Concise Papers

Optimization Models for Reliability of Modular Software Systems

Oded Berman and Noushin Ashrafi

Abstract—**This paper presents four optimization models to demonstrate that the optimization of software reliability within the available resources can be accomplished. The models help us find the optimal software system structure while considering basic information on reliability and cost of modules. Each model is applicable to a distinct situation. All four models maximize reliability while ensuring that expenditures remain within budget.**

Index Terms— **Dynamic programming, fault tolerance, integer programming, modularization, optimization, software reliability.**

I. Introduction

In developing software systems, the manager's goal is to produce a software system within limited resources and in accordance with user requirements. One important user requirement concerns the reliability of the software. Reliability for a software system is defined as the probability that software operates without failure in a specified environment, during a specified exposure period [5]. Failure is defined as a discrepancy between expected and actual output. Fault is a defect in the program that, when executed, causes a failure. One method to improve software reliability is by the application of redundancy. A careful use of redundancy may allow the system to tolerate faults generated during software design and coding thus improving software reliability. Improving software reliability, using redundancy, however, requires additional resources. The question then is "how to incorporate redundancy into software structure such that reliability is maximized and cost remains under control?"

A number of reliability models for prediction and assessment of the reliability of fault-tolerant software systems have been developed. However, the problem of reliability optimization for fault-tolerant software has not been addressed by many researchers. Belli and Jedrzejowicz [3] contribute this lack of interest partly to "... the complexity of reliability optimization problems, difficulties in identification of dependencies between the resources and component reliabilities ..., and lack of necessary information on components' reliability properties."

The advancement of technology and the immense software development cost has made the use of COTS (commercial off-the-shelf) modules a reality and may be a necessity. Boehm [4] suggests the use of COTS software whenever possible, as an appropriate process model for a software development project. Considering the concept of COTS in software development and the availability of mathematical models to assess module reliability, it is now possible to have information on module reliability and cost. Furthermore, the failure independence of the redundant modules are more acceptable because these modules may be developed by completely different groups, different tools, and in different environments [2].

In this study we show the application of well-known optimization models to determine the optimal redundancy level of fault-tolerant software systems. The models use basic information on module reliability and cost and allow the tradeoff between these two factors. Belli and Jedrzejowicz have developed two optimization models for fault-tolerant software. Their first model is similar to the second model of this study, however, their expression of reliability is quite complicated and no solution to the problem has been offered. Our earlier work [1] used optimization models to determine the redundancy level of a software package consisting of several independent functions where each function is performed by a program with known reliability and cost. Our current work, however, breaks down this approach one step further and deals with software systems consisting of one or more programs where each program consists of series of modules, which upon sequential execution will perform a function. The optimal redundancy level of the modules is to be determined. Four models are presented, each applicable to a different software system structure (ranging from a very simple structure to more sophisticated ones). The diversity of the models gives the software engineer flexibility in choosing an appropriate model for a given software system. The next section explains the structure of the models. Then we present formulations and solution methods for each model. The final section offers concluding remarks and a discussion of the limitation of the proposed models.

A. Notations

- K Number of functions the software system is required to perform.
- n Number of modules within the software system.
- F_k Frequency of the use of function k, $k = 1, 2, \cdots, K$.
- m_i Number of versions available for module i, $i = 1, \cdots, n$.
- R_{ij} Estimated reliability of version j of module i.
- X_{ij} Binary variable that is equal to 1 if version j is selected for module i, else 0.
- R_i Estimated reliability of module i.
- R Estimated reliability of the software system.
- C_{ij} Cost of developing version j for module i.
- B Available budget.

B. Assumptions

1) Modular programming is used for software development.

2) Module versions are developed independently, and their reliabilities and costs can be estimated. Note that this assumption makes the models directly applicable only to those software systems that are developed using COTS modules. These modules are independently generated and tested. Their reliability can be estimated using any of the reliability estimation models available. Their cost is the purchasing cost.

3) There is a specified budget for the software system.

II. Software System Structures

Consider software systems that are developed using modular techniques and are required to perform one or more functions as

Manuscript received January 1993; revised May 1993. Recommended by F. Bustani.

O. Berman is with the Faculty of Management, University of Toronto, Toronto, Ont., M5S 1V1, Canada.

N. Ashrafi is with the Department of Management Science and Information Systems, College of Management, University of Massachusetts/Boston, Boston, MA 02125.

IEEE Log Number 9213119.

specified by the user. Each function is performed by executing a program where each program is further divided into several modules. Each module may be called by more than one program. We assume that functionally equivalent and independently developed versions of modules are available, each with an estimated reliability and cost. Optimization models of this study will determine the optimal redundancy level of the modules for each software structure so as to maximize reliability at a limited given cost. Four models for different software structures are formulated.

A. Model 1: Selecting the Optimal Set of Modules for One Function System (without Redundancy)

Software system consists of a single program performing one major function. The program is comprised of a set of modules, which are executed sequentially. There are more than one version of each module available but, due to budget limitation and/or noncritical nature of the software, keeping multiple versions of modules is not desirable. The model developed for this situation allows the optimal selection of a set of modules for the single program such that the reliability is maximized while meeting the constraint that the overall development cost remains within budget.

Formulation of Model 1: The problem of maximizing reliability by choosing the optimal set of modules can be formulated as follows:

$$\max R = \prod_{i=1}^{n} R_i \qquad (P_1)$$

subject to

$$\sum_{j=1}^{m_i} X_{ij} = 1, \qquad i = 1, \cdots, n \qquad \text{(i)}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i} X_{ij} C_{ij} \le B \qquad \text{(ii)}$$

$$X_{ij} = 0, 1 \qquad i = 1, \cdots, n \qquad j = 1, \cdots, m_i$$

where

$$R_i = \sum_{j=1}^{m_i} X_{ij} R_{ij}. \qquad (1)$$

The objective function of P_1 reflects that the modules are executed sequentially. The set of constraints (i) ensures that exactly one version is selected for each module. Constraints (ii) guarantees that total expenditures will not exceed B.

Problem P_1 is a nonlinear integer programming problem. We suggest a Branch and Bound approach [7] to solve the problem. A simple numerical example for this model is given below. Readers interested in our Branch and Bound scheme can refer to Appendix A.

Suppose $n = 3$, $m_1 = 3$, $m_2 = 3$, and $m_3 = 2$. The reliability and cost of the modules are given as follows:

$$\begin{array}{lll} R_{11} = 0.90, & R_{12} = 0.80, & R_{13} = 0.85 \\ C_{11} = \$3, & C_{12} = \$1, & C_{13} = \$2 \\ R_{21} = 0.95, & R_{22} = 0.80, & R_{23} = 0.70 \\ C_{21} = \$3, & C_{22} = \$2, & C_{23} = \$1 \\ R_{31} = 0.98 & R_{32} = 0.94 & \\ C_{31} = \$3 & C_{32} = \$2. & \end{array}$$

Given $B = 6$ the problem can be formulated as follows:

$$\max (0.9X_{11} + 0.8X_{12} + 0.85X_{13}) \cdot (0.95X_{21} + 0.8X_{22} + 0.7X_{23})(0.98X_{31} + 0.94X_{32})$$

subject to

$$\begin{aligned} X_{11} + X_{12} + X_{13} &= 1 \\ X_{21} + X_{22} + X_{23} &= 1 \\ X_{31} + X_{32} &= 1 \end{aligned}$$

$$3X_{11} + X_{12} + 2X_{13} + 3X_{21} + 2X_{22} + 1X_{23} + 3X_{31} + 2X_{32} \le 6$$

where

$$X_{11}, X_{12}, X_{13}, X_{21}, \cdots, X_{32} = 0, 1.$$

The optimal solution found by our Branch and Bound method is (X_{12}, X_{21}, X_{32}) with objective function value of 0.714 and cost = \$6.

B. Model 2: Selecting the Optimal Set of Modules for One Function Software System (with Redundancy)

The second situation addressed in this paper occurs, when the software system performs a more critical function whose failure can be very severe. In such situations, software can be made fault-tolerant by keeping redundant versions for each module. It is reasonable to assume that the allocated budget, for systems performing such functions, is large enough to allow redundancy of modules. The objective in this situation is to determine the optimal set of modules, allowing redundancy, so as to maximize the reliability of the software system while remaining within the budget.

Formulation of Model 2: We now discuss the same problem described in the previous section allowing redundancy. The problem can be formulated as follows:

$$\max R = \prod_{i=1}^{n} R_i \qquad (P_2)$$

subject to

$$X_{ij} = 0, 1 \qquad i = 1, \cdots, n \qquad j = 1, \cdots, m_i$$

where

$$\sum_{j=1}^{m_i} X_{ij} \ge 1, \qquad i = 1, \cdots, n \qquad \text{(i)}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m_i} X_{ij} C_{ij} \le B \qquad \text{(ii)}$$

$$R_i = 1 - \prod_{j=1}^{m_i} (1 - R_{ij})^{X_{ij}}. \qquad (7)$$

The reliability of module i is defined as the probability that at least one of the m_i versions is performing correctly (given as one minus the probability that none of the m_i versions is performing correctly). Constraint set (i) guarantees that for each module i at least one version is selected.

Problem P_2 can be solved using Dynamic Programming [6] algorithm. A numerical example is given below and, Appendix B presents our Dynamic Programming approach. Using the same numerical example that was used for model 1, with a budget of \$10, the optimal solution is found to be $X_{31} = 1$, $X_{21} = X_{23} = 1$, $X_{12} = X_{13} = 1$ and the overall reliability of the system is 0.9359.

C. Model 3: Selecting the Optimal Set of Modules for a System with K Functions (without Redundancy)

The third model deals with software systems consisting of several programs each performing a specific function. Each program contains a series of modules. Programs can be called by their corresponding functions and modules can be called by any program. The objective of this model is to determine the optimal set of modules for the programs, not allowing redundancy, such that the reliability of the software system is maximized while remaining within the budget.

Formulation of Model 3: Let S_k denote the set of modules corresponding to program k. For each module $i \in S_k$ there are m_i versions available. We note that the same module can be called by different programs. We number all the modules to be called by all programs from 1 to n. The problem of maximizing reliability by choosing an optimal set of modules (without redundancy) can be formulated as follows:

$$\max R = \sum_{k=1}^{K} F_k \prod_{i \in S_k} R_i \qquad (P_3)$$

subject to

$$\sum_{j=1}^{m_i} X_{ij} = 1, \qquad i = 1, \cdots, n \qquad \text{(i)}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} X_{ij} C_{ij} \le B \qquad \text{(ii)}$$

$$X_{ij} = 0, 1 \qquad i = 1, \cdots, n \qquad j = 1, \cdots, m_i$$

where R_i is given by (1).

Problem P_3 can be solved by a Branch and Bound approach very similar to the one for P_1 with minor modifications. A numerical example is given below. Consider the example at the bottom of the page.

The problem can be formulated as

$$\max \begin{bmatrix} 0.70[(0.80X_{11} + 0.85X_{12})(0.70X_{21} + 0.90X_{22})] + \\ 0.30[(0.70X_{21} + 0.90X_{22})(0.95_{31} + 0.90X_{32})] \end{bmatrix}$$

subject to

$$\begin{aligned} X_{11} + X_{12} &= 1 \\ X_{21} + X_{22} &= 1 \\ X_{31} + X_{32} &= 1 \\ 2X_{11} + 3X_{12} + 1X_{12} + 3X_{22} + 4X_{31} + 3X_{32} &\le 8 \end{aligned}$$

$$X_{ij} = 0, 1 \qquad i = 1, 2, 3 \qquad j = 1, \cdots, m_i.$$

The objective function above can be rewritten as

$$\begin{aligned} \max\ & 0.392X_{11}X_{12} + 0.504X_{11}X_{22} + 0.4165X_{12}X_{21} \\ & + 0.5355X_{12}X_{22} \\ & + 0.1995X_{21}X_{31} + 0.189X_{21}X_{32} \\ & + 0.2565X_{22}X_{31} + 0.243X_{22}X_{32}. \end{aligned}$$

The optimal solution is (X_{11}, X_{22}, X_{32}) which costs \$8 and has an optimal objective function value of 0.747. Observe that when $B = \$10$ the optimal solution is (X_{12}, X_{22}, X_{31}) with an objective function value of 0.792.

D. Model 4. Selecting the Optimal Set of Modules for a System with K Functions (with Redundancy)

The problem we discuss in this section is identical to P_3 except that redundancy is now permitted, i.e., we allow the choice of more than one version for each one of the modules. The problem can be formulated as follows:

$$\max R = \sum_{k=1}^{K} F_k \prod_{i \in S_k} R_i \qquad (P_4)$$

subject to

$$\sum_{j=1}^{m_i} X_{ij} \ge 1, \qquad i = 1, \cdots, n \qquad \text{(i)}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} X_{ij} C_{ij} \le B \qquad \text{(ii)}$$

$$X_{ij} = 0, 1 \qquad i = 1, \cdots, n \qquad j = 1, \cdots, m_i$$

where R_i is given by (7). Because of the set of constraints (i) and since we deal with $K > 1$ functions, none of the methods discussed so far in this paper can be used to solve problem P_4. Moreover, because the objective function is nonlinear we cannot solve the problem directly as an integer programming problem. In Appendix C we show how the problem can be solved using integer programming. Using the same numerical example that was used for model 3, we obtain the results that are given below.

After transforming the problem to Integer Programming, the problem is solved using Lindo on an IBM PC. The optimal solution for a budget of \$9 is $X_{11} = X_{21} = X_{22} = X_{32} = 1$ and the optimal objective function value is 0.8052, which is slightly better than 0.792 the objective function value of P_3 for a larger budget of \$10.

III. Conclusions and Discussions

This paper presents optimization models for software systems that are developed using modular design technique. Four different software structures are considered: 1) one program, no redundancy; 2) one program, with redundancy; 3) multiple programs, no redundancy; 4) multiple programs, with redundancy. The optimization problems are solved by using our version of established optimization methods. The practical usefulness of this study is to draw the attention of software practitioners to an existing methodology which may be used to make an optimal selection out of an available pool of modules with known reliability and cost. All four models maximize software reliability while ensuring that expenditures remain within available resources. The software manager is allowed to select the appropriate model for a given situation.

Finally, we would like to comment on two assumptions made throughout the paper. First, it is assumed that it is known whether or not a function is performing in satisfactory manner. This assumption is not restrictive since the models can be basically modified to include an auxiliary program (with its known reliability and cost). This auxiliary program can be used to determine whether or not a function is operating in a satisfactory manner. Second, for the models with redundancy it is assumed that there is statistical independence among

---

$K = 2$,		$F_1 = 0.70$,	$F_2 = 0.30$,	$n = 3$,	$B = \$8$
$s_1 = \{1, 2\}$,	$s_2 = \{2, 3\}$,	$m_1 = 2$,	$m_2 = 2$,	$m_3 = 2$	
$R_{11} = 0.80$	$R_{12} = 0.85$	$R_{21} = 0.70$	$R_{22} = 0.90$	$R_{31} = 0.95$	$R_{32} = 0.90$
$C_{11} = \$2$	$C_{12} = \$3$	$C_{21} = \$1$	$C_{22} = \$3$	$C_{31} = \$4$	$C_{12} = \$3$.

different program versions (by using, for example, COST modules). This is indeed a strong assumption that needs to be used with caution.

APPENDIX A

First we note that problem P_1 is feasible if

$$\sum_{i=1}^{n}\left(\min_j C_{ij}\right) \le B. \tag{2}$$

Let X_i be the selected module for task i for $i = 1, \cdots, n$ and let X_i^0 for $i = 1, \cdots, n$ be the optimal solution of P_1 when constraint (ii) is ignored. Obviously if $(X_1^0, \cdots, X_n^0)$ is a feasible solution of P_1 it must also be an optimal solution of P_1 with an objective function value of

$$\prod_{i=1}^{n}\left(\max_j R_{ij}\right). \tag{3}$$

Without any loss of generality suppose $(X_1, \cdots, X_k)$ for $k < n$ is a partial solution of P_1. An upper bound on this partial solution is given by

$$\left(\prod_{i=1}^{k} R_i\right)\left(\prod_{i=k+1}^{n} \max_j R_{ij}\right). \tag{4}$$

The left part of (4) reflects the actual contribution of the partial solution to the objective function value while the right part indicates the optimal choice ignoring the budget constraint (i.e., choosing $(X_{k+1}^0, \cdots, X_n^0)$). Obviously, $(X_1, \cdots, X_k)$ cannot lead to any possible feasible solution if

$$\sum_{i=1}^{k} C_i + \sum_{i=k+1}^{n}\left(\min_j C_{ij}\right) > B. \tag{5}$$

The upper bound (4) is attained only if

$$\sum_{i=1}^{k} C_i + \sum_{i=k+1}^{n} C_i^0 \le B \tag{6}$$

where C_i^0 is the cost corresponding to X_i^0 for $i = k+1, \cdots, n$.

The Branch and Bound procedure starts with choosing a known feasible solution to the problem that can serve as a lower bound (denoted by LB). One feasible solution (if one exists) is given by choosing the cheapest version for each module. In each level i of the decision tree we make a choice of one version for module i, $1 \le i \le n$. For each partial solution three numbers are calculated.

1) a—the upper bound value given by (4).

2) b—the lowest possible cost of any solution that include the partial solution given in the expression left to the inequality sign of (5).

3) c—the cost of solution $(X_1, X_2, \cdots, X_k, X_{k+1}^0, \cdots, X_n^0)$ given in the expression left to the inequality sign of (6).

A partial solution $X_1, X_2, \cdots, X_k$ is fathomed if either (i) $b > B$; or (ii) $a < LB$. A new solution is found whenever $c \le B$ [solution $(X_1, X_2, \cdots, X_k, X_{k+1}^0, \cdots, X_n^0)$]. A new feasible solution becomes the new incumbent solution if $a > LB$. The Branch and Bound starts by branching the root node in level 0 to m_1 branches, each corresponding to a choice of X_{1j} for $j = 1, \cdots, m_1$. The next node to branch is chosen to be the node with the largest upper bound.

APPENDIX B

Let us define the state of the system S to be the budget available and state i to reflect module i for $i = 1, \cdots, n$. Let $R_i(S)$ be the reliability of the system composed of module i, $i+1, \cdots, n$. Given that S is the budget available $(B - S)$ is the budget left for modules $1, \cdots, i-1$

The recursive formula for $R_i(S)$ when $i < n$ is

$$R_i(S) = \max\left(\left[1 - \prod_{j=1}^{m_i}(1 - R_{ij})^{x_{ij}}\right] \cdot R_{i+1}\left(B - \sum_{j=1}^{m_i} C_{ij}X_{ij}\right)\right) \tag{7}$$

where the maximization takes place for X_{ij} values for which

$$\sum_{j=1}^{m_i} X_{ij} \ge 1 \quad \text{and} \quad \sum_{j=1}^{m_i} C_{ij}X_{ij} \le S. \tag{8}$$

The recursive formula for $R_n(S)$ is

$$R_n(S) = \max\left[1 - \prod_{j=1}^{m_i}(1 - R_{nj})^{X_{nj}}\right] \tag{9}$$

where the maximization takes place over X_{nj} values such that

$$\sum_{j=1}^{m_i} X_{nj} \ge 1 \quad \text{and} \quad \sum_{j=1}^{m_i} C_{nj}X_{nj} \le S.$$

Given state i and state S, $R_i(S)$ should be calculated for all S in the range

$$\left[\sum_{k=i}^{n} \min_j C_{kj}, \cdots, B - \sum_{k=1}^{i-1} \min_j C_{kj}\right]. \tag{10}$$

APPENDIX C

Now we show how to rewrite the objective function as a linear function. Note that $(1 - R_{ij})^{X_{ij}}$ in (7) can be written as

$$1 - X_{ij}R_{ij} \qquad X_{ij} = 0, 1 \tag{11}$$

since, if $X_{ij} = 0$ then $(1 - R_{ij})^{X_{ij}} = 1$ and if $X_{ij} = 1$ then $(1 - R_{ij}) = 1 - R_{ij}$. Therefore, (7) can be rewritten as

$$R_i = 1 - \prod_{j=1}^{m_i}(1 - X_{ij}R_{ij}) \tag{12}$$

and the objective function of P_4 can be expressed as

$$\max \sum_{k=1}^{K} F_k \prod_{i \in S_k}\left[1 - \prod_{j=1}^{m_i}(1 - R_{ij}X_{ij})\right]. \tag{13}$$

The objective function (13) is still not linear since it includes product of binary variables. However, using [6] to express a product of n binary variables Z_i as

$$\prod_{i=1}^{n} Z_i, \qquad Z_i = 0, 1$$

we define

$$y = \prod_{i=1}^{n} Z_i, \qquad y = 0, 1 \tag{14}$$

by the two linear functions

$$Z_1 + Z_2 + \cdots + Z_n - y \le n - 1 \tag{15}$$

$$\frac{1}{n}Z_1 + \frac{1}{n}Z_2 + \cdots + \frac{1}{n}Z_n - y \ge 0. \tag{16}$$

REFERENCES

[1] N. Ashrafi and O. Berman, "Optimal design of large software systems considering reliability and cost," submitted to *IEEE Trans. Reliability*, vol. 41, no. 2, pp. 281–287, 1992.

[2] A. Avizienis and J. P. Kelly, "Fault tolerance by design diversity: Concepts and experiments," *Computer*, vol. 17, no. 8, pp. 67–80, 1984.

[3] F. Belli and P. Jedrzejowicz, "An approach to reliability optimization of software with redundancy," *IEEE Trans. Software Eng.*, vol. 17, no. 3, pp. 310–312, 1991.

[4] B. W. Boehm, *Software Risk Management* (Tutorial). New York: IEEE Computer Society Press, 1989.

[5] J. D. Musa, "A theory of software reliability and its application," *IEEE Trans. Software Eng.*, vol. SE-1, no. 3, pp. 312–327, 1975.

[6] H. A. Taha, *Operations Research: An Introduction.* New York: Macmillan, 1976.

[7] H. M. Wagner, *Principles of Operations Research.* Englewood Cliffs, NJ: Prentice-Hall, 1975.