" activities with international participation.
Main published books:
~ - Information Management, Military Technical Academy Publishing House, Bucharest, 2006.
~ - Information Technology for Management, Regional Center of Defense Resources Management Publishing House, Brasov, 2001.

[2] Codifications of references:

[BARLOW 98]	BARLOW, R., PROSCHAN, F. **Mathematical Theory of Reliability,** John Wiley, New York, 1998
[GHITA 96]	GHITA, A., IONESCU, V. **Metode de calcul în fiabilitate,** Editura Academiei Tehnice Militare, Bucharest, 1996
[GHITA 00]	GHITA, A., IONESCU, V., BICA, M. **Metode de calcul în mentenabilitate,** Editura Academiei Tehnice Militare, Bucharest, 2000
[ROME 92]	*** **TR-92-52 - Software Reliability Measurement and Test Integration Techniques,** produced for Rome Laboratory, New York, 1992
[ROME 97]	*** **System and Software Reliability Assurance Notebook,** produced for Rome Laboratory, New York, 1997
[SERB 96]	SERB, A. **Sisteme de calcul tolerante la defectari** Editura Academiei Tehnice Militare, Bucharest, 1996
[VASILESCU 05]	VASILESCU, C. **Alocarea optima a fiabilitatii seturilor specifice de aplicatii software din sistemele C4ISR,** The 10th International Scientific Conference, Academia Fortelor Terestre, Sibiu, November 24-26, 2005