Journal of Computational Science 5 (2014) 233–242

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Optimal component selection based on cohesion & coupling for component based software system under build-or-buy scheme

P.C. Jha[a], Vikram Bali[b,*], Sonam Narula[a], Mala Kalra[c]

[a] Department of Operational Research, University of Delhi, India
[b] Rayat Bahra Institute of Engineering and Bio-Technology, Mohali, Punjab, India
[c] National Institute of Technical Teachers Training & Research, Chandigarh, India

ARTICLE INFO

Article history:
Received 7 March 2013
Received in revised form 9 July 2013
Accepted 15 July 2013
Available online 3 August 2013

Keywords:
Software
Components
Cohesion
Coupling
Build-or-buy
Fuzzy

ABSTRACT

Component based software system approach is concerned with the system development by integrating components. The component based software construction primarily focuses on the view that software systems can be built up in modular fashion. The modular design is a logical collection of several independent developed components that are assembled with well defined software architecture. These components can be developed in-house or can be obtained commercially from outside market making *build* versus *buy* decision an important consideration in development process. Cohesion and coupling (C&C) plays a major role in determining the system quality in terms of reliability, maintainability and availability. Cohesion is defined as the internal interaction of components within the module. On the other hand, coupling is the external interaction of the module with other modules i.e. interaction of components amongst the modules of the software system. High cohesion and low coupling is one of the important criteria for good software design. Intra-modular coupling density (ICD) is a measure that describes the relationship between cohesion and coupling of modules in a modular software system and its value lies between zero and one. This paper deals with the selection of right mix of components for a modular software system using build-or-buy strategy. In this paper, fuzzy bi-criteria optimization model is formulated for component selection under build-or-buy scheme. The model simultaneously maximizes intra-modular coupling density (ICD) and functionality within the limitation of budget, reliability and delivery time. The model is further extended by incorporating the issue of compatibility amongst the components of the modules. A case study is devised to explain the formulated model.

1. Introduction

Component based software system (CBSS) development is a promising solution for the development of large scale and complex systems. In contrast to CBSS, traditional development techniques for software development are suspected to have high development cost, low productivity, unmanageable software quality and high risk to move to new technology. CBSS provides efficient mechanism to significantly reduce development cost, time-to-market, improve maintainability; reliability and overall quality of software systems [19]. This approach has raised a remarkable amount of interests both in academia and software industry. The development of software system using CBSS approach facilitates the use of modularity where in each module is more controllable and developed by integrating components. The cost and quality of a modular software system are majorly affected by selection of suitable components. The developers have different options for the development of these small independent components such as choosing from available commercial-off-the shelf (COTS) components developed by different developers, in-house development from the scratch, or modifying the functioning of some existing in house components [10].

COTS (or in-house built) components tend to give higher quality software and enables the organization to have new technologies due to the fact that the development of these components finds advantage from the techniques such as design diversity, data diversity and environment diversity. COTS components are built by independent team of developers in different language and on different platform. They are relatively smaller in size and less complex and can be purchased from the various software developers in the market as standard components. Different versions of a COTS product are often available in the market with different specifications. With all these advantages of COTS, there are some risks which are associated in using the COTS products such as functional problems, security issues, compatibility issues, integration and interoperability issues, vendor issues, procurement and licensing issues and testing issues.

* Corresponding author. Tel.: +91 9891372928.
E-mail addresses: jhapc@yahoo.com (P.C. Jha), vikramgcet@gmail.com (V. Bali), sonam.narula88@gmail.com (S. Narula), malakalra2004@yahoo.co.in (M. Kalra).

1877-7503/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.jocs.2013.07.003

The idea of building a component in-house comes when a completely customized job is required for the internal customer of an organization. There is always a tussle between build-or-buy decisions in software development. Making this kind of decision is more than just the comparison of relative prices and availability of the component. Other important parameters like reliability, delivery time and interactions amongst the components should also be considered. Build decision is usually preferable when technology is easily available and cheaper, reliability of available COTS component can not be trusted, available component may not be compatible with the proposed system architecture. Sometimes there are existing in-house components that can be reused by modifying them to adjust to the present requirements. Reusability of existing components gives cost and time advantage. However such components are also categorized as build components.

Efficiency of a component based software system depends largely on architecture of the system. Cohesion and coupling (C&C) plays a major role in determining the system quality in terms of reliability, maintainability and availability. Cohesion is an intra-module property. It is defined as degree to which all components of a module work together as a functional unit. High cohesion is desirable as it accounts for improved strength and quality of the module. Coupling on the other hand is an inter-module property. It is defined as the degree of interdependence between components among the modules. Tight coupling is an undesirable property in modular based software development. "intra-modular coupling density (ICD)" is a measure for describing the relationship between cohesion and coupling of modules in the design of an object oriented software system [7].

A CBSS adopts modular approach for software development to improve the flexibility and comprehensibility of software systems [21]. In past, some of the authors have attempted to define criteria and develop metrics for software modularity. In the past few years, some work has been done in the area of development of modular software system based on the criteria of minimizing the coupling and maximizing the cohesion of software modules [21,7,8,13]. Stevens et al. have proposed a software quality metrics of cohesion and coupling. A highly cohesive module exhibits high reusability and loosely coupled systems enable easy maintenance of a software system [23]. Optimization problems of optimum selection of COTS components are widely studied by many researchers in the literature ([2,4,5,12,14,16–20,26,28,31] Authors in their work have developed optimization models with the objectives of maximization of reliability or minimization of cost or some of them have kept budget as a constraint. Few of the authors have addressed the build-or-buy strategy for optimal component selection problem. An optimization framework for "build-or-buy" decisions in software architecture was proposed by Cortellessa et al. [10]. Very few authors so far have discussed the optimal component selection problem based on cohesion and coupling of the software system incorporating build-or-buy strategy.

In this paper, bi-criteria optimization model is formulated to carry out the optimal selection of components incorporating build-or-buy strategy for CBSS based on the objectives of maximization of ICD as well as functionality of the software system under the constraints of cost, reliability and delivery time. Development of software system using CBSS approach may cause incompatibility issues amongst the COTS components belonging to different modules. The issue of incompatibility must be resolved to smooth the progress of interaction amongst the components of different modules. The incompatibility is caused because of the COTS components, as they are purchased from different vendors from the software market which leads to problem such as integration, execution, interfacing, licensing, etc. keeping in view these problems the paper also addresses the issue of incompatibility.

The paper is organized as follows. In Section 2, we have reviewed the literature. In Sections 3 and 4, notations and assumptions of the formulated model are presented. Section 5 discusses various criteria for component selection in CBSS. In Section 6, mathematical problem is formulated. In Section 7, we describe the methodology and present fuzzy optimization model for selecting components. Section 8 discusses the case study of the manufacturing enterprise for which the solution is provided in Section 9. Finally in Section 10, we furnish our concluding remarks.

2. Literature review

Optimization problems of optimum selection of components are widely studied by many researchers in the literature for development of safe and reliable software systems. The models used basic information on components reliability, cost, development time and delivery time. Chi et al. [9] investigated the trade-off between system reliability improvement and resource consumption. Software reliability to cost relation is developed from software reliability related cost model and software redundancy models with common cause failures. Ashrafi and Berman [4] presented optimization models which address the trade-off between reliability and cost in development of large software packages that consists of several programs. The authors formulated models considering with and without redundancy. Jung and Choi [15] introduced two optimization models for the COTS selection in the development of modular software systems considering cost/reliability trade-off. They have also addressed the issue of incompatibility amongst the COTS components of the modules. Abreu and Goulao [7] presented a quantitative approach to measure and assess the solutions of software modularity based on the criteria of minimal coupling and maximal cohesion. The optimal component selection problem due to Kapur et al. [16] considers software built by assembling COTS components performing multiple functions. In their work they have discussed optimal reliability allocation for modular software systems. Shen et al. [24] developed a fuzzy optimization model for selecting the best COTS product in the development of a software system based on COTS. Sarkar et al. [22] presented an information theoretic metrics that measure the quality of modularization of a non-object software system. These metrics are tested on open-source systems of large legacy-code business applications. Neubauer and Stummer [20] presented a two phase decision support approach based on multiobjective optimization for the COTS selection. Zachariah and Rattihalli [29] used goal programming approach in multi-criteria optimization model for the COTS selection. Cortellessa et al. [10] developed an optimization framework to support "build-or-buy" decision which affects the software cost as well as the ability of the system to meet its other requirements in selecting software components. Cortellessa developed the model that treats component selection as a cost-minimization problem under delivery time and quality constraints. Jha et al. [14] have formulated an optimization model for COTS selection for a fault tolerant modular software system. Joint optimization models were formulated with the objective of reliability maximization and minimization of execution time of the software system under budgetary constraint. Kwong et al. [19] proposed a modified way of measuring the cohesion and coupling of software modules considering the function ratings of various software components for CBSS development. Kwong et al. [30] further designed the scheme of typical reuse mode wherein six reuse modes are addressed from the sequence of activities. Optimization model proposed in this paper helps in selecting a reuse scenario for minimizing cost, maximizing reliability and satisfying system constraints. Gupta et al. [12] introduced an interactive fuzzy approach for solving a multi-objective COTS selection model that provides a strategy for selecting optimal COTS

alternatives products from the pool of alternative components that differ in their properties, viz. quality, reliability, functionality and cost. In their latest work Gupta et al. [31] have used fuzzy mathematical programming (FMP) for developing a bi-objective optimization model that aims to select the best-fit COTS components for a modular software system under multiple application development task.

3. Notation

M	the number of software modules
N	the number of software components available for modules
V_{ij}	the instances available for ith component of jth module; $i = 1,2,\ldots,N_j$; $j = 1,2,\ldots,M$
S_{ij}	the ith software component of jth software module, s.t. $Sc_{ij} = Sc_{ij} = Sc_i$ for all $j,j' = 1,2,\ldots,M$
S_i	the ith software component
Sc_i	the ith COTS component; $i = 1,2,\ldots,N$
S_{Bi}	the ith build component; $i = 1,2,\ldots,N$
m_j	the jth software module; $j = 1,2,\ldots,M$
$r_{ii'}$	the number of interactions between S_i and S'_i components; i; $i' = 1,2,\ldots,N$. As the coupling and cohesion are undirected relations, $r_{ii'} = r_{i'i}$
f_{ij}	f_{ij} are real numbers ranging from 0 to 1 depicting the function rating of S_{Bi} to m_j for build component; $i = 1,2\ldots,N$; $j = 1,2,\ldots,M$
f_{ijk}	f_{ijk} are real numbers ranging from 0 to 1 depicting the function rating of kth instance of Sc_i to m_j for COTS component; $i = 1,2,\ldots,N$; $j = 1,2,\ldots,M$
H	a threshold value of ICD_j of each module that needs to be set by decision makers
C_{ij}	cost of ith component available for jth module for build component
C_{ijk}	cost of kth instance of ith component available for jth module for COTS component
N_{ij}^{tot}	total number of tests performed on the ith component of jth module for in-house product
N_{ij}^{Suc}	number of successful tests performed on the ith component of jth module for in-house product
μ_{ijk}	the probability of failure on demand of kth instance of ith COTS component of jth module
ρ_{ij}	the probability that the in-house component is failure free during a single run given that N_{ij}^{Suc} test cases have been successfully performed
q_{ij}	the average number of the failure of the ith component of jth module
π_{ij}	the probability that a single execution of a software fails on a test case chosen from a certain input distribution
φ_{ij}	the probability of no failures occurring in a Poisson distribution with parameter q_{ij}
R_j	Reliability of jth module
d_{ijk}	delivery time of kth instance of ith component available for jth module for COTS component
t_{ij}	development time of ith component developed for jth module for build component
τ_{ij}	average time required to perform a test case for ith component of jth module
B	maximum budget limit set by the decision makers
T_i	maximum threshold given on delivery time of a ith component whether build-or-buy
x_{ijk}	binary variable $\begin{cases} 1, & \text{if } k\text{th instance of } Sc_i \text{ is selected for } m_j \\ 0, & \text{otherwise} \end{cases}$
y_{ij}	binary variable $\begin{cases} 1, & \text{if } S_{Bi} \text{ is selected for } m_j \\ 0, & \text{otherwise} \end{cases}$
z_{ij}	binary variable $\begin{cases} 1, & \text{if} S_i \text{ is selected for } m_j \\ 0, & \text{otherwise} \end{cases}$

4. Assumptions

1. CBSS is developed using modular approach.
2. The number of modules for the software system is finite.
3. Each module is a logical collection of several independent developed components. The components available are also finite in number.
4. At least one component is supposed to get selected from each module either COTS or in-house.
5. A threshold value of ICD_j, reliability, budget and delivery time are set by the decision makers.
6. The cost of an alternative is the development cost, if developed in house; otherwise it is the buying price for the COTS product.
7. The cost of an in-house component can be specified by using basic parameters of the development process.
8. Different COTS alternatives with respect to cost, reliability, functional rating and delivery time of a component are available.
9. Different in-house alternatives with respect to functional rating, unitary development cost, estimated development time, average time and testability of a component are available.
10. Redundancy is not allowed for a given instance, i.e. exactly one software instance is suppose to get selected for ith COTS component.
11. Interaction data for components is exactly same for all modules, irrespective of the selection happened. Interaction associated is set by the software development team.

5. Component selection in CBSS

Software development using CBSS approach strengthens reliability, quality of the system and saves time to market and development costs. Designers rely on components for the successful development of a CBSS.

In this work, we have assumed that for each component of a software architecture, several instances are available as COTS products. All the COTS instances of the same component (if any) can be considered functionally equivalent, namely no adaptation cost is needed to embed any instance in the whole system. The cost, delivery time and reliability attributes of interest of each COTS instance is assumed to be given from the vendor. We also assume that potential for building an in-house instance of each component might be available. Parameters related to the development cost, delivery time and reliability attributes of in-house instances can be estimated, when unknown in advance, from the software developers.

The solution of the optimization model indicates the instance to choose for each component (either one of the available COTS products or an in-house developed one) for a module in order to fulfill the multiple conflicting objectives under the constraints of cost, reliability and delivery time. If no COTS components are available then the in-house development of a component is a mandatory decision, independently of the cost incurred. On the other end, one of the available COTS products must be chosen for components that cannot be in-house built (e.g. for lack of expertise).

6. The problem formulation

Software development using CBSS approach follows a top down strategy wherein the first step involves identification of the functional requirements. After this, the number and nature of software modules are then determined. The next task is to integrate software components for modules. The selection of components should be in such a way so as to have maximum interactions of components within the software modules and minimum interactions of software components amongst the software modules.

Let S be a software architecture made of M modules, with a maximum number of N components available for each module. Fig. 2 shows how CBSS can be developed using software components.

In order to select COTS components for modular software systems, the following criteria may be used.

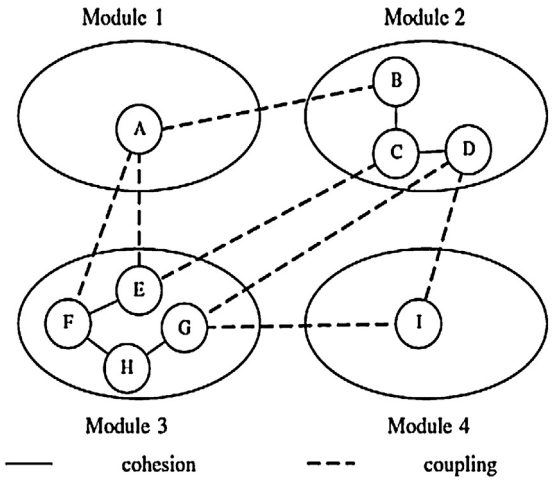

Fig. 1. Cohesion & coupling.

6.1. Intra-modular coupling density (ICD)

In this research, we have employed Abreu and Gaulao's [7] approach which yields the quantitative measures of cohesion and coupling. The authors in their work presented intra-modular coupling density (ICD) to measure the relationship between cohesion and coupling of modules in design of modular software system and is given as follows:

$$ICD = \frac{CI_{IN}}{CI_{IN} + CI_{OUT}} \quad \text{(i)}$$

where CI_{IN} is the number of component interactions within modules, and CI_{OUT} is the number of interactions between component of distinct modules.

Referring to Eq. (i), the ratio of cohesion to all interactions within the jth module can be expressed as ICD_j. However, it can be found if any module contains only one component, the values of ICD for that module becomes zero. To make up for the deficiency 1 is added to the numerator of Eq. (i) to form another measure of ICD as follows:

$$ICD_j = \frac{(CI_{IN})_j + 1}{(CI_{IN})_j + (CI_{OUT})_j} \quad \text{(ii)}$$

where ICD_j is the intra-modular coupling density for the jth module; $(CI_{IN})_j$ is the number of component interactions within the jth module; and $(CI_{OUT})_j$ is the number of component interactions between the jth module and other modules. Fig. 1 (replicated from [19]) shows diagrammatic depiction of cohesion and coupling of software modules in the development of modular software system.

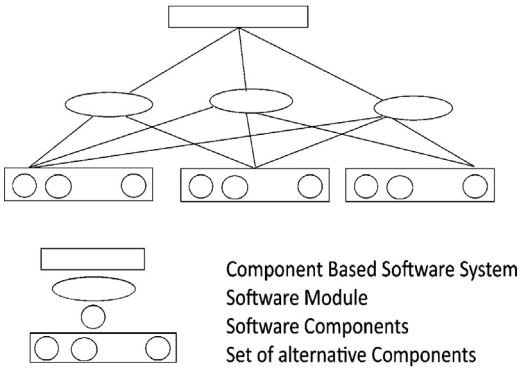

Fig. 2. A CBSS system.

The cohesion ([7]; Kwong et al. [19]) incorporating both in-house and COTS components impact [10] within the jth module $(CI_{IN})_j$ can be devised as follows:

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

All interactions including cohesion and coupling associated with the jth module, CA_j, can be expressed as:

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

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

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

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

$$CI_{IN} = \sum_{j=1}^{M} \sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} z_{ij} z_{i'j}$$

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:

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

6.2. Functional performance

Functionality of the COTS components can be defined as the ability of the component to perform according to the specific needs of the customer/organization requirements. The functional capabilities of the COTS components are different for different components as they are provided by different suppliers in the COTS market. We use functional ratings of the COTS components to the software module as coefficients in the objective function corresponding to maximizing the functional performance of the modular software system. These ratings are assumed to be given by the software development team. Functionality objective maximizes overall functionality of the system. The sum of the functionality of all the modules is selected from "build-or-buy" strategy. So overall functionality constraint can be expressed as:

$$F = \sum_{j=1}^{M} \sum_{i=1}^{N} \left(f_{ij} y_{ij} + \sum_{k=1}^{Vij} f_{ijk} x_{ijk} \right)$$

6.3. Threshold on ICD constraint

This constraint shows the minimum threshold H on ICD value of each module.

$$\frac{\sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} z_{ij} z_{i'j} + 1}{\sum_{i=1}^{N-1} \sum_{i'=i+1}^{N} r_{ii'} z_{ij} \sum_{j'=1}^{M} z_{i'j'}} \ge H; \quad j = 1, 2, \ldots, M; \quad j' = 1, 2, \ldots, M$$

6.4. Build versus buy decision

The equation stated below guarantees that redundancy is allowed in the module for both the build and buy components (i.e. in-house and COTS components). For a module, more than one component can be selected.

$$y_{ij} + \sum_{k=1}^{Vij} x_{ijk} = z_{ij}; \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M$$

$$\sum_{i=1}^{N} z_{ij} \geq 1; \quad j = 1, 2, \ldots, M$$

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

$$\sum_{j=1}^{M} z_{ij} = 1; \quad i = 1, 2, \ldots, N$$

6.5. Budget constraint

Cost is the major factor in determining the selection of components. 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 ith component available for jth module is $c_{ij}(t_{ij} + \tau_{ij} N_{ij}^{tot})$. So overall cost constraint can be expressed as:

$$\sum_{j=1}^{M} \sum_{i=1}^{N} \left(c_{ij}(t_{ij} + \tau_{ij} N_{ij}^{tot}) y_{ij} + \sum_{k=1}^{Vij} C_{ijk} x_{ijk} \right) \leq B.$$

6.6. Delivery time constraint

The delivery time of the component is the time of acquiring and integrating the components within and amongst the modules of the software system. The total delivery time includes development, integration and system testing time.

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

$$T_i = (t_{ij} + \tau_{ij} N_{ij}^{tot}) y_{ij} + \sum_{k=1}^{Vij} d_{ijk} x_{ijk}; \quad i = 1, 2, \ldots, N; \quad j = 1, \ldots, M$$

it is assumed that manpower is available to independently develop in-house component instances. Therefore, the delivery time constraint can be reformulated as follows:

$$\max_{i=1,2,\ldots,N} (T_i) \leq T$$

6.7. Test cases performed

The effect of testing on cost, reliability and delivery time of COTS product 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) test performed on the same component can be obtained as

$$N_{ij}^{Suc} = (1 - \pi_{ij}) N_{ij}^{Tot}; \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M$$

and it will be used to build reliability constraint.

6.8. Reliability of in-house components

Software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment [1]. Software developers aim at building software with minimum resources and simultaneously achieving a maximum reliability. The reliability of a COTS component for a specified module is provided by its vendor whereas the reliability of the built component is estimated by the software development team.

Authors in [10] have defined the probability of failure on demand of an in-house developed, the ith component of jth module under the assumption that the on-field users' operational profile is the same as the one adopted for testing in [6]. 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

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

The following equalities come straightforwardly:

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

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

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

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

therefore, we have $\rho_{ij} = \dfrac{(1 - \pi_{ij})}{(1 - \pi_{ij}) + \pi_{ij}(1 - \pi_{ij})^{N_{ij}^{Suc}}}$

6.9. Average number of failures

The probability of failure on demand of an in-house developed ith component of jth module can be expressed as $1 - \rho_{ij}$. Now we can write the average number of failures of ith component and jth module as q_{ij} where

$$q_{ij} = (1 - \rho_{ij}) y_{ij} + \sum_{k=1}^{V_{ij}} \mu_{ij} x_{ij}$$

6.10. Threshold on reliability constraint

The probability that no failure occurs during the execution of the ith component of jth module is given by $\varphi_{ij} = e^{-q_{ij}}$, which represents the probability of no failures occurring in a Poisson distribution with parameter q_{ij}

$$\prod_{i=1}^{N} \varphi_{ij} = \prod_{i=1}^{N} e^{-q_{ij}} \geq R_j; \quad j = 1, 2, \ldots, M$$

6.11. Optimization model-I

The multi-objective optimization model for component selection using CBSS development under build-or-buy scheme can be formulated as follows:

Problem (P1)

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

$$\text{Max } F = \sum_{j=1}^{M}\sum_{i=1}^{N}\left(f_{ij} y_{ij} + \sum_{k=1}^{Vij} f_{ijk} x_{ijk}\right) \quad (2)$$

Subject to $X \in S = \{x_{ijk}, y_{ij}, z_{ij}$ are binary variable/

$$\frac{\sum_{i=1}^{N-1}\sum_{i'=i+1}^{N} r_{ii'} z_{ij} z_{i'j} + 1}{\sum_{i=1}^{N-1}\sum_{i'=i+1}^{N} r_{ii'} z_{ij} \sum_{j'=1}^{M} z_{i'j'}} \geq H; \quad j=1, 2, \ldots, M; \quad j'=1, 2, \ldots, M \quad (3)$$

$$\sum_{j=1}^{M}\sum_{i=1}^{N}\left(c_{ij}(t_{ij} + \tau_{ij} N_{ij}^{tot}) y_{ij} + \sum_{k=1}^{Vij} C_{ijk} x_{ijk}\right) \leq B \quad (4)$$

$$(t_{ij} + \tau_{ij} N_{ij}^{tot}) y_{ij} + \sum_{k=1}^{Vij} d_{ijk} x_{ijk} \leq T_i; \quad i = 1, 2, \ldots, N; \quad j = 1, \ldots, M \quad (5)$$

$$N_{ij}^{Suc} = (1 - \pi_{ij}) N_{ij}^{Tot}; \quad i = 1, 2, \ldots, n; \quad j = 1, 2, \ldots, m_i \quad (6)$$

$$\rho_{ij} = \frac{(1 - \pi_{ij})}{(1 - \pi_{ij}) + \pi_{ij}(1 - \pi_{ij})^{N_{ij}^{Suc}}} \quad (7)$$

$$q_{ij} = (1 - \rho_{ij}) y_{ij} + \sum_{k=1}^{V_{ij}} \mu_{ijk} x_{ijk} \quad (8)$$

$$\prod_{i=1}^{N} \varphi_{ij} = \prod_{i=1}^{N} e^{-q_{ij}} \geq R_j; \quad j = 1, 2, \ldots, M \quad (9)$$

$$y_{ij} + \sum_{k=1}^{Vij} x_{ijk} = z_{ij}; \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M \quad (10)$$

$$\sum_{j=1}^{M} z_{ij} = 1; \quad i = 1, 2, \ldots, N \quad (11)$$

$$\sum_{i=1}^{N} z_{ij} \geq 1; \quad j = 1, 2, \ldots, M \quad (12)$$

$$x_{ijk} \in \{0, 1\}; \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M; \quad k = 1, 2, \ldots, V_{ij} \quad (13)$$

$$y_{ij} \in \{0, 1\}; \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M \quad (14)$$

$$z_{ij} \in \{0, 1\}; \quad i = 1, 2, \ldots, N; \quad j = 1, 2, \ldots, M\} \quad (15)$$

6.12. Optimization model-II

Optimization model-II is an extension of optimization model-I and addresses the issue of compatibility amongst the COTS components. In the development of software system, sometimes the COTS product for one module is incompatible with the alternative COTS products for other modules due to problem such as implementation technology, interfaces and licensing. Therefore, the compatibility constraints are also incorporated in the optimization model.

Compatibility constraints are used to deal with incompatibility amongst COTS components. The incompatibility constraint can be denoted by $x_{rs} \leq x_{u1t}$, that is if the module s chooses COTS component r, then the module t must choose the COTS component $u1$. This decision is called contingent decision constraint [15]. Suppose that there are two contingent decisions in the model, such as, the COTS alternative for the module s is only compatible with the COTS products $u1$ and $u2$ for the module t, i.e. either $x_{u1t} = 1$ if $x_{rs} = 1$ or $x_{u2t} = 1$ if $x_{rs} = 1$, these constraint can be represented as either $x_{rs} \leq x_{u1t}$ or $x_{rs} \leq x_{u2t}$. Since the presence of "either-or" constraint makes the optimization problem non-linear, it can be linearized by binary variable y_k as follows:

$$y_k = \begin{cases} 0, & \text{if } k\text{th constraint is active} \\ 1, & \text{if } k\text{th constraint is inactive} \end{cases}$$

Thus, only one out of z contingent decision constraints for any COTS products between two modules is guaranteed to be active if M is sufficiently large.

$$x_{rs} - x_{u_k h} \leq M y_k$$

$$\sum_{k=1}^{z} y_k = z - 1$$

$$y_k \in \{0, 1\}; \quad k = 1, 2, \ldots, z$$

Therefore, optimization problem-I can be reformulated by incorporating the additional constraints of compatibility as follows:

Problem (P2)

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

$$\text{Max } F = \sum_{j=1}^{M}\sum_{i=1}^{N}\left(f_{ij} y_{ij} + \sum_{k=1}^{Vij} f_{ijk} x_{ijk}\right)$$

Subject to $X \in S$

$$x_{rs} - x_{u_k h} \leq M y_k$$

$$\sum_{k=1}^{z} y_k = z - 1$$

$$y_k \in \{0, 1\}; \quad k = 1, 2, \ldots, z$$

7. Fuzzy approach for solving multi-objective optimization problem

From years conventional optimization techniques have been in practice to solve problems with well defined structure. Such optimization problems are formulated using crisp objective functions and constraints. However, real world situations are often not deterministic. There can be uncertainties or vagueness associated with data or preference of human judgment. Under such situations using crisp optimization is not a wise decision. This caused researchers to use fuzzy multi-objective optimization method with fuzzy parameters. Following steps are required to perform for solving fuzzy multi-objective optimization problem [25].

Step 1: Construct multi-objective optimization problem. Refer problem (P1).

Step 2: Solve multi-objective optimization problem by considering first objective function. This process is repeated for all the remaining objective functions. If all the solutions (i.e.

Table 1
Example description & functionality of COTS & in-house components.

Functional requirements	S_k	Software components	Module 1 (front office)	Module 2 (back office)	Module 3 (finance)
Inventory control and management	S_1	Sc_1	0.68	0.51	0.00
		Sc_2	0.22	0.63	0.01
		Sc_3	0.15	0.79	0.00
		Sc_4	0.23	0.87	0.00
		S_{B1}	0.54	0.67	0.34
Payment collection and authorization	S_2	Sc_5	0.94	0.10	0.55
		S_{B2}	0.62	0.23	0.87
Sales	S_3	Sc_6	0.75	0.45	0.22
		S_{B3}	0.76	0.23	0.30
Automatic updates	S_4	Sc_7	0.08	0.94	0.01
		Sc_8	0.10	0.22	0.00
		Sc_9	0.00	1.00	0.20
		Sc_{10}	0.20	0.45	0.05
		S_{B4}	0.27	0.69	0.30
E-Commerce	S_5	Sc_{11}	0.00	0.98	0.20
		Sc_{12}	0.00	0.31	0.10
		S_{B5}	0.11	0.82	0.25
Financial reporting	S_6	Sc_{13}	0.11	0.12	0.31
		Sc_{14}	0.05	0.07	0.71
		S_{B6}	0.36	0.54	0.78
Business rules and protocol	S_7	Sc_{15}	0.22	0.02	0.42
		S_{B7}	0.12	0.05	0.56
Shift wise reporting statistics	S_8	Sc_{16}	0.30	0.02	0.00
		Sc_{17}	0.80	0.10	0.00
		S_{B8}	0.67	0.37	0.47
Accounts	S_9	Sc_{18}	0.00	0.70	0.32
		Sc_{19}	0.00	0.06	0.78
		S_{B9}	0.07	0.14	
Finance	S_{10}	Sc_{20}	0.00	0.00	0.18
		S_{B10}	0.03	0.11	0.35

$X^1 = X^2 = \cdots = X^k = x_{ij}$, $i = 1,\ldots,N$; $j = 1,\ldots,M$) are same, select one of them as an optimal compromise solution and stop. Otherwise, go to step 3.

Step 3: Evaluate the kth objective function at all solutions obtained and determine the best (worst) lower bound (L_k) and best (worst) upper bound (U_k) as the case may be.

Step 4: Define membership function of each objective of optimization model. The membership function for ICD is given as follows.

$$\mu_{ICD}(x) = \begin{cases} 1, & \text{if } ICD(x) \geq ICD_u, \\ \dfrac{ICD(x) - ICD_l}{ICD_u - ICD_l}, & \text{if } ICD_l < ICD(x) < ICD_u, \\ 0, & \text{if } ICD(x) \leq ICD_l \end{cases}$$

where ICD_l is the worst lower bound and ICD_u is the best upper bound of ICD objective function.

The membership function for functionality is given as follows.

$$\mu_f(x) = \begin{cases} 1, & \text{if } f(x) \geq f_u, \\ \dfrac{f(x) - f_l}{f_u - f_l}, & \text{if } f_l < f(x) < f_u, \\ 0, & \text{if } f(x) \leq f \end{cases}$$

where f_l is the worst lower bound and f_u is the best upper bound of functionality objective function.

Step 5: Develop fuzzy multi-objective optimization model.

Following Bellaman–Zadeh's maximization principle [3] and using the above defined fuzzy membership functions, the fuzzy multi-objective optimization model for COTS selection is formulated as follows:

$$\begin{aligned} \text{Max} \quad & \lambda \\ \text{subject to} \quad & \lambda \leq \mu_{ICD(x)} \\ & \lambda \leq \mu_{f(x)} \\ & 0 \leq \lambda \leq 1 \\ & X \in S \end{aligned}$$

Solve the above model. Present the solution to the decision maker. If the decision maker accepts it then stop. Otherwise, evaluate each objective function at the solution. Compare the upper (lower) bound of each objective function with new value of the objective function. If the new value is lower (higher) than the upper (lower) bound, consider it as a new upper (lower) bound. Otherwise, use the old values as it is. If there are no changes in current bounds of all the objective functions then stop otherwise go to step 4.

The solution process terminates when decision maker accepts the obtained solution and considers it as the preferred compromise solution which is in fact a compromise feasible solution that meets the decision maker's preference.

8. Case study

A case study of CBSS development is presented to illustrate the proposed methodology of optimizing the selection of COTS and in-house developed components for CBSS development. A local software system supplier planned to develop a software system for small and medium size manufacturing enterprises. In this case, a software system is decomposed into three modules M_1, M_2 and M_3. A total of twenty software instances (Sc_1–Sc_{20}) are available in market to make up a set of ten alternative software components (S_1–S_{10}) for each module. Also these ten components can be developed in-house (S_{B1}–S_{B10}). Exactly one software instance in each set of alternatives may get selected for a particular software module for fulfilling the functional requirement. For example Sc_1, Sc_2, Sc_3, Sc_4 and S_{B1} all belong to the set of alternative software components S_1. Hence only one of the five components will be selected for fulfilling the S_1 functional requirement, whereas, Sc_1, Sc_2, Sc_3, Sc_4 are COTS components and S_{B1} is in-house developed component.

Table 2
Interaction among components.

		S_1				S_2	S_3	S_4				S_5		S_6		S_7	S_8		S_9		S_{10}
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
S_1	1	0	0	5	3	0	4	9	6	3	6	8	5	2	1	0	0	0	0	0	3
	2	0	0	3	1	0	4	5	3	7	8	5	3	4	2	0	0	1	0	0	2
	3	5	3	0	4	1	3	3	0	2	7	10	2	0	1	0	4	3	1	0	1
	4	3	1	4	0	2	2	7	8	6	9	2	3	0	0	0	0	0	2	1	1
S_2	5	0	0	1	2	0	10	1	2	3	2	1	2	0	0	0	0	0	2	1	2
S_3	6	4	4	3	2	10	0	1	2	2	2	2	0	0	0	0	8	7	1	1	2
S_4	7	9	5	3	7	1	1	0	0	3	7	3	2	2	1	0	0	0	1	2	1
	8	6	3	0	8	2	2	0	0	2	6	2	1	2	3	0	1	0	2	2	2
	9	3	7	2	6	3	2	3	2	0	4	10	7	2	1	0	0	0	2	1	1
	10	6	8	7	9	2	2	7	6	4	0	10	7	4	2	1	3	4	2	1	3
S_5	11	8	5	10	2	1	2	3	2	10	10	0	2	3	2	3	0	0	4	2	4
	12	5	3	2	3	2	0	2	1	7	7	2	0	4	1	2	0	0	3	1	5
S_6	13	2	4	0	0	0	0	2	2	2	4	3	4	0	0	8	1	0	10	7	10
	14	1	2	1	0	0	0	1	3	1	2	2	1	0	0	6	3	0	4	3	3
S_7	15	0	0	0	0	0	0	0	0	0	1	3	2	8	6	0	0	0	9	10	10
S_8	16	0	0	4	0	0	8	0	1	0	3	0	0	1	3	0	0	2	2	0	0
	17	0	1	3	0	0	7	0	0	0	4	0	0	0	0	0	2	0	0	0	0
S_9	18	0	0	1	2	2	1	1	2	2	2	4	3	10	4	9	2	0	0	1	10
	19	0	0	0	1	1	1	2	2	1	1	2	1	7	3	10	0	0	1	0	4
S_{10}	20	3	2	1	1	2	2	1	2	1	3	4	5	10	3	10	0	0	10	4	0

Table 3
Cost & delivery time data set of COTS components.

S_k	Components	Cost			Delivery time		
		Module 1	Module 2	Module 3	Module 1	Module 2	Module 3
S_1	S_{C1}	10	9	8	3	4	5
	S_{C2}	9	8	9	4	5	4
	S_{C3}	8	7	6	5	6	7
	S_{C4}	8	10	7	5	3	6
S_2	S_{C5}	7	7	8	6	6	5
S_3	S_{C6}	9	8	9	4	5	4
S_4	S_{C7}	6	9	6	7	4	7
	S_{C8}	7	6	7	6	7	6
	S_{C9}	8	10	10	5	3	3
	S_{C10}	10	8	8	3	5	5
S_5	S_{C11}	9	8	7	4	5	6
	S_{C12}	9	9	9	4	4	4
S_6	S_{C13}	10	7	8	3	6	5
	S_{C14}	8	6	9	5	7	4
S_7	S_{C15}	7	8	6	6	5	7
S_8	S_{C16}	6	9	7	7	4	6
	S_{C17}	7	7	10	6	6	3
S_9	S_{C18}	8	8	7	5	5	6
	S_{C19}	9	9	8	4	4	5
S_{10}	S_{C20}	9	10	6	4	3	7

Individual functional requirements and their corresponding alternative software components, as well as the function ratings of software components corresponding to software modules, are shown in Table 1. The function ratings describe the degree of functional contributions of the software components toward the software modules. The function ratings range from 0 to 1 where 1 refers to a very high degree of contribution while 0 indicates zero degree of contribution.

Table 2 shows the degrees of interaction among software components. The range of the degrees is 1–10. The degree '1' means a very low

Table 4
Cost & development time data set of in-house components.

Components	Cost			Development time		
	Module 1	Module 2	Module 3	Module 1	Module 2	Module 3
S_{B1}	7	8	9	6	7	8
S_{B2}	8	8	7	7	7	6
S_{B3}	6	7	6	5	6	5
S_{B4}	9	9	9	8	8	8
S_{B5}	6	7	8	5	6	7
S_{B6}	7	8	7	6	7	6
S_{B7}	8	7	9	7	6	8
S_{B8}	8	6	6	7	5	5
S_{B9}	7	7	7	6	6	6
S_{B10}	6	6	6	5	5	5

degree of interaction while the degree '10' refers to a very high degree of interaction.

In Table 3, cost (C_{ij} for all i, j) in 100$ unit; & delivery time (d_{ij} for all i, j) in days associated with COTS components is given (Table 4).

9. Solution

After developing the optimization model using the above data, the solution of each single objective problem is found and hence upper and lower bounds are obtained as follows:

	X^1	X^2
ICD	0.822	0.676
F	3.15	7.43

where X^1 and X^2 are optimal set of components selected for ICD and functionality objectives respectively. Then fuzzy multi-objective is developed and solved using the LINGO software [27].

Initial parameters:

Software system	COTS components	In-house developed components
Budget $B = 120$ threshold on ICD $H = 0.40$	Probability of failure on demand $\mu_{ij} = 0.0002$	Testing time $\tau_{ij} = 0.05$ Testability $\pi_{ij} = 0.002$

Delivery/development time	λ	ICD	Functionality	COTS components selected			In-house components selected		
				Module 1	Module 2	Module 3	Module 1	Module 2	Module 3
7	0.26	0.71	6.05	Sc_5 Sc_6 Sc_{17}	Sc_4 Sc_9 Sc_{11}	Sc_{13} Sc_{15} Sc_{19} Sc_{20}	–	–	–
8	0.259	0.69	6.25	Sc_5 Sc_6 Sc_{17}	Sc_3 Sc_7 Sc_{11}	Sc_{13} Sc_{20}	–	–	S_{B7} S_{B9}

In lieu of compatibility constraints we observe 4th component of second module is found to be compatible with the 13th component of third module. From the solution obtained it can be seen clearly as the delivery time for the system increases more of in-house build components gets selected.

10. Conclusion

The underlying structure in the present study gives the description for selecting software components under build-or-buy strategy for component based modular software systems. The formulated methodology provides a discipline for the selection of software components so as to obtain an optimal/near optimal solution. Case study of a manufacturing enterprise is used to explain the methodology. However, this methodology involves some subjective judgments from software development teams, such as the determination of the scores of interaction and functional ratings. In this regard, the fuzzy mathematical programming could be introduced to deal with the fuzziness caused by subjective judgments. The solution of the model gives the optimal value of objective functions along with the optimal set of components selected. The solution obtained depicts the variation in selected components when delivery time is increased. Only COTS components are selected with shorter delivery time whereas on increasing the delivery time we get a mix of COTS and in-house components.

The present work can also be applied to a fault tolerant software system. Fault tolerant systems have ability to tolerate faults and prevent system from failures. Reliability of such systems can be improved by creating redundancy at modular level. Different techniques of fault tolerance can be incorporated in the model formulated.

References

[1] ANSI/IEEE, Standard Glossary of Software Engineering Terminology, 1991, STD-729-1991.
[2] F. Belli, P. Jadrzejowich, An approach to reliability optimization of software with redundancy, IEEE Transaction on Software Engineering 17 (3) (1991) 310–312.
[3] R.E. Bellman, L.A. Zadeh, Decision-making in a fuzzy environment, Management Science 17 (4) (1970) B141–B164.
[4] O. Berman, N. Ashrafi, Optimization models for reliability of modular software systems, IEEE Transactions on Software Engineering 19 (11) (1993) 1119–1123.
[5] O. Berman, U.D. Kumar, Optimization models for recovery block schemes, European Journal of Operational Research 115 (1999) 368–379.
[6] Bertolino, L. Strigini, On the use of testability measures for dependability assessment, IEEE Transactions on Software Engineering 22 (2) (1996) 97–108.
[7] F. Brito e Abreu, M. Goulao, Coupling and cohesion as modularization drivers: are we being over-persuaded? in: Proceedings of the fifth European Conference on Software Maintenance and Reengineering, 2001.
[8] G. Carlo, J. Mehdi, M. Dino, Fundamentals of Software Engineering, Prentice-Hall Inc., 2001.
[9] D.H. Chi, H.H. Lin, W. Kuo, Software reliability and redundancy optimization, in: Proceedings of the Annual Reliability and Maintainability Symposium, IEEE, 2001, pp. 41–45.
[10] V. Cortellessa, F. Marinelli, P. Potena, An optimization framework for build-or-buy decisions in software architecture Computers and Operations Research Elsevier Science 35 (10) (2008) 3090–3106.
[12] P. Gupta, M.K. Mehlawat, S. Verma, COTS selection using fuzzy interactive approach, Optimization Letters 6 (2) (2012) 273–289.
[13] S. Ian, Software Engineering, Addison-Wesley Longman Publishing Co., Inc., 2001.
[14] P.C. Jha, P.K. Kapur, S. Bali, U.D. Kumar, Optimal component selection of COTS based software system under consensus recovery block scheme incorporating execution time, International Journal of Reliability, Quality and Safety Engineering 17 (3) (2010) 209–222.
[15] H.W. Jung, B. Choi, Optimization models for quality and cost of modular software systems, European Journal of Operational Research 112 (3) (1999) 613–661.
[16] P.K. Kapur, A.K. Bardhan, P.C. Jha, Optimal reliability allocation problem for a modular software system, Journal of Operational Research Society of India 40 (2) (2003) 133–148.
[17] U.D. Kumar, Reliability analysis of fault tolerant recovery blocks, Journal of Operational Research Society of India 35 (4) (1998) 281–294.
[18] D. Kumar, P.C. Jha, P.K. Kapur, U.D. Kumar, Optimal component selection problem for COTS based software system under consensus recovery block scheme: a goal programming approach, International Journal of Computer Applications 47 (4) (2012) 9–14.
[19] C.K. Kwong, J.F. Mu Tang, X.G. Luo, Optimization of software components selection for component-based software system development, Computers & Industrial Engineering 58 (2010) 618–624.
[20] T. Neubauer, C. Stummer, Interactive decision support for multiobjective COTS selection, in: Proceedings of the 40th Annual Hawaii International Conference on System Sciences, HICSS' 07, IEEE, 2007.
[21] S. Parsa, O. Bushehrian, A framework to investigate and evaluate genetic clustering algorithms for automatic modularization of software systems, Lecture Notes in Computer Science (2004) 699–702.
[22] S. Sarkar, G.M. Rama, A.C. Kak, API-based and information-theoretic metrics for measuring the quality of software modularization, IEEE Transactions on Software Engineering 33 (1) (2007) 14–32.
[23] R. Seker, A.J. van der Merwe, P. Kotze, M.M. Tanik, R. Paul, Assessment of coupling and cohesion for component-based software by using Shannon languages, Journal of Integrated Design & Process Science 8 (4) (2004) 33–43.
[24] X. Shen, Y. Chen, L. Xing, Fuzzy optimization models for quality and cost of software systems based on COTS, in: Proceedings of the Sixth International Symposium on Operations Research and Its Applications (ISORA' 06), Xinjiang, China, August 8–12, 2006, ORSC & APORC, 2006, pp. 312–318.
[25] W. Stevens, G. Myers, L. Constantine, Structured design, IBM Systems Journal 13 (2) (1974) 115–139.
[26] J. Tang, L. Mu, C.K. Kwong, X. Luo, An optimization model for software component under multiple applications development, European Journal of Operational Research 212 (2011) 301–311.
[27] H. Thiriez, OR software LINGO, European Journal of Operation Research 124 (2000) 655–656.
[28] Z. Wu, C.K. Kwong, J. Tang, J. Chan, Integrated models for software component selection with simultaneous consideration of implementation and verification, Elsevier Computers & Operations Research 39 (2012) 3376–3393.
[29] B. Zachariah, R.N. Rattihalli, A multicriteria optimization model for quality of modular software systems, Asia Pacific Journal of Operational Research 24 (6) (2007) 797–881.
[30] Z. Wu, J. Tang, C.K. Kwong, C.Y. Chan, A model and its algorithm for software reuse optimization problem with simultaneous reliability and cost consideration, International Journal of Innovative Computing, Information and Control 5 (7) (2011) 2611–2621.
[31] P. Gupta, M.K. Mehlawat, S. Verma, Fuzzy COTS selection for modular software systems based on cohesion and coupling under multiple applications environment, International Journal of Applied Evolutionary Computation 3 (4) (2012).

Dr. P.C. Jha obtained his Ph.D., M.Phil. and M.Sc. degrees in Operational Research (OR) from the University of Delhi. He is the Reader in the Department of OR, University of Delhi. He has published more than 45 research papers in the areas of software reliability, marketing and optimization in Indian and international journals and edited books. He has guided master's projects, MBA and M.Phil. dissertations and is supervising Ph.D. students in OR. His research interests include modeling and optimization in software reliability and marketing

Vikram Bali is the Head of the Department of Information Technology at Rayat Bahra Institute Engineering and Bio-Technology, Mohali, Punjab. His area of interest lies in the area of Software Engineering. He has authored few books and articles in national and international conferences. He is a life member of CSI and ISTE.

Sonal Narula has done her M.Sc. and M.Phil. from the Department of Operational Research, University of Delhi. Her area of interest is in the field of optimization and software reliability. She has presented papers at national and international conferences.

Mala Kalra is working as assistant professor in the Department of Computer science & Engineering at National Institute of Technical Teachers Training & Research, Chandigarh. She has presented papers at national and international conferences. Currently is working in the area of cloud computing.