Joint Optimization of ICD and Reliability for Component Selection in Designing Modules of the Software System Incorporating "Build-or-Buy" Scheme

Indumati[1], P. C. Jha[2] and U. Dinesh Kumar[3]

[1 2]*Department of Operational Research, University of Delhi, Delhi, India*

[3]*Indian Institute of Management, Bangalore*

{indumati.singh@gmail.com, jhapc@yahoo.com, udkumar@gmail.com}

Abstract- **Designing software with high quality attributes which meets functional requirements is quite a consolidated activity. In this paper, we have introduced an optimization framework which supports the decision whether to buy software components or to build them in-house. In this paper, the formulation of an optimization model of software component selection for software development is described along with build or buy decision approach while selection components for the CBSS. The model has two objectives: maximizing reliability of CBSS and maximizing cohesion and minimizing coupling of software module while keeping satisfactory values of quality attributes with redundancy allowed for the selection of the components; i.e. we can select more than one program for each functional requirement to fulfill its objectivity with respect to components. An optimization model, for optimal selection of components has been proposed and is illustrated with numerical example.**

Keywords—**Intra-modular coupling density (ICD), Coupling, Cohesion, CBSS (Components Based Software System), Build-or-Buy**

1. INTRODUCTION

In [8], authors described architecture-based software reliability analysis, which is part of design phase of the software development life cycle and has gained prominence in the recent years to cope with the growing complexity of software applications and the stringent reliability expectations imposed on. Quantitative measures are highly valuable in the early design phases when there is a significant latitude exists for reliability improvement, it is necessary that these measures are accurate. As reliability is one of the most important quality attributes of software, so it is most important to predict this measure quantitatively. The reliability for software is defined as probability that software operates without failure in a specified environment, during a specified exposure period. Software reliability aims at reducing / eliminating failures in the software systems. Reliability in software system is typically measured during or after system implementation. In this paper we have estimated reliability and have optimized it for the considered modular CBSS. In the architecture-based approach, system reliability is expressed in terms of the failure behaviour of its components and modules. Failure is defined as a discrepancy between expected and actual output. An important advantage of this approach is that it can be applied earlier in the life cycle, to identify potential problems and to enable decision makers to make more responsive choices sooner. This should subsequently contribute to effective management of an organization's technical and economic resources. In developing software systems, the manager's goal is to produce a software system within limited resources and in accordance with user requirements. When the design of software architecture reaches a good level of maturity, software engineers have to undertake selection decision about software components. COTS (Commercial off-the-shelf) have deeply changed the approach to software design and implementation. Benefits of COTS based development include significant reduction in the development cost, time and improvement in the dependability requirement. COTS components are used without any code modification and inspection .The components, which are not available in the market or cannot be purchased economically, can be developed within the organization. Respective developers of the components provide information about their quality normally in terms of reliability. COTS components are received from the distributor and are used 'as is'. No changes are normally made to their source codes. Only the code that is necessary to integrate these products is required to develop in house. Large software systems have

modular structures. The advancement of technology has made the use of COTS products as modules a possibility. A component can now be chosen for a module from the number of alternatives available in the market.

Some previous studies attempted to define criteria and develop metrics for software modularity. In the development of modular based conventional software systems, the criteria of minimizing the coupling and maximizing the cohesion of software modules were commonly used in ([1]; [12]). Coupling is about the measure of interactions among software modules while cohesion is about the measure of interactions among the software components which are within a software module. A good software system should possess software modules with high cohesion and low coupling. In Seker et al. [15], a highly cohesive module exhibits high reusability and loosely coupled systems enable easy maintenance of a software system. The software quality metrics of coupling and cohesion were proposed by authors given in [16], based on characteristics of ‘‘good” programming practices that aimed at reducing maintenance and modification costs. In [5], authors considered class-function factors as one of the drivers in the software decomposition. In [12] authors discussed the criteria used in dividing a software system into modules, and evaluate their proposed criteria in the conventional and unconventional decomposition. Authors in [13] presented a new set of metrics for analyzing the interaction between the modules of a large software system & in [14] later presented a new set of metrics for assessing the quality of software modularity. According to [1] proposed a quantitative approach to measure and assess the solutions of software modularity based on the criteria of minimal coupling and maximal cohesion. Large software system has modular structure to perform set of functions with different modules having different components for each module. Kwong et al., [11] have addressed this concept of optimal selection of software components for software modules in CBSS development very nicely, they have used heuristic technique of Genetic Algorithum for optimization purpose.

In the last few years many efforts have been spent to support the design of software architecture. According to [2], selecting the appropriate set of components and connectors to make the system meeting functional and non-functional requirements remains a hard task to accomplish and it very often depends on the architects' experience. Some relevant progresses have been made in building software that meets their functional requirements [9]. As opposite, limited contributions have been brought to support the selection of components on the basis of their non – functional characteristics (such as performance and reliability). As a consequence, software developers have no automated tools to support the analysis “aimed at characterizing the performance and reliability behaviour of software applications based on the behaviour of the “components” and the “architecture” of the application” [10]. Beside all the above considerations, real software projects ever more suffer from limited budgets, and the decisions taken from software developers are heavily affected by cost issues. The best decision might not be feasible due to high cost, and wrong cost estimations may have a critical impact for the project success. Therefore, tools that support decisions strictly related to meet functional and non – functional requirements, while keeping cost within a predicted budget, would be very helpful to the software developer's tasks. In [6], authors have introduced a framework that helps developers to decide whether to buy-or- build components of certain software architecture. In software architecture, each component can be either purchased and probably adapted to the new software system, and/or it can be developed in-house. This is a “build-or-buy” decision that affects the software cost as well as the ability of the system to meet its requirements.

In developing a software system, different functions are required to be performed and different modules are available for those functions. To provide maximum reliable software, one should give emphasis on reliability of modules. To access the module reliability of software, one should have information on module reliability. In this paper, the reliability of in-house and COTS based modular software is devised with possible redundancy at the module level with simultaneous impact of the same on budget as well. The frequency with which the functions are used is not same for all of them and not all the modules are called during the execution of a function. The objective functions in the model perform weighted maximization of system quality which we have assumed to be a weighted function of module reliabilities, weights are decided as per access frequency of each module. We assume that for all the components available for a module, cost increases if higher reliability is desired. Several components of a software module may be available for selection as COTS, all almost equivalent from the functional viewpoint. Purchase of high quality COTS products can be justified by the frequent use of a module. Basically, the alternatives/components differ each other for costs and non-functional properties. Also the best of amount testing effort is required to improve the reliability of the in-house build component which leads to an increase in cost. Software reliability assessment and prediction techniques are of utmost importance to quantify the performance of a software system. Here we have introduced reliability of the modules and system and considered it to be one of the objective to be attained

and as constraints per module as well. In this paper, we have incorporated build-or-buy concept of [6] for optimal selection of software components for software modules in CBSS development given in Kwong et al., [11]. The optimal selection problem is done solved weighted maximization of system quality/reliability along with simultaneously maximizing ICD (ICD is incorporating impact of maximizing cohesion and minimizing coupling) subject to total budget constraint, total delivery time constraint & ICD (each module constraint has threshold) with one more system constraint of functional requirement, we have allowed redundancy for the selection process. We have solved this more complex Integer Non-Linear Programming Problem directly using approach given in [4] with Lingo software (Version11). The optimization model used here has a quadratic binary type fractional objective and constraint (ICD) to perform the optimal selection of software components for CBSS development. The rest of the paper is organized as follows: Section 2 describes the formulation of an optimization model for the selection of software components for CBSS development. Section 3 describes optimization model. An illustrative example and a discussion of the results are presented in Section 4. Finally, conclusions and further works are described in Section 5.

2. FORMULATION OF OPTIMIZATION MODEL OF SOFTWARE COMPONENTS FOR CBSS

Software modules should be firstly identified for developing a CBSS and each module at least contains one software component. Software components are concrete software products that contain executable program codes and could be provided by various third-party software components providers and/or may be produced in-house. Figure 1 shows how a CBSS can be developed using software components (Kwong et al., [11]).

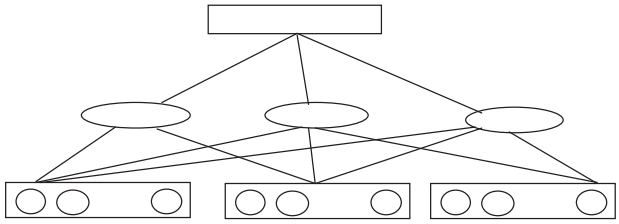
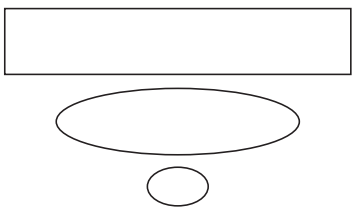


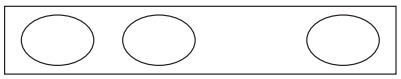
FIGURE 1
FORMULATION OF A CBSS



Component Based Software System

Software Module

Software Component



Set of Alternative Components

Generally, a CBSS is developed based on a top-down approach. Based on the approach, functional/customer requirements are first defined. The number and nature of software modules are then determined. The next task is to select software components to formulate the modules. The software components should be selected such that the interactions of the software components within a software module are maximized, and interactions of the software components among software modules are minimized. Let S be a software architecture made of M modules, with a maximum number of N components (build-or buy) available for each module. In general, a "build-or-buy" decisional strategy can be described as a set of 0-1 variables.

2.1 NOTATIONS:

M	the number of software modules
N	the number of software components
L	the number of sets of software components
w_l	the frequency of use, of function l
Q_l	set of modules required to perform l^{th} function
Sc_i	i=1,2,….N, the i^{th} software component
Sc_{ix}	i=1,2,….N, the i^{th} software COTS component
Sc_{iy}	i=1,2,….N, the i^{th} software in-house component
m_j	j=1,2,….M, the j^{th} software module
S_k	k=1,2,….L, a set of alternative components for the k^{th} functional requirements of a CBSS
$i \in S_k$	denotes that S_{Ci} belongs to the k^{th} set
Sc_{ij}	the i^{th} software component of j^{th} software module, s.t. $Sc_{ij} = Sc_{ij'} = Sc_i$ = for all j, j'=1,2,….M
$r_{ii'}$	i,i' =1,2,….N, the number of interactions between S_{Ci} & $S_{Ci'}$ s.t., $r_{ii'=} r_{i'i.}$
$R_{ij,}$	i=1,2,….N, j=1,2,….M, reliability of i^{th} component available for j^{th} module for COTS product ; $0 \le R_{ij} \le 1 ; \forall i, j$
ρ_{ij}	i=1,2,….N, j=1,2,….M, reliability of i^{th} component of j^{th} module for in-house product ; $0 \le \rho_{ij} \le 1 ; \forall i, j$
C_{ij}	i=1,2,….N, j=1,2,….M, cost of i^{th} component available for j^{th} module for COTS product
cij	i=1,2,….N, j=1,2,….M, unitary development cost for i^{th} component of j^{th} module for in-house
S_{ij}	i=1,2,….N, j=1,2,….M, reliability of i^{th} component available for j^{th} module ; $0 \le S_{ij} \le 1 ; \forall i, j$
π_{ij}	probability that a single execution of software fails on a test case chosen from a certain input distribution while testability of i^{th} component of j^{th} module for in-house product & is assumed to be $\pi_{ij} = 0.002 \ \forall i, j$

t_{ij} estimated development time of i^{th} in-house component of j^{th} module

d_{ij} delivery time of i^{th} COTS component available for j^{th} module

τ_{ij} average time required to perform test case is assumed to be $\tau_{ij} = 0.05 \; \forall i, j$

N_{ij}^{suc} number of successful tests performed on the i^{th} component of j^{th} module for in-house product

N_{ij}^{tot} total number of tests performed on the i^{th} component of j^{th} module for in-house product

R_j reliability of j^{th} software module ; $0 \le R_j \le 1 ; \forall j$

$R_{j'}$ minimum reliability level required to attain for j^{th} software module ; $0 \le R_{j'} \le 1; \forall j'$

H a threshold value of ICD_j of each module that needs to be set by decision makers

C maximum budget limit set by the decision makers

T maximum threshold given on delivery time of the whole system

x_{ij} i=1,2,..,N, j=1,2,..,.M, x_{ij} is the binary variable ; x_{ij} =1, if S_{Cix} is selected for m_j for buy product; otherwise 0

y_{ij} i=1,2,..,N, j=1,2,..,M, y_{ij} is the binary variable ; y_{ij} =1, if S_{Ciy} is selected for m_j for in-house product; otherwise 0

z_{ij} i=1,2,….N, j=1,2,….M, z_{ij} is the binary variable ; z_{ij} =1, if S_{Ci} is selected for m_j for either buy S_{Cix} or in-house product S_{Ciy} ; otherwise 0

2.2 ASSUMPTIONS:

1. Atleast one component is supposed to get selected from each module either COTS or in-house, redundancy is allowed for both the build or buy components.
2. A threshold value of $ICD_{j,}$ budget, delivery time and reliability are set by the decision makers.
3. Functional requirements – More than one software component in S_k either in-house or buy or combination of both) may get selected to implement k^{th} requirement.
4. The cost of an alternative is the development cost, if developed in house; otherwise it is the buying price for the COTS product. Reliability data set is given for COTS components are known.
5. The Cost and reliability of an in-house component can be specified by using basic parameters of the development process, e.g. a component cost may depend on a measure of developer skills, or the component reliability depends on the amount of testing.
6. Interaction data for components is exactly same for all modules irrespective of the selection happened for COTS or in-house component.
7. Different COTS alternatives with respect to cost, reliability and delivery time of a module are available.
8. Different in-house alternatives with respect to unitary development cost, estimated development time, average time and testability of a module are available.

2.3 MODEL DESCRIPTION

2.3.1. TESTABILITY CONDITIONS

The effect of testing on cost, reliability and delivery time of COTS products is instead assumed to be accounted in the COTS parameters. Basing on the testability definition, we can assume that the number N_{ij}^{suc} of successful (i.e. failure-free) tests performed on the same component can be obtained as:

$$N_{ij}^{suc} = (1 - \pi_{ij}) N_{ij}^{tot} \qquad i = 1, 2, ..., N ; j, j' = 1, 2, ..., M \quad (1)$$

2.3.1.1 BUILD VERSUS BUY DECISION

If i^{th} component of j^{th} module is bought (i.e. some x_{ij}=1) then there will be no in-house development (i.e. y_{ij}=0) and vice versa.

$$x_{ij} + y_{ij} \le 1 \quad i = 1, 2, ..., N ; j = 1, 2, ..., M \quad (2)$$

2.3.1.2 REDUNDANCY

The equation stated below guarantees that redundancy is allowed for both the build and buy components (i.e. in-house and COTS components).

$$z_{ij} = x_{ij} + y_{ij} \quad i = 1, 2, ..., N ; j = 1, 2, ..., M \quad (3)$$

$$\sum_{i=1}^{N} z_{ij} \ge 1 \quad j = 1, 2, ..., M \quad (4)$$

2.3.2. COST CONSTRAINT

Cost constraint represents the overall cost of the system. The sum of the cost of all the modules is selected from "build - or - buy" strategy. The in-house development cost of the i^{th} component available for j^{th} module can be expressed as $c_{ij}(t_{ij} + \tau_{ij} N_{ij}^{tot})$. So over all cost constraint can be expressed as:

$$\sum_{j=1}^{M} [\sum_{i=1}^{N} c_{ij}(t_{ij} + \tau_{ij} N_{ij}^{tot}) y_{ij} + (\sum_{i=1}^{N} C_{ij} x_{ij})] \le C \quad (5)$$

2.3.3. DELIVERY TIME CONSTRAINT

The maximum threshold T has been given on the delivery time of the whole system. In case of a COTS components the delivery time is simply given by d_{ij}, whereas for an in-house developed component the delivery time of i^{th} component available for j^{th} module is $(t_{ij} + \tau_{ij} N_{ij}^{tot})$ .So overall delivery time can be expressed as:

$$\sum_{j=1}^{M} [\sum_{i=1}^{N} (t_{ij} + \tau_{ij} N_{ij}^{tot}) y_{ij} + (\sum_{i=1}^{N} d_{ij} x_{ij})] \le T \quad (6)$$

2.3.4. RELIABILITY CONSTRAINT

Reliability of an in-house developed component

Authors in [6] have defined the probability of failure on demand of an in-house developed, the i^{th} component of j^{th} module under the assumption that the on-field users' operational profile is the same as the one adopted for testing in [3]. Let A be the event " N_{ij}^{suc} failure – free test cases have been performed " and B be the event " the alternative is failure free during a single run ".If ρ_{ij} is the probability that the in- house developed alternative is failure free during a single run given that N_{ij}^{suc} test cases have been successfully performed, from the Bayes Theorem we get

$$\rho_{ij} = P(B/A) = \frac{P(A/B)P(B)}{P(A/B)P(B) + P(A/\bar{B})P(\bar{B})}$$

The following equalities come straightforwardly:

$$P(A/B) = 1$$

$$P(B) = 1 - \pi_{ij}$$

$$P(A/\bar{B}) = (1-\pi_{ij})^{N_{ij}^{suc}}$$

$$P(\bar{B}) = \pi_{ij}$$

therefore, we have

$$\rho_{ij} = \frac{1-\pi_{ij}}{\left(1-\pi_{ij}\right) + \pi_{ij}\left(1-\pi_{ij}\right)^{N_{ij}^{suc}}} \quad ; \quad 0 \le \rho_{ij} \le 1 \; ; \; i = 1,2,...,N \; ; \; j = 1,2,...,M \quad (7)$$

Reliability equation of both in-house and COTS components

$$S_{ij} = \rho_{ij} y_{ij} + R_{ij} x_{ij} \quad ; \quad 0 \le \rho_{ij}, R_{ij}, S_{ij} \le 1; \; i = 1,2,...,N; \; j = 1,2,...,M \quad (8)$$

2.3.5. RELIABILITY OBJECTIVE FUNCTION / CONSTRAINT

Reliability objective function maximizes the system quality (in terms of reliability) through a weighted function of module reliabilities. Reliability of modules that are invoked more frequently during use is given higher weights. Analytic Hierarchy Process (AHP) can be effectively used to calculate these weights. So total reliability $(0 \le R \le 1)$ can be expressed as:

$$R = \sum_{l=1}^{L} w_l \prod_{j \in Q_l} R_j$$

$$where \quad 0 \le \left\{ R_j = \prod_{i \in M_j} [\rho_{ij}]^{y_{ij}} [R_{ij}]^{x_{ij}} \right\} \le 1 \; ; \; i = 1,2,...,N; j = 1,2,...,M \quad (9)$$

The minimum threshold $R_{j'}$ has been given on reliability constraint for each module & can be expressed as:

$$R_j = \prod_{i \in M_j} [\rho_{ij}]^{y_{ij}} [R_{ij}]^{x_{ij}} \ge R_{j'} \quad ; \; i = 1,2,...,N \; ; \; j, j' = 1,2,...,M \; ; \; ; \; 0 \le R_{j'} \le 1; \forall j' \quad (10)$$

2.3.6. COHESION, COUPLING & ICD OBJECTIVE FUNCTION / CONSTRAINT (AS PER "BUILD OR BUY" STRATEGY)

The cohesion ([1]; Kwong et al., [11]) incorporating both in-house and COTS components impact [6] within the j^{th} module $(CI_{IN})_j$ can be devised as follows:

$$(CI_{IN})_j = \sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} z_{ij} z_{i'j} \quad (11)$$

All interactions including cohesion and coupling associated with the j^{th} module, CA_j, can be expressed as:

$$CA_j = (CI_{IN})_j + (CI_{OUT})_j = \sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} z_{ij} (\sum_{j'=1}^{M} z_{i'j'}) \quad (12)$$

All interactions including cohesion and coupling of a software system, CA_j can be expressed as shown below:

$$CA = \sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} (\sum_{j=1}^{M} z_{ij}) (\sum_{j=1}^{M} z_{i'j}) \quad (13)$$

The sum of cohesions within all modules CI_{IN} can be expressed as shown below:

$$CA = \sum_{j=1}^{M} \sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} z_{ij} z_{i'j} \quad (14)$$

According to [1] yielded the quantitative measures of cohesion and coupling. In their work, intra-modular

coupling density (ICD) was introduced to measure the relationship between cohesion and coupling of modules in the design of an object oriented software system as follows:

$$ICD = \frac{CI_{IN}}{CI_{IN} + CI_{OUT}} \tag{15}$$

where CI_{IN} is the number of class interactions within modules, and CI_{OUT} is the number of interactions between classes of distinct modules. ICD was employed as a criterion to present the ratio between coupling and cohesion in CBSS development. Referring to Equation (15), the ratio of cohesion to all interactions within the j[th] module can be expressed as ICD_j. However, it can be found if one module contains only one component, such as module1or module 4, as shown in below Figure 2; the values of ICD for modules 1 and 4 are all zero (Kwong et al., [11]).

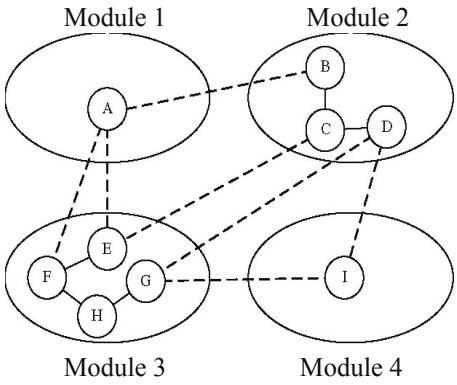


FIGURE 2
COHESION AND COUPLING OF SOFTWARE MODULES IN CBSS

To make up for the deficiency, another measure of ICD is given, by simply adding 1 to the numerator of Equation (15):

$$ICD_j = \frac{(CI_{IN})_j + 1}{(CI_{IN})_j + (CI_{OUT})_j} \tag{16}$$

where ICD_j is the intra-modular coupling density for the j[th] module; $(CI_{IN})_j$ is the number of component interactions within the j[th] module; and $(CI_{OUT})_j$ is the number of component interactions between the j[th] module and other modules. It is widely recognized that loose coupling and tight cohesion can achieve high maintainability of a software system. Therefore, the values of ICD would have great influence on the maintainability of a CBSS for improving system quality and can be expressed as:

$$ICD = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N-1}\sum_{i'=i+1}^{N} r_{ii'} z_{ij} z_{i'j}}{\sum_{i=1}^{N-1}\sum_{i'=i+1}^{N} r_{ii'} \left(\sum_{j=1}^{M} z_{ij}\right)\left(\sum_{j=1}^{M} z_{i'j}\right)} \quad ; \quad 0 \le ICD \le 1 \tag{17}$$

3. OPTIMIZATION MODEL

Optimization model based on the criteria identified above can be formulated as below in problem (P1):

$$Max\ ICD = \frac{\sum_{j=1}^{M}\sum_{i=1}^{N-1}\sum_{i'=i+1}^{N} r_{ii'} z_{ij} z_{i'j}}{\sum_{i=1}^{N-1}\sum_{i'=i+1}^{N} r_{ii'} \left(\sum_{j=1}^{M} z_{ij}\right)\left(\sum_{j=1}^{M} z_{i'j}\right)} \tag{18}$$

$$Max\ R = \sum_{l=1}^{L} w_l \prod_{j \in Q_l} R_j \quad where \quad 0 \le \left\{ R_j = \prod_{i \in M_j} [\rho_{ij}]^{y_{ij}} [R_{ij}]^{x_{ij}} \right\} \le 1 \ ; \ i = 1,2,...,N; j = 1,2,...,M \tag{19}$$

$$S_{ij} = \rho_{ij} y_{ij} + R_{ij} x_{ij} \quad i = 1,2,.....,N \ ; \ j = 1,2,.....,M$$

subject to $S = \{ z_{ij}\ binary\ \text{var}iable \in \{0,1\} \ ;\ i = 1,..,N;\ j = 1,..,M$

$$\frac{\sum_{j=1}^{M}\sum_{i=1}^{N-1}\sum_{i'=i+1}^{N} r_{ii'} z_{ij}\, z_{i'j} + 1}{\sum_{i=1}^{N-1}\sum_{i'=i+1}^{N} r_{ii'} z_{ij} \left(\sum_{j'=1}^{M} z_{i'j'}\right)} \ge H \quad j, j' = 1,2,.....,M \tag{20}$$

$$\prod_{i \in M_j} [\rho_{ij}]^{y_{ij}} [R_{ij}]^{x_{ij}} \ge R_{j'} \quad j, j' = 1,2,.....,M \tag{21}$$

$$N_{ij}^{suc} = (1 - \pi_{ij}) N_{ij}^{tot} \quad i = 1,2,.....,N; j, j' = 1,2,.....,M \tag{22}$$

$$\rho_{ij} = \frac{1 - \pi_{ij}}{\left(1 - \pi_{ij}\right) + \pi_{ij} \left(1 - \pi_{ij}\right)^{N_{ij}^{suc}}} \quad i = 1,2,....,N; j = 1,2,....,M \tag{23}$$

$$\sum_{j=1}^{M} [\sum_{i=1}^{N} c_{ij} (t_{ij} + \tau_{ij} N_{ij}^{tot}) y_{ij} + (\sum_{i=1}^{N} C_{ij} x_{ij})] \le C \tag{24}$$

$$\sum_{j=1}^{M} [\sum_{i=1}^{N} (t_{ij} + \tau_{ij} N_{ij}^{tot}) y_{ij} + (\sum_{i=1}^{N} d_{ij} x_{ij})] \le T \tag{25}$$

$$\sum_{i \in S_k} \sum_{j=1}^{M} z_{ij} \ge 1 \tag{26}$$

$$z_{ij} = x_{ij} + y_{ij} \quad i = 1,2,.....,N; j = 1,2,.....,M \tag{27}$$

$$x_{ij} + y_{ij} \le 1 \quad i = 1,2,.....,N; j = 1,2,.....,M \tag{28}$$

$$\sum_{i=1}^{N} z_{ij} \ge 1 \quad j = 1,2,.....,M \ \Big\} \tag{29}$$

4. NUMERICAL ILLUSTRATION

A numerical example is illustrated to illustrate the proposed methodology of optimizing the selection of software components for CBSS development in Table 1. In the example a software system is decomposed into three modules, m_1, m_2 and m_3. Ten software components (Sc_{1x}–Sc_{10x}) that are available in market & (Sc_{1y}–Sc_{10y}) that can be build in-house to make up four sets of alternative software component (S_1–S_4) for each module are considered. Total components available for selection are twenty and may be represented by (Sc_1–Sc_{10}). Say for example a local software system supplier planned to develop a financial system for small and medium-size enterprises. Four functional requirements of the system were identified, namely, Facsimile/Fax, (R'1), Encryption (R'2), Credit Card Authorization (R'3), Ecommerce/Accounts (R'4). The software system development team of the company has defined three software modules, business related module (m_1), security module (m_2) and assistance module (m_3), that the financial software system needs to contain. The business-related module mainly provides the functions of e-commerce and financial reporting. The security module mainly provides the functions of encryption and authorization while the assistance module is to provide auxiliary functions for the system such as fax and software automatic updating.

TABLE 1
EXAMPLE DESCRIPTION – FINANCIAL SYSTEM DESIGN (Kwong et al., [11])

Functional requirements	S_k	Software components	
Facsimile/Fax,R'1	S_1	Sc_{1x}, $Sc_{1y} = Sc_1$	OpalVOIP
		Sc_{2x}, $Sc_{2y} = Sc_2$	HylaFAX
		Sc_{3x}, $Sc_{3y} = Sc_3$	Faxman
		Sc_{4x}, $Sc_{4y} = Sc_4$	HylaPEx
Encryption, R'2	S_2	Sc_{5x}, $Sc_{5y} = Sc_5$	CryptoXpress CF
Credit Card Authorization, R'3	S_3	Sc_{6x}, $Sc_{6y} = Sc_6$	IBiz E
eCommerce/Account, R'4	S_4	Sc_{7x}, $Sc_{7y} = Sc_7$	Symphero
		Sc_{8x}, $Sc_{8y} = Sc_8$	Shopformat
		Sc_{9x}, $Sc_{9y} = Sc_9$	Account manager
		Sc_{10x}, $Sc_{10y} = Sc_{10}$	Account services manager

Table 2 shows the degrees of interaction among software components, range of the degrees is 1–10 where degree '1' means a very low degree of interaction while degree '10' refers to a very high degree of interaction. Table 3 shows reliability data set $(0 \le R_{ij} \le 1 ; \forall i, j)$; associated with COTS components. In Table 4 cost (in 100$ unit) associated with COTS components is given.

TABLE 2
INTERACTIONS AMONG SOFTWARE COTS & IN-HOUSE COMPONENTS

		S_1				S_2	S_3	S_4			
		Sc_1	Sc_2	Sc_3	Sc_4	Sc_5	Sc_6	Sc_7	Sc_8	Sc_9	Sc_{10}
S_1	Sc_1	0	0	5	10	0	0	0	0	4	10
	Sc_2	0	0	0	8	6	6	2	3	0	10
	Sc_3	5	0	0	4	0	2	6	3	6	3
	Sc_4	10	8	4	0	0	9	0	5	6	4
S_2	Sc_5	0	6	0	0	0	8	0	7	0	4
S_3	Sc_6	0	6	2	9	8	0	0	0	0	0
S_4	Sc_7	0	2	6	0	0	0	0	7	0	0
	Sc_8	0	3	3	5	7	0	7	0	10	8
	Sc_9	4	0	6	6	0	0	0	10	0	0
	Sc_{10}	10	10	3	4	4	0	0	8	0	0

TABLE 3
RELIABILITY DATA SET (COTS)

Reliability (associated with COTS components)					
Module1	m_1	Module2	m_2	Module3	m_3
R_{11}	0.95	R_{12}	0.99	R_{13}	0.98
R_{21}	0.98	R_{22}	0.98	R_{23}	0.91
R_{31}	0.92	R_{32}	0.99	R_{33}	0.94
R_{41}	0.95	R_{42}	0.97	R_{43}	0.96
R_{51}	0.96	R_{52}	0.96	R_{53}	0.9
R_{61}	0.95	R_{62}	0.97	R_{63}	0.95
R_{71}	0.96	R_{72}	0.96	R_{73}	0.9
R_{81}	0.98	R_{82}	0.9	R_{83}	0.95
R_{91}	0.95	R_{92}	0.96	R_{93}	0.96
R_{101}	0.96	R_{102}	0.9	R_{103}	0.97

TABLE 4
COST DATA SET (COTS)

Cost (associated with COTS components)					
Module1	m_1	Module2	m_2	Module3	m_3
C_{11}	10	C_{12}	7	C_{13}	6
C_{21}	6	C_{22}	8	C_{23}	7
C_{31}	8	C_{32}	9	C_{33}	8
C_{41}	9	C_{42}	10	C_{43}	9
C_{51}	8	C_{52}	6	C_{53}	7
C_{61}	6	C_{62}	7	C_{63}	8
C_{71}	7	C_{72}	8	C_{73}	9
C_{81}	8	C_{82}	9	C_{83}	6
C_{91}	9	C_{92}	6	C_{93}	7
C_{101}	6	C_{102}	10	C_{103}	8

In Table 5 development cost (in 100$ unit) and Table 6 estimated time (in hours) associated with in-house components is given.

TABLE 5
DEVELOPMENT COST DATA SET (IN-HOUSE)

Development Cost (associated with in-house components)					
Module1	m_1	Module2	m_2	Module3	m_3
c_{11}	5	c_{12}	2	c_{13}	1
c_{21}	2	c_{22}	3	c_{23}	2
c_{31}	3	c_{32}	4	c_{33}	3
c_{41}	4	c_{42}	5	c_{43}	4
c_{51}	3	c_{52}	1	c_{53}	2
c_{61}	1	c_{62}	2	c_{63}	3
c_{71}	2	c_{72}	3	c_{73}	4
c_{81}	3	c_{82}	4	c_{83}	1
c_{91}	4	c_{92}	1	c_{93}	2
c_{101}	1	c_{102}	5	c_{103}	3

TABLE 6
ESTIMATED DEVELOPMENT TIME DATA SET (IN-HOUSE)

Estimated Development Time					
Module1	m_1	Module2	m_2	Module3	m_3
t_{11}	9	t_{12}	6	t_{13}	5
t_{21}	6	t_{22}	7	t_{23}	6
t_{31}	7	t_{32}	4	t_{33}	7
t_{41}	8	t_{42}	9	t_{43}	8
t_{51}	7	t_{52}	5	t_{53}	6
t_{61}	5	t_{62}	6	t_{63}	7
t_{71}	6	t_{72}	7	t_{73}	8
t_{81}	7	t_{82}	5	t_{83}	5
t_{91}	8	t_{92}	5	t_{93}	6
t_{101}	5	t_{102}	9	t_{103}	7

TABLE 7
DELIVERY TIME DATA SET (COTS)

Delivery Time					
Module1	m_1	Module2	m_2	Module3	m_3
d_{11}	1	d_{12}	4	d_{13}	5
d_{21}	5	d_{22}	3	d_{23}	4
d_{31}	3	d_{32}	2	d_{33}	3
d_{41}	2	d_{42}	1	d_{43}	2
d_{51}	3	d_{52}	5	d_{53}	4
d_{61}	5	d_{62}	4	d_{63}	3
d_{71}	4	d_{72}	3	d_{73}	2
d_{81}	3	d_{82}	2	d_{83}	5
d_{91}	2	d_{92}	5	d_{93}	4
d_{101}	5	d_{102}	1	d_{103}	3

Now here we are having two simultaneous objectives out of which one is a fractional function so there does not exist any direct method to obtain an optimal solution for such class of multi-objective problems.

Let $$f(z_{ij}) = \sum_{j=1}^{M} \sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} z_{ij}\, z_{i'j}$$

$$g(z_{ij}) = \sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} \left(\sum_{j=1}^{M} z_{ij}\right)\left(\sum_{j=1}^{M} z_{i'j}\right)$$

Problem (P1) can equivalently be written as mathematical programming problem using approach in [4]:

$$Max \quad F_1(z_{ij}) = f(z_{ij}) - g(z_{ij})$$

$$Max \quad F_2(z_{ij}) = \sum_{l=1}^{L} w_l \prod_{j \in Q_l} R_j$$

subject to

$$Z \in S = \left\{ z_{ij} = (x_{ij}, y_{ij}) \ \ binary \ \text{var} iable \in \{0,1\} \ / \ \text{Satisfying eq. (20) to (29)} ; \ i = 1,...N; j = 1,2,...,M \right\} \quad \text{(P2)}$$

Using the Geoffrion's equivalent scalarization formulation as per [7] of the problem (P2) for fixed weights $\lambda \in \Omega = \{\lambda \in R / \ \lambda \geq 0\}$ for the objective function is as follows:

$$Max\, G(z_{ij}) = \left(\lambda * \left(f(z_{ij}) - g(z_{ij}) \right) \right) + (1 - \lambda) * R$$

subject to

$$Z \in S = \left\{ z_{ij} = (x_{ij}, y_{ij}) \ \ binary \ \text{var} iable \in \{0,1\} \ / \ \text{Satisfying eq. (20) to (29)} ; \ i = 1,...N; j = 1,2,...,M \right\} \quad \text{(P3)}$$

The original problem was of a Bi-Criteria form. Using Geoffrion's scalarization the problem gets converted to a single objective problem by attaching weights to both the objectives. The normalized data set for fixed weights approach [7] for the objective function ICD is given in below Table 6.

TABLE 6
NORMALIZED INTERACTIONS DATA SET

		S_1				S_2	S_3	S_4			
		Sc_1	Sc_2	Sc_3	Sc_4	Sc_5	Sc_6	Sc_7	Sc_8	Sc_9	Sc_{10}
S_1	Sc_1	0.00	0.00	0.02	0.03	0.00	0.00	0.00	0.00	0.01	0.03
	Sc_2	0.00	0.00	0.00	0.03	0.02	0.02	0.01	0.01	0.00	0.03
	Sc_3	0.02	0.00	0.00	0.01	0.00	0.01	0.02	0.01	0.02	0.01
	Sc_4	0.03	0.03	0.01	0.00	0.00	0.03	0.00	0.02	0.02	0.01
S_2	Sc_5	0.00	0.02	0.00	0.00	0.00	0.03	0.00	0.02	0.00	0.01
S_3	Sc_6	0.00	0.02	0.01	0.03	0.03	0.00	0.00	0.00	0.00	0.00
S_4	Sc_7	0.00	0.01	0.02	0.00	0.00	0.00	0.00	0.02	0.00	0.00
	Sc_8	0.00	0.01	0.01	0.02	0.02	0.00	0.02	0.00	0.03	0.03
	Sc_9	0.01	0.00	0.02	0.02	0.00	0.00	0.00	0.03	0.00	0.00
	Sc_{10}	0.03	0.03	0.01	0.01	0.01	0.00	0.00	0.03	0.00	0.00

The optimal solution so obtained of the problem (P3) is an optimal solution for the problem (P1). In this example, $\lambda = 0.5$ is taken i.e., equal weightage is given to both the objectives. Table 8 shows number of cases of selection of software components (build and/or buy combinations) for given inputs.

TABLE 8
INPUT - OUTPUT

Results of selection of software components for modules		Module m_1	Module m_2	Module m_3	R	ICD	Objective Value
Cases	Input	Output					
Case 1	R_1'=0.99, R_2'=0.99, R_3'=0.99, C=34, T=22	Sc_{71y}	Sc_{52y}, Sc_{62y}	Sc_{13y}	0.99	0.99	0.4969070
Case 2	R_1'=0.98, R_2'=0.91, R_3'=0.92, C=32, T=21	Sc_{71y}	Sc_{52y}, Sc_{62x}	Sc_{13y}	0.97	1	0.4857899
Case 3	R_1'=0.94, R_2'=0.95, R_3'=0.99, C=33, T=20	Sc_{71x}	Sc_{62x}, Sc_{52y}	Sc_{13y}	0.94	1	0.4723057
Case 4	R1'=0.94, R2'=0.91, R3'=0.94, C=35, T=21	Sc_{71x}	Sc_{52y}, Sc_{62x}	Sc_{13x}	0.93	1	0.4650470
Case 5	R1'=0.9, R2'=0.85, R3'=0.82, C=37, T=18	Sc_{61x}, Sc_{51x}	Sc_{72x}	Sc_{13x}	0.88	0.99	0.4398458

The solution mentioned is a local optimal solution for the above mentioned Integer Non-Linear Programming Formulation for the given data set, solved using software Lingo (Version 11), solver type Branch & Bound. Here selected components which are having 'x' subscript confined to COTS components & with subscript 'y' confined to in-house components. By Table 8 we can conclude that in Case 1, we have got selected all 4 in-house components, as we have more delivery time as compared to other cases, so we are able to produce highly reliable modules, leading to highest system reliability of 0.99 along with ICD of 0.99 within comparatively low budget of 34 units. In Case 2, 3 & 4 components selected are mixed sets of build (in-house) or buy (COTS) component. In Case 1 in-house component Sc_{71y} got selected for Module 1, i.e., Module 1 is able to fulfill 1 functional requirements (S_4); similarly build components Sc_{52y}, Sc_{62y} got selected for Module 2 i.e., this Module is able to fulfill two functional requirements (S_2, S_3) & build component Sc_{13y} got selected from Module 3 i.e., for fulfilling functional requirement S_1. Similarly rest of four cases can be read. In this illustration, for fulfilling all 4 functional requirements, we are able to select one component. In Case 2, we have got selected one buy component as a result we got diminished reliability of 0.97 with maximum ICD of 1. In Case 5, we have got selected all four buy components, as a result we got lowest system reliability of 0.88 along with ICD of 0.99 in maximum budget of 37 units & minimum delivery time of 18 units only. After reading all five solutions we can observe that if more delivery time & budget availability is permissible then preference should be given to create in-house components, as in-house component is more reliable in comparison to COTS component in which we have more risk of low quality components at higher cost. Production of more in-house components is giving higher objective value to the model. Hence, an optimization model for optimal selection of software components using build-or-buy strategy for CBSS development is established.

5. DISCUSSION AND CONCLUSION

In this paper, a methodology of selecting software components with impact of whether to build-or- buy components for CBSS development is described. Based on the proposed methodology, an optimization model is formulated to perform the selection of software components for the software modules of a CBSS to obtain an optimal/near optimal solution. An example of a financial system design was used to illustrate the methodology. However, this methodology involves some subjective judgments from software development teams, such as the determination of the scores of interaction etc. Reliability is supposed to get fixed as per customer requirement. In this regard, the fuzzy set theory could be introduced to deal with the fuzziness caused by subjective judgments.

REFERENCES

[1] F. B. Abreu and M. Goulão, "Coupling & cohesion as modularization drivers: Are we being over- persuaded", in *Proceedings of the fifth European conference on software maintenance and re-engineerin.: IEEE Computer Society,* Washington, DC, USA, 2001.

[2] A. Bertolino & R. Mirandola, "CB-SPE tool, Putting component-based performance engineering into Practice", *Proceedings of 7th international symposium on component-based software engineering Conference (CASCON 97)*, Toronto, Ontario, Nov. pp. 10-13, 2004.

[3] A. Bertolino and L. Strigini, "On the use of testability measures for dependability assessment", *IEEE Transactions on Software Engineering*, Vol. 22, No. 2, pp. 97-108, 1996.

[4] D. Bhatia, N. Kumar and R.K. Bhudhiraja, "Duality theorem for non-differentiable multi-objective programs", *Indian Journal of pure and applied Mathematics,* Vol. 28, No. 8, pp. 1031-1042, 1997.

[5] C. K. Chang & S. Hua, "A new approach to module-oriented design of OO software", *In-proceedings of computer software and applications conference*, pp.29–34, 1994.

[6] V. Cortellessa, F. Marinelli and P. Potena, "An optimization framework for 'build-or-buy' decisions in software architecture", *Computers and Operations Research*, Vol. 35, pp. 3090-3106, 2008.

[7] A.M. Geoffrion, "Proper efficiency and theory of vector maximization", *Journal of Mathematical Analysis and Applications,* Vol. 22, pp. 613-630, 1968.

[8] K. Go˘seva-Popstojanova, M. Hamill and X. Wang, "Adequacy, Accuracy, Scalability & Uncertainty of Architecture-based Software Reliability: Lessons Learned from Large Empirical Case Studies", *in Proc. Intl. Symposium on Software Reliability Engineering*, Raleigh, NC, pp. 197–203, 2006.

[9] P. Inverardi and M. Tivoli, "Deadlock – free software architectures for COM/DCOM application", *Journal of Systems and Software*, 65(3), pp. 173-183, 2003.

[10] H. W. Jung and B. Choi, "Optimization models for quality and cost of modular software system", *European Journal of Operations Research*, 112, pp. 613-619, 1998.

[11] C. K. Kwong, L.F. Mu, J.F. Tang and Luo X.G., "Optimization of software components selection for component based software system development", *Computer & Industrial Engineering, Elsevier*, 2010.

[12] S. Parsa and O. Bushehrian, "A framework to investigate and evaluate genetic clustering algorithms for automatic modularization of software systems. Lecture Notes in Computer Science, pp. 699-702, 2004.

[13] S. Sarkar, A. C. Kak and N. S. Nagaraja, "Metrics for analyzing module interactions in large software systems", *in proceedings of*

the 12th Asia Pacific software engineering conference, pp. 264–271, 2005.

[14] S. Sarkar, G. M. Rama and A.C. Kak, "API-based and information theoretic metrics for measuring the quality of software modularization", *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, pp. 14–32, 2007.

[15] R. Seker, A. J. Van der Merwe, P. Kotze , M.M. Tanik and R. Paul, "Assessment of coupling & cohesion for component based software by using Shannon languages", *Journal of Integrated Design & Process Science*, Vol. 33, No. 4, pp. 33–43, 2004.

[16] W. P. Stevens, G. Myers & L. Constantine, "Structured design", *IBM Systems Journal,* Vol. 13, No. 2, pp. 115–139, 1974.

ABOUT THE AUTHORS

Indumati. She obtained her M.Sc. degrees in OR from University of Delhi, Delhi. She is currently a Research Fellow in the Department of Operational Research, University of Delhi, Delhi. Her research interests include modeling and optimization in software reliability. She is a lifetime member of the Society for Reliability Engineering, Quality and Operations Management.

Dr. P. C. Jha. He Obtained his Ph. D., M. Phil and M.Sc. degrees in Operational Research (OR) from University of Delhi. He is an Associate Professor in the Department of OR, University of Delhi. He has organized a DST sponsored national Workshop cum Conference on Mathematical Modeling Optimization and its Application and 41st Annual Convention of OR Society of India and a DST sponsored workshop on Optimization and its Applications. He has contributed a chapter in edited book Handbook of Performability Engineering, Springer Verlag. His research interests include modeling and optimization in Software Reliability, Marketing, Supply Chain Management and Optimization. He has published more than 45 research papers in Indian & International journals and edited books. He has guided master's projects, MBA and M. Phil. Dissertations and is supervising Ph.D. students in OR.

Dr. U. Dinesh Kumar. He obtained his Ph.D from IIT Bombay in 1994. He is presently working as a Professor QMIS area in IIM Bangalore. He has received Best Young Teacher Award by the Association of Indian Management Schools – 2003 and obtained Visiting Fellowship – Queensland University of Technology – 2004. He has published more than 55 research papers in refereed Journals, conferences, written a number of books and edited a number of volumes. He has conducted several in-house and management development programmes His area of expertise lies in Stochastic models (Markov and Non-Markov models), Reliability and Software Reliability, Optimization, Multi Criteria Decision Making, Six Sigma, Maintainability and Maintenance, Integrated Logistic Support, New Product Development. He is a lifetime member of Professional Societies like Operational Research Society of India, Life member of Indian Statistical Association. Member of SOLE - The International Society of Logistics, USA.