Evolutionary Algorithms for the Component Selection Problem

Andreea Vescan, Crina Groşan, Horia F. Pop
Babeş - Bolyai University
Department of Computer Science
M. Kogălniceanu 1, 400084, Cluj-Napoca, Romania
{avescan, cgrosan, hfpop}@cs.ubbcluj.ro

Abstract

Component selection is a crucial problem in Component Based Software Engineering. Component Based Software Engineering (CBSE) is concerned with the assembly of pre-existing software components that leads to a software system that responds to client-specific requirements.

We are approaching the component selection involving dependencies between components (requirements). We formulate the problem as multiobjective. The approach used is an evolutionary computation technique. Various representations were used with various applied methods to deal with the multiobjectives.

1 Introduction

Component-Based Software Engineering (CBSE) is concerned with composing, selecting and designing components [5]. As the popularity of this approach and hence number of commercially available software components grows, selecting a set of components to satisfy a set of requirements while minimizing cost is becoming more difficult.

In Component-based Software Engineering (CBSE), the construction of cost-optimal component systems is not a trivial task. It requires not only to optimally select components but also to take their interplay into account.

In this paper, we address the problem of (automatic) component selection.

Informally, our problem is to select a set of components from available component set which can satisfy a given set of requirements while minimizing sum of the costs of selected components. The dependencies between the components must be taken into account. To achieve this goal, we should assign each component a set of requirements it satisfies. Each component is assigned a cost which is the overall cost of acquisition and adaptation of that component.

In general, there may be different alternative components that can be selected, each coming at their own set of offered requirements. We aim at a selection approach that guarantees the optimality of the generated component system, an approach that takes into consideration also the dependencies between components (restrictions on how the components interact). The compatibility of components is not discussed here, it will be treated in a future development.

The paper is organized as follows: Section 2 starts with the problem formulation. Related work on Component Selection is discussed in Section 3. New considerations are stated in Section 4. The proposed approach (that uses an evolutionary algorithm) is presented in Section 5. We conclude our paper and discuss future work in Section 6.

2 Component Selection Problem. Formal statement

Informally, our problem is to select a subset of components (each of them satisfying a set of functionalities) and to connect them such that the target component system fulfills the requirements that need to be satisfied.

2.1 Simple Component Selection Problem (SCSP)

Simple Component Selection Problem (SCSP) is the problem of choosing a number of components from a set of components such that their composition satisfies a set of objectives. The notation used for formally defining SCSP, as laid out in [6] with a few minor changes to improve appearance is described in the following.

Consider SR the set of the final system requirements (target requirements) as

$$SR = \{r_1, r_2, ..., r_n\},$$

and SC the set of components available for selection as

$$SC = \{c_1, c_2, ..., c_m\}.$$

1529-4188/08 $25.00 © 2008 IEEE
DOI 10.1109/DEXA.2008.138

Each component c_i can satisfy a subset of the requirements from SR,

$$SR_{c_i} = \{r_{i_1}, r_{i_2}, ..., r_{i_k}\}.$$

The goal is to find a set (subset) of components Sol in such a way that to every requirement r_j from the set SR can be assigned a component c_i from Sol where r_j is in SR_{c_i}.

2.2 Criteria-based Component Selection Problem (CCSP)

Criteria Component Selection Problem (CCSP) is the problem of choosing a number of components from a set of components such that their composition satisfies a set of objectives and using various criteria. A much more detail discussion it follows in the next Section 3.

Another variation of Component Selection Problem is that stated in [9]. In addition to the above description a cost of a component c_i is considered $cost(c_i)$. The goal is to find a set of components Sol in such a way that to every requirement r_j from the set SR can be assigned a component c_i from Sol where r_j is in SR_{c_i}, while minimizing the number of components in the solution Sol and/or while minimizing $\sum_{c_i \in Sol} cost(c_i)$. Another criteria introduced in a previous paper concerns the dependencies between the involved components. To specify the component dependencies we have introduced in [15] a dependency matrix. We are only interested in the provided functionalities (that are in the set of requirements SR of the final system) of the components.

The dependencies specification table must contain dependencies between each requirement in the set of given requirements SR.

Dependencies	r_1	r_2	r_3
r_1	$\surd$	$\surd$	
r_2			$\surd$
r_3	$\surd$	$\surd$	

Table 1. Dependencies specification table

In Table 1 we have specified the dependencies between the requirements r_1, r_2, r_3: the second requirement depends on the third requirement, the third requirement depends on the first and the second requirement.

Some exceptional cases are required to be checked: no self dependency (the first requirement depends on it self), no reciprocal dependency (the second requirement depends on the third and the third depends on the second requirements) and no circular dependencies (the second requirement depends on the third, the first depends on the second and the third depends on the first). All the above situations are presented in Table 1.

We have discussed the case when only the dependencies between the requirements from the set of requirements SR. A more real case should be also considered: a component could have other requirements that need to be satisfied before some of its provided services are used.

3 Related work

Component selection methods are traditionally done in an architecture-centric manner. An approach was proposed in [13]. The authors present a method for simultaneously defining software architecture and selecting off-the-shelf components. They have identified three architectural decisions: object abstraction, object communication and presentation format. Three type of matrix are used when computing feasible implementation approaches. The proposed method analysis the relationship between components and architecture. The goal is to discover the effect of chosen architectural decisions on the usable component base.

Another type of component selection approach is built around the relationship between requirements and components available for use. The goal is to recognize the mutual influence between requirements and components in order to obtain a set of requirements that is consistent with what the market has to offer (PORE [2] and CRE [3]). In [7] the authors have presented a framework for the construction of optimal component systems based on term rewriting strategies. By taking these techniques from compiler construction, especially optimizing code generation, they have developed an algorithm that finds a cost-optimal component system. They present a unified approach to the construction of component systems which allows to first select an optimal set of components and adapters and afterwords to create a working system by proving the necessary glue code.

Paper [9] proposes a comparison between a Greedy algorithm and a Genetic Algorithm. The discussed problem considers a realistic case in which cost of components may be different. The selection function from the greedy approach take into consideration both number of provided (offered requirements) of the components and the cost of the component. For the genetic algorithm only the minimization of $\sum_{c_i \in Sol} cost(c_i)$ is considered. The minimization of used components is not discussed. The [6] approach addresses the selection of a minimal set of components to satisfy a set of objectives. Selecting the component with the maximal number of provided operations is considered. The approach [4] formulates both ranking and selection of candidate software components as a series of feature subset selection problems to which search based software engineering can be applied. The algorithm in [4] consider all the components to be previous sorted according to their weight value.

Then all components with the highest weight are included in the solution until the budget bound has been reached.

Existing methods for the Component Selection Problem include OTSO [11] and BAREMO [12]. These methods define criteria upon which to judge alternatives for a component role and the synthesis of multiple criteria to decide the most promising alternative. They aim to answer the question: given a description of a component needed in a system, what is the best existing alternative available in the market?

4 New considerations

One of the most popular definition of a component [14] was offered by a working group at ECOOP (the European Conference on Object-Oriented Programming).

A software component ([14]) is a unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and it is subject to composition by third parties.

This definition emphasizes component composition. As a unit of composition, each component has its specified interface that determines how it can be composed with other components.

Sterling extended the above definition then distinguished three aspects of a component:

- A specification that describes what the component does and how it should be used.
- A packaging perspective in which a component is considered as a unit of delivery.
- An integrity perspective in which a component is considered as an encapsulation boundary.

They defined a component simply as: "A software package which offers services through interfaces".

Although these definitions differ in detail, they assert that a component is an independent software package that provides functionality via well-defined interfaces. Moreover, they all emphasize the importance of well-defined interfaces. The interface could be an export interface through which a component provides functionality to other components or an import interface through which a component gains services from other components. They also emphasize the "black-box" nature of a component: that is, a component can be incorporated in a software system without regard to how it is implemented.

Figure 1 describes our component selection problem: from a set of existing available components SC we need to select a subset that satisfies a set of objectives SR. We have denoted by r_i the $i-th$ requirement or offered service.

In our previous work we have considered in the composition only the provided services (as satisfied requirements

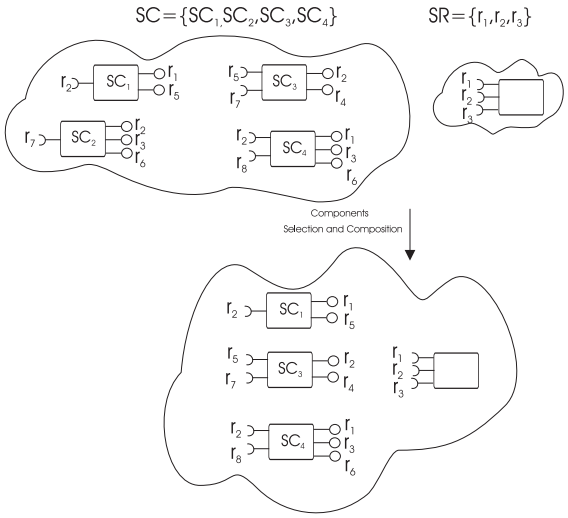

Figure 1. Component Selection Problem. Reasoning

in the final system) of a component. We haven't took under consideration the required services of a component, only those requirements as dependencies that were in the set of SR. For example, in the above Figure 1 the final system computed after selection consists of C_1, C_3 and C_4 components. The r_2 requirement of the first and the forth components is satisfied because it is included into the final solution, but the requirements r_7 and r_8 of the second and the fourth components are not satisfied.

A more complex situation is going next to be considered: components required services are going to be used. As stated above, the requirements r_7 and r_8 is going to be considered in our next research work.

5 Evolutionary approaches for the component selection problemn

Evolutionary algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. They are well known suitable approaches for optimization problems.

There are several ways to deal with a multiobjective optimization problem. We have used both the weighted sum method [10] and the Pareto dominance [1] principle.

The weighted sum method. Let us consider we have the objective functions f_1, f_2,..., f_n. This method takes each objective function and multiplies it by a fraction of one, the "weighting coefficient" which is represented by w_i. The modified functions are then added together to obtain a single cost function, which can easily be solved using any method

which can be applied for single objective optimization.

Mathematically, the new function is written as:

$$\sum_{i=1}^{n} w_i f_i, \ where \ 0 \leq w_i \leq 1 \ and \ \sum_{i=1}^{n} w_i = 1.$$

The Pareto dominance principle. Consider a maximization problem. Let x, y be two decision vectors (solutions) from the definition domain. Solution x dominate y (also written as x $\succ$ y) if and only if the following conditions are fulfilled:

1. $f_i(x) = f_i(y), \forall \, i = 1, n;$
2. $\exists j \in \{1, 2, , n\} : f_j(x) > f_j(y).$

That is, a feasible vector x is Pareto optimal if no feasible vector y can increase some criterion without causing a simultaneous decrease in at least one other criterion.

In what follows, we present two evolutionary approaches which use different representations for the component selection problem. Each of them is described in detail below.

5.1 Requirements - based chromosome representation

A solution (chromosome) is represented as a string of size equal to the number of requirements from SR. The value of i - th gene represent the component satisfying the i-th requirement. The values of these genes are not different from each other (which means, same component can satisfy multiple requirements).

The approaches use principles of evolutionary computation and multiobjective optimization [8].

In [16] the component selection problem involving dependencies between components (requirements) is approached. We have formulated the problem as multiobjective, involving 2 objectives and one constraint. The approach used is an evolutionary computation technique, a steady-state evolutionary model as given in the Algorithm 1.

The problem can be formulated as a multiple objective optimization problem having 2 objectives: the total cost of the components used and the number of components used. Both objectives are to be minimized. Besides these, the dependencies were also treated, but were considered as constraints.

The new function obtained by aggregating the two objectives can be written as:

$F(x) = \alpha \cdot f_1(x) + (1 - \alpha) \cdot f_2(x)$.

The component selection problem was approached using this time the Pareto dominance principle to deal with the multiobjective optimization problem. In that proposal [17] the dependencies are not considered because at end all the dependencies are satisfied due to the fact that all requirements must be satisfied and the dependencies are only between them. The approach used is an evolutionary computation technique, a steady-state evolutionary algorithm as given in the Algorithm 1.

5.2 Components - based chromosome representation

The problem can be also formulated as multiobjective, involving 3 objectives [18]. The approach used is an evolutionary computation technique (denoted as $EAcP$) combined with a greedy algorithm which is used to fine tune the solutions after the application of the genetic operators.

The following three objective functions were considered: the number of remain requirements to be satisfied, fRemReq; the total cost of the components used, fCost and the number of components used, fNoComp.

All objectives are to be minimized. There are several ways to deal with a multiobjective optimization problem. In that proposal the Pareto dominance [1] principle was used.

5.3 Example and numerical experiments

A short and representative example is presented in this section. Starting for a set of six requirements and having a set of twenty available components the goal is to find a subset of the given components such that all the requirements are satisfied.

Requirements - based chromosome representation. The parameters used by the evolutionary approach are as follows: mutation probability: 0.7; crossover probability: 0.7; and number of different runs: 100.

Different experiments were considered: population size is 10 and number of iterations is 10, population size is 20 and number of iterations is 20 and population size is 50 and number of iterations is 50. While compared the third experiment with the previous experiments we noted that we are getting a lower number of different solutions while cumulation the results obtained in all the 100 runs, but the quality of these solutions is improving much more while compared with first experiment and the second one. For instance, in the first experiment the greater value for the cost objective is 83 and in the second experiment is 71 while in the third experiment this is not more than 51. So, by increasing the number of iterations and the populations size we can observe that the diversity of the final solutions is decreasing but their quality is improving very much. But we should also mention that the best solution in terms of cost (which is 46) or number of components (which is 2) is obtained in all experiments.

Components - based chromosome representation. We have performed 3 different experiments considering differ-

ent population sizes and different number of generations. For all experiments we used the same values for mutation and crossover probabilities which are: mutation probability: 0.7; crossover probability: 0.7 and number of different runs: 100.

Some of the nondominant solutions are listed below: one is dominant by the cost and the other by the number of used components:

- the solution [c5[r0, r1], c9[r3, r4], c14[r2, r5]] has the cost 46 but has 3 components and the solution [c6[r2, r3, r4, r5], c17[r0, r1]] has the cost 49 and only 2 components.

- the solution [c4[r0, r1, r3], c6[r2, r4, r5]] uses 2 components but the cost is 54 and the solution [c0[r2, r5], c9[r3, r4], c14[r2, r5]] uses 3 components and the cost is only 45.

Better results are obtained with a smaller population and a smaller number of individuals. This fact shows that these parameters are not influencing that much the final result.

6 Conclusion and future work

CBSE is the emerging discipline of the development of systems incorporating components. A challenge is how to assemble components effectively and efficiently. Component Selection Problem has been investigated in this paper. We have discussed the proposed evolutionary approaches. In relation to existing approaches we have also considered the dependencies between the requirements that have to be satisfied by the final system.

7 Acknowledgement

This material is based upon work supported by the Romanian National University Research Council under award PN-II no. ID_550/2007.

References

[1] A. Abraham, L. Jain, and R. Goldberg. *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*. Springer Verlag, London, 2005.

[2] C. Alves and J. Castro. Pore: Procurement-oriented requirements engineering method for the component based systems engineering development paradigm. In *Int'l Conf. Software Eng. CBSE Workshop, 1999.*, 1999.

[3] C. Alves and J. Castro. A systematic method for cots component selection. In *Brazilian Symposium on Software Engineering, Rio De Janeiro, Brazil*, 2001.

[4] P. Baker, M. Harman, K. Steinhofel, and A. Skaliotis. Search based approaches to component selection and prioritization for the next release problem. In *ICSM '06: Proceedings of the 22nd IEEE International Conference on Software Maintenance*, pages 176–185, Washington, DC, USA, 2006. IEEE Computer Society.

[5] I. Crnkovic and M. Larsson. *Building Reliable Component-Based Software Systems*. Artech House publisher, 2002.

[6] M. R. Fox, D. C. Brogan, and J. Paul F. Reynolds. Approximating component selection. In *WSC '04: Proceedings of the 36th conference on Winter simulation*, pages 429–434. Winter Simulation Conference, 2004.

[7] L. Gesellensetter and S. Glesner. Only the best can make it: Optimal component selection. *Electron. Notes Theor. Comput. Sci.*, 176(2):105–124, 2007.

[8] C. Grosan. *A comparison of several evolutionary models and representations for multiobjective optimization*. ISE Book Series on Real Word Multi-Objective System Engineerin, chapter 3, Nova Science,, 2005.

[9] N. Haghpanah, S. Moaven, J. Habibi, M. Kargar, and S. H. Yeganeh. Approximation algorithms for software component selection problem. In *APSEC*, pages 159–166. IEEE Computer Society, 2007.

[10] Y. Kim and O. deWeck. Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. *Structural and Multidisciplinary Optimization*, 29(2):149–158, 2005.

[11] J. Kontio. Otso: A systematic process for reusable software component selection, 1995.

[12] A. Lozano-Tello and A. Gómez-Pérez. Baremo: how to choose the appropriate software component using the analytic hierarchy process. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 781–788, New York, NY, USA, 2002. ACM.

[13] E. Mancebo and A. Andrews. A strategy for selecting multiple components. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 1505–1510, New York, NY, USA, 2005. ACM.

[14] C. Szyperski. *Component Software, Beyond Object-Oriented Programming*. ACM Press, Addison-Wesley, 1998.

[15] A. Vescan. Dependencies in component selection problem. In *5th International Workshop on Formal Aspects of Component Software (submitted)*. ENTCS, 2008.

[16] A. Vescan. An evolutionary multiobjective approach for the component selection problem. In *Proc. of the First IEEE International Conference on the Applications of Digital Information and Web Technologies (accepted)*, 2008.

[17] A. Vescan. Pareto dominance - based approach for the component selection problem. In *Proc. of the 2nd UKSim European Symposium on Computer Modeling and Simulation (submitted)*, 2008.

[18] A. Vescan and C. Grosan. A hybrid evolutionary multiobjective approach for component selection problem. In *Proc. of the 3rd International Workshop on Hybrid Artificial Intelligence Systems (accepted)*, 2008.