Optimal Component Selection for Component-Based Systems

Muhammad Ali Khan[(a)], Sajjad Mahmood[(b)]
(a) Preparatory Year Mathematics Program, (b) Information and Computer Science Department
King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia
malikhan@kfupm.edu.sa, smahmood@kfupm.edu.sa

***Abstract*- In Component-based Software (CBS) development, it is desirable to choose software components that provide all necessary functionalities and at the same time optimize certain nonfunctional attributes of the system (for example, system cost). In this paper we investigate the problem of selecting software components to optimize one or more nonfunctional attributes of a CBS. We approach the problem through the lexicographic multi-objective optimization perspective and develop a scheme that produces Pareto-optimal solutions. Furthermore we show that the Component Selection Problem (CSP) can be solved in polynomial time if the components are connected by serial interfaces and all the objectives are to be minimized, whereas the corresponding maximization problem is NP-hard.**

I. INTRODUCTION

The extensive use of software has placed new expectations on software industry [1] and there is an ever growing push towards software reuse. Component-based software (CBS) development is an approach that aims to move the software industry away from developing each system from scratch. It focuses on integrating existing *off-the-shelf components* to build a software system, with a potential benefit of delivering quality system by using quality components. The success of CBS [2, 3, 4] depends on the ability to select suitable components. An inappropriate component selection can lead to adverse effects, such as introducing extra cost, in integration and maintenance phases [3].

Nonfunctional aspects play a significant role in determining software quality. Given the fact that lack of proper handling of nonfunctional aspects [5] of a software application has led to a series of software failures (e.g. [6]), nonfunctional attributes such as reliability, security and performance should be considered during the component selection phase of CBS development. Its importance for CBS development is further highlighted as component selection is a complex and risk prone process. We believe that for CBS to become part of mainstream software engineering culture there is a need for component selection approaches that take into consideration the nonfunctional aspects of a system.

In this paper, we present a component selection approach that qualifies nonfunctional attributes of the system and helps system analysts to evaluate the suitability of a component for a software application. The *component selection problem* (CSP) is represented as a multi-objective optimization problem on a clustered graph, with nonfunctional attributes formulated as objective functions. We use metrics to quantify contributions of components and their associated interfaces to each objective function. A lexicographic optimization algorithm is proposed that generates Pareto-optimal solutions for CSP.

Our approach is more general compared to existing component selection techniques (for example, [7, 8, 9, 10]), as we make no assumptions regarding the nature of interfaces and our lexicographic procedure can optimize any number of nonfunctional objectives. Furthermore, we analyze the computational complexity of component selection in detail, pointing out both polynomially solvable and NP-hard instances.

The rest of this paper is organized as follows. Section II reviews related literature. In Section III, we model CSP as an optimization problem. Section IV describes the lexicographic multi-objective framework for solving CSP while Section V deals with computational complexity of the problem. We conclude the paper by outlining our contributions and future work in Section VI.

II. RELATED WORK

Most component selection approaches are based on system functionalities or system architecture [9]. Considerably less attention has been devoted to component selection based on desired nonfunctional system attributes.

Lee et al. [11] have presented a component identification algorithm with focus on high cohesion and low coupling values. For the component identification process, at first,

T. Sobh, K. Elleithy (eds.), *Innovations in Computing Sciences and Software Engineering*,
DOI 10.1007/978-90-481-9112-3_79,

architecture design is analyzed to identify the architecture layers and subsystems. In the next step, the subsystem dependencies are determined using sequence diagrams. The subsystems are re-organized to make subsystem dependency less complex. This re-organization of subsystems is performed by re-arranging classes among subsystems. After subsystems are re-organized, the clustering algorithm is used to identify components. In [7] an approach has been proposed which assists in selecting components by using a clustering algorithm based on a set of predefined rules and heuristics.

Simulation composability is another well-known approach to adaptable component selection. In [12] it is shown that the complexity of optimal selection of adaptable components varies from polynomial to NP-complete, and even exponential depending on our assumptions.

Haghpanah et al. [9] have formulated component selection in terms of a series of feature subset selection problems. However, they point out that incorporating nonfunctional requirements into their solution framework remains a challenge.

Our work is motivated by the need for a general component selection scheme that achieves the optimal trade-off among nonfunctional attributes of a system. The same problem has been considered by Sedigh-Ali and Ghafoor [8]. One of the limitations of their graph-theoretic model is that it only applies when all components are linked by serial interfaces. Furthermore, the problem was formulated only to optimize three specific nonfunctional attributes (cost, reliability and complexity). We also use a graph-based representation for component selection but contrary to [9], we do not assume serial interfaces and our lexicographic multi-objective optimization technique works for any number of nonfunctional attributes.

Recently, Vescan et al. [10] have formulated the component selection as a multi-objective problem. They use an evolutionary computation technique to select a set of components which can satisfy a given set of functional requirements while minimizing the costs associated with component selection. One of the limitations of their work is that component compatibilities and integration effort are not considered. We address this issue in the present work.

III. Problem Formulation

In this section we describe a general instance of the component selection problem using clustered graphs.

A. Assumptions

Suppose we want to develop a CBS with p system functionalities labeled 1, 2, … , p. We assume that for each required functionality i a set C_i of candidate components is available, each providing the same level of functionality i. We further assume that the sets C_i are pairwise disjoint, that is $C_i \cap C_j = \emptyset$ for $i \neq j$. Thus a candidate component can provide a single functionality only. We would like to emphasize that these assumptions are not oversimplifying and are standard in CBS literature [8] as they can be reasonably satisfied by choosing components of suitable granularity. Unlike in [8] we assume nothing regarding the nature of interfaces between components (although later on we shall see that the type of interfaces has a significant impact on the computational complexity of the problem). Finally let n be the total number of candidate components.

B. Representation

We represent the problem as a clustered graph $G = (X, E)$ with node set X and edge set E. Let $|X| = n$ and $|E| = m$ where $|X|$ denotes the number of elements in set X. A node x corresponds to a candidate component and the node set is partitioned into p clusters $C_1, C_2, \ldots, C_p$, one for each set of candidate components. An interface between two components x and y is represented by an undirected edge (x, y). We say that cluster C_i and C_j are *adjacent*, denoted by $C_i \sim C_j$, if components (nodes) in C_i and C_j are linked by interfaces (edges). We define an *induced subgraph* s of G as a subgraph containing all edges between its nodes [13, p. 49].

This graphical representation is similar to the ones given in [8] and [14] but is more general. The nonfunctional attributes of a CBS depend on the nonfunctional attributes of its components (nodes) and their interfaces (edges). Examples of such attributes include cost, reliability, response time and complexity of the system [15, p. 160]. We would obviously like to maximize the first two while minimizing the later two. A nonfunctional attribute can be quantified by defining a suitable metric.

Suppose that we are interested in optimizing the nonfunctional metrics $F_1, F_2, \ldots, F_r$, and numerical values of all metrics of interest are known for every component and likewise for every interface. Since we are formulating component selection as an optimization problem, we shall call $F_1, F_2, \ldots, F_r$ the *objective functions* or simply *objectives*, each to be maximized or minimized.

Let the metric $f_k(x)$ denote the contribution of component (node) x, if selected in a feasible solution s, to the k^{th} objective function F_k and $f_k(x,y)$ the corresponding contribution of the interface (edge) (x, y). The component level metrics describe the role of each component in the system, while the interface level metrics encompass the effect of integrating individual components. For instance if F_k represents the cost objective then $f_k(x)$ is the cost of acquiring

component x. Whereas $f_k(x, y)$ can be interpreted as the cost of acquiring middleware or the cost of developing the integration code internally for components x and y.

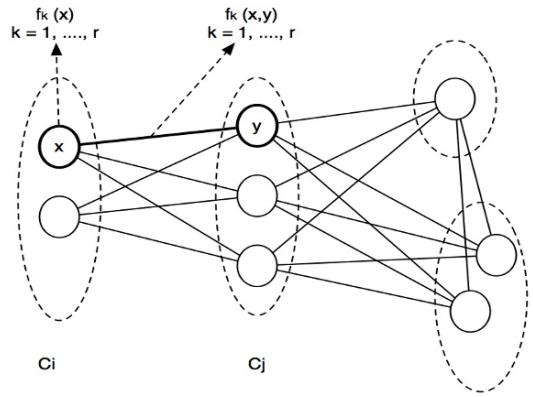

Fig. 1. An instance of the component selection problem

We remark that the values of component level metrics can be found through a variety of sources such as vendor specifications, black-box testing, simulation and extrapolating the market data [8]. The same applies to interface metrics if the integration code is obtained as middleware. For internally written integration code white-box testing can be used.

C. *Statement of Problem*

We can now state the component selection problem (CSP) as a multi-objective optimization problem (MOP) that asks for an induced subgraph s of G containing exactly one node (component) from each cluster C_i such that all the objectives $F_1, F_2, \ldots, F_r$ are optimized.

A typical instance of the component selection problem is shown in Fig. 1. Our approach to solving CSP is described in the next section.

IV. Solving Component Selection Problem as MOP

Compared to single objective optimization problems, solving MOPs require more sophisticated techniques as we try to strike an optimal balance among different objective functions. Instead of looking for a single optimum we have to search for the *Pareto front* (also known as *Pareto set*) which is the set of non-dominated solutions that cannot be improved in one objective without worsening another [16, p. 19]. Several methods are known to achieve this such as lexicographic multi-objective optimization (LMO), aggregate objective function (AOF) method, goal programming (GP) and evolutionary multi-objective optimization (EMO). Although evolutionary algorithms have been preferred in recent literature on component selection [9, 10], we adopt LMO based on following grounds:

- Evolutionary algorithms (EAs) are problem independent and thus provide little insight into the nature and complexity of the problem. Furthermore theories explaining how EAs perform are few and only recently some progress has been made on rigorously analyzing the computational complexity of EAs [17].
- Goal programming is attractive for its ease of implementation. However it is known to produce solutions that are not Pareto efficient [18].
- Despite the fact that AOF method is probably the most intuitive multi-objective optimization method it suffers from several drawbacks. Firstly by combining different objective functions into a single AOF the multi-objective nature of the problem is lost. Secondly most AOF techniques such as the weighted-sum method tend to be highly subjective. Last but not the least AOF methods fail when the Pareto front is concave [19].
- An LMO scheme orders the solution vectors lexicographically according to a priority ranking of objectives. The LMO optimizes a first objective and then as far as a choice remains a second one and so on. The k^{th} objective in the ranking is considered only after the prior $k-1$ objectives have been successively optimized. LMO is subjective as it depends on a priority ranking. However, this approach lends itself to component selection problem as nonfunctional software attributes are always subject to stakeholder's priorities. A domain expert plays an important role in assessing the relative importance of nonfunctional attributes of interest and can prioritize the objectives accordingly. Moreover, LMO always produces Pareto-optimal solutions [16, p. 135], it is fast and allows a rigorous computational complexity analysis (refer to Section V for details).

In this paper, our proposed LMO solution is inspired by Volgenant's work [20]. We have selected Volgenant method as it is particularly suitable for shortest path type problems; and many special instances of component selection problem are multi-objective variants of the shortest path problem.

A. *The Lexicographic Approach*

All objectives considered in this paper are *sum objectives*. A sum objective F can be expressed mathematically as $F = \sum_i f$, where $\sum_i f$ is the sum of contributions to the objective F of all components and interfaces occurring in a

feasible solution s. Many objectives that do not appear to be sum objectives can be represented in this way. For instance an objective of the form $F = \prod_s f$, where every $f > 0$, can be transformed into an equivalent sum objective

$$\log(F) = \sum_s \log(f).$$

We point out that restricting to sum objectives does not result in a significant loss of generality as most nonfunctional software attributes fall in this category (e.g., cost, response time and complexity are sum objectives while reliability can also be expressed as a sum objective).

Recall that an instance of CSP consists of a clustered graph $G = (X, E)$ with node set X partitioned into p clusters $C_1, C_2, \ldots, C_p$, edge set E and a series of (sum) objective functions $F_1, F_2, \ldots, F_r$, each to be maximized or minimized. A feasible solution is an induced subgraph s of G containing exactly one node from each cluster. For each component (node) x the values $f_1(x), f_2(x), \ldots, f_r(x)$ are known and likewise for each interface (edge) (x, y).

Let F be an objective of interest (one of $F_1, F_2, \ldots, F_r$) for the above CSP instance Let $S(E)$ denote the set of feasible solutions and $S^*(E)$ the set of optimal solution over the edge set E for the objective function F. We introduce the quantities

$$\delta(x, y) = \begin{cases} 1 & \text{If the egde } (x, y) \text{ does not occur in any} \\ & \text{optimal solution of objective } F \\ 0 & \text{otherwise} \end{cases}$$

If $\delta(x, y) = 0$ then the optimal value of objective function F remains unchanged under the additional constraint that (x, y) must occur in an optimal solution. We now define

$$E^* = \{(x, y) \mid \delta(x, y) = 0\}.$$

We observe that E^* is the union of edge sets of all optimal solutions of F. Let G^* be the graph with edge set E^* and node set X^* consisting of end nodes of edges in E^*. Then G^* is also a clustered graph with p clusters. We further note that G^* is completely determined by its edge set E^* so it suffices to consider E^* only.

The next theorem shows that for any objective F the set of optimal solutions over E is equal to the set of feasible solutions over E^*.

Theorem 1 For a given objective F we have $S^*(E) = S(E^*)$.

Proof. Suppose an optimal solution $s^* \in S^*(E)$ contains an edge (x, y) with $(x, y) \notin E^*$. Then $\delta(x, y) = 1$ and the solution s^* can be improved by replacing the edge (x, y) with a better alternative. This contradicts the optimality of s^*. Thus s^* only contains edges from E^*.

Conversely assume that a feasible solution $s \in S(E^*)$ contains an edge (x, y). Then $(x, y) \notin E^*$ and so $\delta(x, y) = 0$. This is true for all edges of s. Therefore $s \in S^*(E)$.

Volgenant [20] proved the directed edge version of Theorem 1 on the same lines. Based on this result he proposed an iterative scheme for solving LMO instances. Volgenant's scheme can be adapted to solve CSP as follows:

Procedure: *Lex CSP*

Initialization	$k := r + 1$;	$E_k^* := E$;
Iteration	$k := k - 1$;	$E_k := E_{k+1}^*$
	determine E_k^* and $S^*(E_k)$	
Termination	If $k > 1$ and $\lvert S^*(E_k) \rvert > 1$	
	then go to *Iteration*	
	else the CSP instance has been solved.	

It must be noted that the objectives should be arranged in ascending order of priority with F_r being the most important. The procedure starts with the edge set E and successively reduces it $(E \supseteq E_r^* \supseteq E_{r-1}^* \supseteq \cdots \supseteq E_1^*)$ until either all objectives are optimized $(k = 1)$ or there is only one solution left $(|S^*(E_k)| = 1)$.

B. *Optimizing Individual Objectives*

Every iteration of the above procedure corresponds to maximizing or minimizing a single objective F over an undirected clustered graph $G = (X, E)$ with node clusters $C_1, \ldots, C_p$. We can formulate this single objective optimization problem as a {0, 1}-integer linear program (ILP) by introducing the variables

$$u(x) = \begin{cases} 1 & \text{If node } x \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

$$v(x, y) = \begin{cases} 1 & \text{If edge } (x, y) \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

The linear programing relaxation of the resulting ILP can be solved by a suitable branch and bound algorithm.

V. Computational Complexity And NP-hardness

The run time complexity of *Lex CSP* procedure developed in Section III depends on how fast the quantities δ and the edge set E_k^* can be determined during an iteration.

Using the notation of [20] let $T(n, m)$ be the time to solve a single objective CSP on *n* nodes and *m* edges. Given an edge (x, y) we can calculate $\delta(x, y)$ by solving the single objective CSP under the restriction that (x, y) must appear in an optimal solution. The restricted problem can be solved in time $T(n-2, m-1)$ as two nodes and one edge have been fixed. If the optimal objective value of the restricted problem is the same as the original problem then $\delta(x, y) = 0$; otherwise $\delta(x, y) = 1$. Hence the set E_k^* can be determined in time $T(n, m) + O(m)T(n-2, m-1)$. For most practical applications the complexity is dominated by the term $T(n, m)$. The running time of *Lex CSP* is therefore $O(r)T(n, m)$ where *r* is the number of objectives.

A. The Case of Serial Interfaces

The run time $T(n, m)$ strongly depends on the nature of component interfaces; and also on whether the objective is being maximized or minimized. An important special case of CSP occurs when the components are connected by serial interfaces [8] (see Fig.2.).

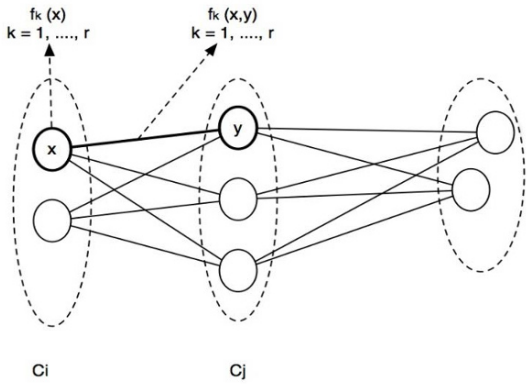

Fig. 2. Component selection problem with serial interfaces

Under this assumption the single objective minimization CSP reduces to a variant of all pairs shortest path problem that can be solved by a Floyd-Warshall type algorithm in time $T(n, m) = O(n^3)$. Thus if all objectives are to be minimized, the *Lex CSP* procedure solves the multi-objective CSP in time at most $O(rn^3)$.

The single objective maximization CSP is much harder to solve as it is equivalent to the all pairs longest path problem which is known to be NP-hard.

Therefore we conclude that the multi-objective minimization CSP can be solved in polynomial time if the components are connected through serial interfaces; while the corresponding maximization CSP is NP-hard.

VI. Concluding Remarks

In this paper, we have presented a new technique for component selection that guides system analysts through a process of identifying suitable components that optimize one or more nonfunctional attributes of a CBS. We formulate component selection as a lexicographic multi-objective optimization problem, with an aim to optimize nonfunctional objectives, and present an efficient solution scheme. The proposed model guides selection by identifying components that will collectively achieve the best tradeoff among the metrics desired for the system. This technique results in a quality management method that can alleviate concerns regarding uncertainty in the cost and quality of a component-based system. The complexity of component selection problem has also been examined extensively.

In future, there is a need to have an integrated requirements analysis and component selection process that analyzes both functional and nonfunctional system requirements and extends the presented technique to select optimal set of components. We also plan to conduct an empirical study to better understand the benefits and limitations of our component selection process for a CBS.

Acknowledgment

The authors would like to thank King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia for continuous support in research.

References

[1] S. Mahmood, R. Lai, Y. S. Kim, J. H. Kim, S. C. Park and H. S. Oh, "A survey of component based system quality assurance and assessment," *Information and Software Technology*, vol. 47, pp. 693 – 707, 2005.

[2] N. A. Maiden and C. Ncube, "Acquiring COTS software selection requirements," *IEEE Software*, vol. 15, pp. 46 – 56, 1998.

[3] K. R. P. H. Leung and H. K. N. Leung, "On the efficiency of domain based COTS product selection method," *Information and Software Technology*, vol. 44, pp. 703 – 715, 2002.

[4] C. Alves and A. Finkelstein, "Investigating conflicts in COTS decision making," *International Journal of Software Engineering and Knowledge Engineering*, vol. 13, pp. 1 – 21, 2003.

[5] L. M. Cysneiros and J. C. S, Leite, "Nonfunctional requirements: from elicitation to conceptual models," *IEEE Transactions on Software Engineering*, vol. 30, pp. 328 - 350, 2004.

[6] A. Finkelstein and J. Dowell, "A comedy of errors: the London ambulance service case study," *Proc. of Eight International Workshop on Software, Specification and Design*, pp. 2 - 5, 1996.

[7] H. Jain, N. Chalimeda, N. Ivaturia and B. Reddy, "Business component identification: a formal approach," *Proc. of Fifth International Conference on Enterprise Distributed Object Computing*, pp. 183 - 187, 2001.

[8] S. Sedigh-Ali, and A. Ghafoor, "A graph-based model for component-based software development", *Proc. of the 10th IEEE Workshop on Object-Oriented Real-Time Dependable Systems*, 2005.

[9] N. Hagpanah, S. Moaven, J. Haibibi, M. Kargar and S. H. Yaganeh, "Approximation algorithm for software component selection problem," *Proc. of 14th Asia Pacific Software Engineering Conference*, pp. 159 - 166, 2007.

[10] A. Vescan, C. Grosan and H. F. Pop, "Evolutionary algorithms for the component selection problem," *Proc. of 19th International Conference on Database and Expert Systems Application*, pp. 509 - 513, 2008.

[11] J. K. Lee, S. J. Jung, S. D. Kim, W. H. Jang and D. H. Ham, "Component identification method with coupling and cohesion," *Proc. of Eight Asia Pacific Software Engineering Conference*, pp. 79 - 86, 2001.

[12] R. G. Bartholet, D. C. Brogan and P. F. Reynolds, "The computational complexity of component selection in simulation reuse," *Proc. of the 2005 Winter Simulation Conference*, pp. 2472 - 2481, 2005.

[13] J. Gross and J. Yellen, *Graph Theory and its Applications*, FL: CRC Press Inc., 1999.

[14] S. Krishnamurthy and A. Mathur, "On the estimation of reliability of a software system using reliabilities of its components," In *Proc. of the 8th Int'l Symp. on Software Reliability Eng. (ISSRE '97)*, 1997.

[15] L. Chung, B.A. Nixon, E. Yu and J. Mylopoulos, *Non-functional Requirements in Software Engineering*, Kluwer International Series in Software Engineering Vol. 5, Kluwer Academic Publishers, 2000.

[16] M. Ehrgott, *Multicriteria Optimization*, Lecture Notes in Economics and Mathematical Systems (no. 491), Berlin: Springer-Verlag, 2000.

[17] P. S. Oliveto, J. He, and X. Yao, "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results," *International Journal of Automation and Computing*, vol. 4, pp. 281-293, 2007.

[18] Caballero R., L. Rey and F. Ruiz, "Determination of Satisfying and Efficient Solutions in Convex Multiobjective Programming," *Optimization,* vol. 37, no. 2, pp. 125-137, 1996.

[19] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted-sums of objectives for Pareto set generation in multicriteria optimization problems," *Structural Optimization* vol. 14, pp. 63–69, 1997.

[20] A. Volgenant, "Solving Some Lexicographic Multi-objective Combinatorial Problem," *European Journal of Operational Research*, vol. 139, no. 3, pp. 578-584, 2002.