Y. K. Malaiya, "Optimal Reliability Allocation," Wiley Encyclopedia of Operations research and Management Science, John Wiley & Sons, Jan. 14, 2011.

Optimal Reliability Allocation

Wiley Encyclopedia of Operations Research and Management Science

Yashwant K. Malaiya
Computer Science Dept.
Colorado State University, Fort Collins CO 80523
malaiya@cs.colostate.edu
Phone: 970-491-7031, FAX 970-491-2466

Abstract--- The overall reliability of a complex system, designed as an assembly of subsystems, depends on the subsystem reliability metrics. The cost of each subsystem is a function of its reliability. This article considers the problem of allocating the reliability values to minimize the total cost while achieving the reliability target. The reliability allocation problem is applicable to mechanical and electrical systems as well as computer hardware and software. The approach involves expressing the system reliability in terms of the subsystem reliability values and specifying the cost function for each subsystem, which allows setting up an optimization problem. Software reliability allocation is examined as a detailed example and some preliminary apportionment approaches are presented. While in many cases, it is possible to use exact optimization methods, a complex case may require use of iterative approaches.

I. INTRODUCTION

A large system is implemented by using a set of interconnected subsystems. While the architecture of the overall system is often fixed because of the functional requirements, implementation choices are generally available for individual subsystems. A designer needs to achieve the target reliability while minimizing the total cost, or alternatively maximize the reliability while using only the available budget. Intuitively, some of the lowest reliability components may need special attention to raise the overall reliability level. Such optimization problems arise while designing a complex software or a hardware system. Such problems also arise in mechanical or electrical systems. A number of studies since 1960 have examined some of the computational aspects of such problems [1].

All the subsystems are essential in a non-redundant system. However, an individual subsystem can often be made more reliable by using a more costly implementation. This additional cost may represent wider columns in a building or more thorough testing of software. In redundant implementations, higher reliability can sometimes be achieved by using several copies of a subsystem, such that the system incorporates a *parallel* or a *k-out-of-n* configuration.

The problem formulation is considered in the next section, followed by approaches used for setting up an optimization problem. As a detailed example, software reliability allocation is examined in detail with a numerical illustration. The last section considers reliability allocation in general complex systems.

II. Allocation as an Optimization Problem

We consider a system that has been designed at a higher level as an assembly of appropriately connected subsystems. The functionality of each subsystem is often unique; however, there can be several choices for many of the subsystems that provide the same functionality but differently reliability levels.

Here we consider the problem formulation for a common and widely applicable case. Let there be n subsystems SS_i, i = 1,.., n, each with reliability R_i and cost C_i. Let the cost C_i be a function of the reliability given by $f_i(R_i)$, this is generally referred to as the *cost function*. Let C_s and R_s represent the total system cost and the overall reliability and let R_{ST} be the specified target reliability. If all the subsystems are essential to the system and if their failures are statistically independent, the system can be modeled as a *series system*. The cost minimization problem can be stated as:

Minimize
$$C_s = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} f_i(R_i) \tag{1}$$

Subject to

$$R_{ST} \le R_s$$
$$i.e. \quad R_{ST} \le \prod_{i=1}^{n} R_i \quad \text{since } R_S = \prod_{i=1}^{n} R_i \tag{2}$$

Note that equation (1) assumes that the cost of interconnecting the subsystems is negligible. An alternative problem would be to maximize R_s while keeping C_s less than or equal to the allocated cost budget.

Depending on the type of the system, the i^{th} subsystem SS_i may have several implementation choices with different reliability values:

A. The subsystem can be made more reliable by extending a continuous attribute (for example diameter of a column in building or time spent for software testing).
B. Different venders may offer their own implementations of SS_i at different costs.
C. It may be possible to use multiple copies of SS_i (for example double wheels of a truck) to achieve higher reliability. Often the number of copies is constrained between a minimum (often one) and a maximum number because of implementation issues.

In the first case, both cost and the reliability can be varied continuously, whereas in the other two cases, the choices are discrete. In the first case, we can define a continuous cost function. In the second case too, the market forces may impose a cost function. In the third case, the subsystem may be modeled as a *parallel* or *k-out-of-n system* for reliability evaluation, provided the failures are statistically independent. A number of publications on reliability allocation consider only the third case, where the optimization problem becomes an integer optimization problem. It becomes a 0-1 optimization problem when choices are discrete and a component from a given list of candidates is either used or not used [2].

III. The Cost function and problem set-up

It is reasonable to assume that the cost function f_i would satisfy these three conditions [3]:

1) f_i is a positive function
2) f_i is non-decreasing, thus higher reliability will come at a higher cost.
3) f_i increases at a higher rate for higher values of R_i

The third condition leads to the fact that it can be very expensive to achieve the reliability value of 1. In fact, for software, it has been shown that it is infeasible to achieve ultra-high reliability in software [4]. Note that the cost function in general will also be a function of the relative complexity of the subsystem.

In some cases, the cost function can be derived from basic considerations, as we will do below for software reliability. In other cases, it may be derived empirically by compiling data for different choices. The cost function is often stated in terms of the reliability, for example, the cost function proposed by Mettas [3] is given in terms of the maximum and minimum achievable reliability values.

In some cases, there are choices that can be made that can enhance reliability with little or no additional cost, sometimes by using newer technology. For example, some disciplined software development processes have been found to yield more reliable software with the same effort. We assume that such choices have already been made, and optimization is sought by exploiting the available choices of attributes.

The cost function can also be given in terms of the failure rate as illustrated below for software reliability allocation.

A transformation of equation (2) can be obtained by logarithms of both sides of the equation [5, 6, 7].

$$\ln(R_{ST}) \leq \sum_{i=1}^{n} \ln(R_i) \tag{3}$$

The transformation in equation (3) can reduce the problem to a linear optimization problem. In some cases, the term $\ln(R_i)$ also has a well-defined physical significance and is termed the *failure rate*. When the failure rate of a subsystem SS_i is constant, its reliability is given by an exponential relationship $R_i(t) = \exp(-\lambda_i t)$, the system failure rate is given by the summation of the subsystem failure rates and hence equation (3) can be restated as

$$\lambda_{ST} \geq \sum_{i=1}^{n} \lambda_i \tag{4}$$

The failure rate is itself a frequently used reliability metric. In some cases such as in software reliability engineering, it is the failure rate that is often specified [8, 9]. The cost function of a subsystem can also be given in terms of its failure rate. If the cost C_i is given by the function $f_i(\lambda_i)$, equation (1) can be restated as

Minimize $$C_S = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} f_i(\lambda_i) \quad (5)$$

For example, let us consider software reliability. When a software is tested, defects in it are found and removed by debugging. A program tested more thoroughly will have fewer bugs and hence higher reliability. Several models relating software reliability growth with the time spent in testing, have been proposed and validated. These are termed software *reliability growth models* (SRGMs). For the popular *exponential software reliability growth model* [8, 9, 10], the failure rate as a function of testing time d is given by

$$\lambda_i(d) = \lambda_{0i} \exp(-\beta_i d)$$

where λ_{0i} and β_i are the model parameters. It should be noted that λ_{0i} depends on initial defect density and βi depends inversely on program size [11]. If we assume that the variable cost is dominated by the testing time, the cost is given by the following function, which satisfies the three conditions mentioned above.

$$d(\lambda_i) = \frac{1}{\beta_i} \ln\left(\frac{\lambda_{0i}}{\lambda_i}\right) \quad (6)$$

IV. Reliability Allocation for Basic Series and Parallel Systems

The reliability allocation problem for two basic reliability structures *series* and *parallel* can be solved by linearizing the constraints [1, 2, 7]. In a *series* system, the constraint is given in equation 2 above, which can be linearized by rewriting it as given in equation (3) above, which may then be solved relatively easily. An example is given below for software reliability allocation.

In a *parallel system*, functionally identical subsystems are configured such that correct operation of at least one of them assures a correctly functioning system. It is assumed that any overhead in implementing such a system is negligible. In real systems, the overhead involved will result in a lower level of reliability. In a number of studies, the problem assumes that the reliability of a subsystem can be increased by using functionally identical components in parallel [2, 7]. For parallel systems, the constraint is given by

$$R_{ST} \leq 1 - \prod_{i=1}^{n} (1 - R_i) \quad (7)$$

The constraint can be linearized by using logarithms of the complements of reliability. Thus, equation (7) can be rewritten as

$$\ln(1 - R_{ST}) \geq \sum_{i=1}^{n} \ln(1 - R_i) \quad (8)$$

Elegbede et al. have recently shown [7] that if the cost function satisfies the three properties given above, the cost is optimal if the all the parallel components have the same cost. For software, computer hardware and mechanical systems, the number of discrete parallel component is likely to be very small.

V. Reliability Allocation in Software Systems

Let us examine the problem of software reliability allocation [6], which serves as a good illustration. Typically, a software consists of sequentially executed components, only one of them is under execution at a time. Each component can be independently tested and debugged to reduce the failure rate below a target value. In some cases, the reliability of a component can be further increased by replication. For replication to be effective, each replicated version must be developed independently such that the failures are relatively independent. The impact of replication can be evaluated by assuming statistical independence. However, it has been shown that sometimes the statistical correlation can be significant, requiring analysis that is more complex.

In this example, we consider a non-redundant implementation of software, divided into n sequential components [6], which is a common case. Let us assume that a component i is under execution for a fraction x_i of the time where $\Sigma x_i = 1$ [5]. Software reliability is often described in terms of the failure rates. Then the reliability allocation problem can be written as

$$\text{Minimize} \quad C = \sum_{i=1}^{n} \frac{1}{\beta_i} \ln\left(\frac{\lambda_{0i}}{\lambda_i}\right) \tag{9}$$

$$\text{Subject to } \lambda_{ST} \le \sum_{i=1}^{n} x_i \lambda_i \tag{10}$$

Solution: let us solve the problem posed by equations (9) and (10) by using the Lagrange multiplier approach by finding the minimum of

$$F(\lambda_1,...,\lambda_n) = C + \theta(\lambda_1 + ... + \lambda_n) - \lambda_{ST}$$

where θ is the Lagrange multiplier. The necessary conditions for the minimum to exist are (i) the partial derivatives of the function F are equal to zero, (ii) $\theta > 0$ and (iii) $x_1\lambda_1+x_2\lambda_2+ \ldots x_n\lambda_n = \lambda_{ST}$ [6]. Equating the partial derivatives to zero and using the third condition, the solutions for the optimal failure rates are found as following

$$\lambda_1 = \frac{\frac{\lambda_{ST}}{x_1}}{\sum_{i=1}^{n} \frac{\beta_1}{\beta_i}} \quad \lambda_2 = \frac{\beta_1 x_1}{\beta_2 x_2}\lambda_1 \quad \cdots \quad \lambda_n = \frac{\beta_1 x_1}{\beta_n x_n}\lambda_1 \tag{11}$$

The optimal testing times of the individual modules, d_i, i=1 to n, are given by equation (12) below.

$$d_1 = \frac{1}{\beta_1} \ln \left(\frac{\lambda_{10} x_1 \sum_{i=1}^{n} \frac{\beta_1}{\beta_i}}{\lambda_{ST}} \right) \quad \text{and} \quad d_i = \frac{1}{\beta_i} \ln \left(\frac{\lambda_{i0} \beta_i x_i}{\lambda_1 \beta_1 x_1} \right) \tag{12}$$

Note that d_i is positive if $\lambda_i \leq \lambda_{i0}$. The testing time for a module must be non-negative. If any of the testing times obtained using (13) are negative, the optimization problem may have be solved iteratively [6] or using special approaches. Sometimes a reused subsystem has a significantly higher reliability because of past testing, which may result in $\lambda_i \geq \lambda_{i0}$. Since the reused subsystem is already reliable enough, it would make sense to spend any testing effort on the other subsystems.

In software reliability engineering, the assumptions involved in formulation of the exponential model imply that the parameter β_i is inversely proportional to the software size [8, 11], measured in terms of the lines of code. In the examples below, we assume that value of x_i is proportional to the subsystem code size. The values of λi and λ_{i0} do not depend on size but depend on the initial defect densities [11] at the beginning of testing. Thus if the exponential model indeed holds, the equation (11) states that the optimal values of the post-test failure rates $\lambda_1, \ldots \lambda_n$ are equal. In addition, if the initial defect densities are also all equal for all the components, then the optimal test times for each module is proportional to its size.

The parameter values needed for (12) and (13) many be determined by some initial testing or using empirical relationships.

Example 1: A software system uses five functional components B1-B5. This example has been constructed assuming sizes 1, 2, 3, 10 and 20 KLOC (thousand lines of code) respectively, and the initial defect densities of 10, 10, 10, 15 and 20 defects per KLOC respectively. Let us assume that measured parameter values are given in the top three rows, which are the inputs to the optimization problem. The solution obtained using equations (11) and (12) are given in the two bottom rows. The test cost has been minimized so that the overall failure rate is less than or equal to 0.04 per unit time. Here the time units can be person hours of testing time, or hours of CPU time used for testing.

Note that the optimal values of λ_i for the five components are equal, even though they started with different initial values. This implies a substantial part of the test effort is allocated to largest components. The total cost in terms of testing is 4984 unit time.

Component	B_1	B_2	B_3	B_4	B_5
β_i	4.59×10^{-3}	2.30×10^{-3}	1.53×10^{-3}	4.59×10^{-4}	2.30×10^{-4}
λ_{i0}	0.046	0.046	0.046	0.069	0.092
x_i	0.028	0.056	0.083	0.278	0.556
Optimal λ_i	0.04	0.04	0.04	0.04	0.04
Optimal d_i	30.1	60.1	90.2	1184	3620

If the total test time were equally distributed for all five components, it would have resulted in significantly higher failure rate of 0.055 per unit time.

VI. Reliability Apportionment rules

The discussion in the section above suggests some preliminary rules may be used for obtaining initial reliability apportionments before a more thorough optimization. Some apportionment rules have been suggested in the literature [5].

Equal reliability apportionment: One can test a set of software components, such that at the end they all individually have the failure rate equal to the target failure rate for the system. Newly developed modules need to be tested until their reliability equals the reliability of the existing modules.

Complexity based apportionment: The software size itself is a complexity metric. Thus, the available test time can be apportioned in proportion to the software size.

Impact based apportionment: A component that is executed more frequently, or is more critical in terms of failures, should be assigned more resources. Note that in the analysis above, x_i can be chosen to reflect the product of frequency and criticality.

VII. Reliability Allocation for Complex Systems

Some systems can have complex reliability allocation choices and may require an iterative approach [1, 2, 7, 12]. Such an approach is also needed if the objective function is multi-objective and includes both total cost and the system reliability [1]. An iterative approach involves the following steps [12]:

1) Design the system using functional subsystems.
2) Perform an initial apportionment of cost or reliability attributes based on suitable apportionment rules or preliminary computation.
3) Predict system reliability.
4) Determine if reallocation is feasible and will enhance the objective function. If so, perform reallocation.
5) Repeat until optimality is achieved.
6) See if this meets the objectives. If not, consider returning to step 1 and revising the design at a higher level.
7) Finalize the design with recommended reliability allocation and the cost projections.

Several optimization methods can be used for steps 2-5 above. These can be classified into three approaches [1].

1) Exact methods: When the problem is not large, exact methods can be desirable. In general, the problem can be a non-linear optimization problem. In a few cases, the problem can be transformed into a linear problem, as shown in the example above.
2) Heuristics based methods: Several heuristics for reliability allocation have been developed. Many of them are based on identifying the variable to which the solution is most sensitive and varying its value.
3) Metaheuristic algorithms: These algorithms are based on artificial reasoning. The best known of them are *genetic algorithms, simulated annealing* and *tabu-search*. These algorithms can be useful when the search space is large and approximate results are sought.

Reliability allocation in systems with replicated subsystems can encounter correlated failures and thus would need a more careful modeling. In software, such correlation can be significant [13]. The factors that affect technology dependant reliability attributes such as defect density in software [14] have been studied by researchers and are still being researched.

Several software tools, both general purpose [15] and special purpose [6], have been developed that can simplify setting up and solving the optimal allocation problem. Reliability allocation problem can also be formulated to address other reliability attributes like availability or maintainability.

REFERENCES

[1] Kuo W, Prasad VR, An Annotated Overview of System-Reliability Optimization, IEEE Transactions on Reliability, June 2000, 49: 176-187.

[2] Majety SRV, Dawande M, Rajgopal J, Optimal Reliability Allocation with Discrete Cost-Reliability Data for Components. Operations Research, Nov-Dec 1999, 47: 899-906.

[3] Mettas A, Reliability allocation and optimization for complex systems. Proceedings Annual Reliability and Maintainability Symposium, Los Angeles, CA, January 2000, 216-221.

[4] Butler RW, Finelli GB, The infeasibility of quantifying the reliability of life-critical real-time software, IEEE Trans. Software Engineering, 1993, 19:3—12.

[5] Lakey PB, Neufelder AM, System and Software Reliability Assurance Notebook; Rome Laboratory, 1996, Rome NY, 6.1-6.24.

[6] Lyu, MR, Rangarajan S, van Moorsel, APA, Optimal allocation of test resources for software reliability growth modeling in software development. IEEE Transactions on Reliability, Jun 2002, 51: 183-192.

[7] Elegbede AOC, Chengbin C, Adjallah KH, Yalaoui F, Reliability allocation through cost minimization, IEEE Transactions on Reliability, March 2003, 52:106- 111.

[8] Musa JD, Iannino A, Okumoto K, Software Reliability, Measurement, Prediction, Application, McGraw-Hill, 1897.

[9] Lyu, MR, Ed., Handbook of Software Reliability Engineering, McGraw-Hill, 1995.

[10] Goel AL, Okumoto K, Time-Dependent Error Detection Rate Model for Software and Other Performance Measures, IEEE Trans. on Reliability, August 1979, 28: 206-211.

[11] Malaiya YK, Denton J, What Do the Software Reliability Growth Model Parameters Represent? Int. Symp. on Software Reliability Engineering, 1997, 124-135.

[12] NASA Document, Organizational Instruction: Reliability Allocation, QDR-008 Rev. E, Oct 1, 2004.

[13] Dai YS, Xie M and Poh KL, Modeling and analysis of correlated software failures of multiple types, IEEE Transactions on Reliability, vol. 54, pp. 100-106, 2005.

[14] Neufelder A, The Facts About Predicting Software Defects and Reliability, The RAC Journal, April 8, 2002, 1-4, http://www.softrel.com/publicat.htm, 2009.

[15] ReliaSoft, Reliability Importance and Optimized Reliability Allocation (Analytical) http://www.weibull.com/SystemRelWeb/blocksimtheory.htm, 2007.