CREATIVE MATH. & INF. **17** (2008), No. 3, 532 - 537
Online version at `http://creative-mathematics.ubm.ro/`
Print Edition: ISSN 1584 - 286X Online Edition: ISSN 1843 - 441X

Dedicated to Professor Iulian Coroian on the occasion of his 70^{th} *anniversary*

Dependencies in the component selection problem

ANDREEA VESCAN

ABSTRACT. Component Selection is a crucial problem in Component–Based Software Engineering. We adapt a Greedy approach to construct a component-based system by introducing a new selection function. The new selection decision takes into account not only the cost of the components but also their interplay. The case study shows that considering the dependencies between the components the cost of the obtained solution may be higher due to the new selection improvement condition, but no dependencies between the selected components exist.

1. INTRODUCTION

Component–Based Software Engineering (CBSE) is concerned with designing, selecting and composing components [1]. 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.

A set of candidate components is considered. The available data about the components includes cost of acquisition and set of provided functionalities. We may have information about the dependencies between components.

In this paper, we address the problem of automatic component selection. In general, there may be different alternative components that can be selected, each coming at their own cost. 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 (view as restrictions on how the components interact).

We discuss the proposed approach as follows. Related work on the Component Selection Problem is discussed in Section 2. Section 3 introduces our approach for Component Selection Problem: Subsection 3.1 presents a formal statement of the Component Selection Problem, a Greedy approach follows in Subsection 3.2 and a new improved selection decision in Subsection 3.4. Using the example in Subsection 3.5 we discuss the new selection decision. We conclude our paper and discuss future work in Section 4.

2. RELATED WORK

Component selection methods are traditionally done in an architecture-centric manner, meaning they aim to answer the question: "Given a description of a

Received: 17.09.2008. In revised form: 27.02.2009. Accepted: 17.05.2009.
2000 *Mathematics Subject Classification*. 68N19, 68Q01.
Key words and phrases. *Component-based systems, component selection, dependencies.*

component needed in a system, what is the best existing alternative available in the market?”. Existing methods include OTSO [6] and BAREMO [7].

Another type of component selection approaches is built around the relationship between requirements and components available for use. In [5] the authors have presented a framework for the construction of optimal component systems based on term rewriting strategies. Paper [2] 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.

Another type of component selection approaches is built around the relationship between requirements and components available for use PORE [9] and CRE [8]. The goal here 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.

In relation to existing component selection methods, our approach aims to achieve goals similar to [3], [4], except we are also interested in the relationship between components, their dependencies. The [3] approach considers selecting the component with the maximal number of provided operations. The algorithm in [4] consider all the components previously 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. For the case where the inclusion of the next highest scoring feature exceeds the budget, the approach checks whether one of the following components can still be fit into the budget. A similar approach was proposed in [10]. 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.

3. Approach proposal

In CBSE the construction of cost-optimal component systems is a nontrivial task. It requires not only to optimally select components but also to take their interplay into account.

We assume the following situation: Given a repository of components and a specification of the component system that we want to construct (set of final requirements), we need to choose components and to connect them such that the target component system fulfills the specification.

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 to 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.

3.1. **Formal Statement of the Component Selection Problem.** A formal definition of the problem is as follows. Consider SR the set of final system requirements (target requirements) as $SR = \{r_1, r_2, ..., r_n\}$, and SC the set of components available for selection as $SC = \{c_1, c_2, ..., c_m\}$. Each component c_i may satisfy a subset of the requirements from SR, $SR_{c_i} = \{r_{i_1}, r_{i_2}, ..., r_{i_k}\}$. In addition $cost(c_i)$ is the cost of component c_i.

The goal is to find a set of components Sol in such a way that 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 $\sum_{c_i \in Sol} cost(c_i)$.

3.2. **Greedy approach.** Greedy techniques are used to find optimum components, either minimums or maximums in some sense. Greedy techniques typically use some heuristic or common sense knowledge to generate a sequence of sub–optimums that hopefully converge to an optimum value. Once a sub-optimum is picked, it is never changed nor is it re-examined.

A Greedy algorithm proceeds as follows: initially the set of chosen objects is empty; the selection function removes an object from the set of available objects; the new enlarged set is checked to see if the enlarged set is a solution; if the enlarged set is no longer feasible, the object is discarded and never considered again. The discarded object is not put back into the set of available objects. If the enlarged set is feasible it is permanently added to the chosen set. The process repeats itself picking a sequence of sub-optimums until either a solution is found or it is shown that no solution is feasible.

3.3. **Maximal Greedy decision selection.** The selection function is usually based on the objective function. We consider the ratio of number of requirements satisfied to the cost of the component as a measure to maximize for our heuristic decision: $|SRc_i \bigcap RSR|/cost(c_i)\ is\ maximal$.

3.4. **Improved Greedy decision selection.** The interdependencies are an important factor when considering selection of a component from a set of available components. This first selection function does not consider the relations between requirements of the selected components.

The selection function is augmented by using the dependencies between the selected components. To specify the component dependencies we introduce a dependency matrix. The dependencies between the requirements from SR are stated. We are only interested in the provided functionalities of the components that are in the set of requirements SR for the final system. We take into account only the dependencies between these requirements.

3.4.1. *Specification of the Component Requirements Dependencies.* The new selection function takes into account the dependencies and selects the components such that $|SRc_i \bigcap RSR|/cost(c_i)\ is\ maximal$ and the number of dependencies of the component c_i considered $is\ minimal$.

3.4.2. *Extra conditions.* All the dependencies of the selected c_i component must be satisfied, i.e. the requirements that the c_i component provides must have all the dependencies already satisfied.

Another condition is verified at each step when selecting a component: the requirements provided by the selected component and the requirements remained to be satisfied.

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

The set of requirements $SR = \{r_0, r_1, r_2, r_3, r_4, r_5\}$ and the set of components $SC = \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9\}$ are given. Table 1 contains for each component the provided services (in term of requirements for the final system) and the cost. The requirements dependencies are given in Table 2. We apply the Greedy algorithm and try to find an optimal solution.

Comp.	Comp. Req.	Cost
c_0	$\{r_0, r_3\}$	8
c_1	$\{r_2, r_5\}$	7
c_2	$\{r_0\}$	6
c_3	$\{r_0\}$	9
c_4	$\{r_1\}$	6
c_5	$\{r_2, r_4\}$	14
c_6	$\{r_3, r_4, r_5\}$	15
c_7	$\{r_4, r_5\}$	14
c_8	$\{r_1, r_2\}$	7
c_9	$\{r_0, r_4, r_5\}$	14

TABLE 1. Requirements

Dep.	r_0	r_1	r_2	r_3	r_4	r_5
r_0		$\surd$				
r_1						
r_2	$\surd$				$\surd$	
r_3		$\surd$				
r_4	$\surd$					
r_5		$\surd$				

TABLE 2. Dependencies

In the following we will discuss the improved selection function for the above example, first case for the first Greedy approach from Section 3.3 and then for the new selection function from Section 3.4.

3.5.1. *Maximal proportion selection decision function.* In the current section we discuss the application of the Greedy algorithm presented in Subsection 3.2 without taking into account the dependencies between the requirements of the components. The criteria for selecting the next component to be included into the solution considers only the proportion $|SRc_i \bigcap RSR|/cost(c_i)$ to be maximal.

The first selected component is c_8 with it's requirements provided to the final system r_1 and r_2. The next selected component is c_0.

The remained set of requirements to be satisfied is $\{r_4, r_5\}$. From the set of components only five may provide the $\{r_4, r_5\}$ requirements and from those only three of them have the maximum ratio value, i.e. 0.142, components c_1, c_7 and c_9. The first component is randomly selected. Four components may provide the last requirement r_4: $\{c_5, c_6, c_7, c_9\}$. Three components have the same ratio and one is randomly selected, i.e. c_5.

The final solution contains the components c_8, c_0, c_1 and c_5, components that satisfied all the requirements from the set of requirements SR. The cost of the final solution is 36, i.e. the sum of the cost of the selected components.

3.5.2. *Improved selection decision function.* The first step of the selection function is the computation of the proportion of number of requirements satisfied to the cost of the component. The component with the maximum proportion is chosen to be a part of the solution but only if it has no dependencies.

In the first iteration of the algorithm, the c_4 is the only component with no dependencies. Only one requirement is satisfied, i.e. $\{r_1\}$. Next step, only four components have the dependencies satisfied ($\{c_0, c_2, c_3, c_6\}$) and c_0 is selected because it has the maximum ratio value, i.e. 0.250.

The set of already satisfied requirements is: $\{r_0, r_1, r_3\}$. Two components (out of three with no dependencies) have the same maximum ratio value, i.e. 0.142., components c_7 and c_9. Component c_7 is randomly chosen.

In the set of remained set of requirements that must be fulfilled there is only one element, $\{r_2\}$. Three components have all the dependencies satisfied and may provide the r_2 requirement, but only two of them have the same maximal ratio value, i. e. 0.142, components c_1 and c_8. Randomly, c_8 is chosen.

The set of the remained requirements RSR is now empty and we have reached a solution with all the requirements satisfied by the selected components: c_4, c_0, c_7 and c_8. The cost of the final solution is 35, i.e. the sum of the cost of the selected components.

3.5.3. *Discussion.* The two approaches find different solutions with different final cost. Although the same solution could be found (for a proper instance of the given set of requirements, components and component costs and dependencies) the first approach may not be the right solution do to the fact that the dependencies are not considered.

Greedy Ratio	Greedy Ratio and Dependencies
$C_1 + C_0 + C_4 + C_5$ $(cost\ 35)$	$C_4 + C_0 + C_7 + C_1$ $(cost\ 35)$
$C_1 + C_0 + C_4 + C_7$ $(cost\ 35)$	$C_4 + C_0 + C_7 + C_8$ $(cost\ 35)$
$C_1 + C_0 + C_4 + C_9$ $(cost\ 35)$	$C_4 + C_0 + C_9 + C_1$ $(cost\ 35)$
$C_8 + C_0 + C_1 + C_5$ $(cost\ 36)$	$C_4 + C_0 + C_9 + C_8$ $(cost\ 35)$
$C_8 + C_0 + C_1 + C_7$ $(cost\ 36)$	
$C_8 + C_0 + C_1 + C_9$ $(cost\ 36)$	
$C_8 + C_0 + C_7$ $(cost\ 29)$	
$C_8 + C_0 + C_9$ $(cost\ 29)$	

TABLE 3. GreedyRatio and GreedyRatioAndDependencies solutions

The improvement of the selection function by using also the dependencies between the considered components helps us to compute the correct and accurate solution.

Only two of the found solutions using GreedyRatio are respecting the requirements dependencies. A solution with a minimum cost is obtained (cost 29) but is not a correct or valid solution since does not conform to the dependencies.

4. Conclusion and future work

CBSE is the emerging discipline of the development of software components and the development of systems incorporating such components. A challenge in component-based software development is how to assemble components effectively and efficiently.

Component Selection Problem has been investigated in this paper. Two different selection functions has been considered for the Greedy algorithm: the proportion of the number of requirements satisfied to the cost of the component, and for the second selection function we have considered also the dependencies between the components.

We intend to extend our approach by specifying and proving the compatibility between two connected components. The protocol for each provided operations of a component have to be specified and included into the composition process.

Another future extension is to use and define metrics for the computation of the cost of a component, taking into consideration not only acquisition cost but also quality attributes (for non-functional requirements).

References

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

[2] Haghpanah, N., Moaven, S., Habibi, J., Kargar, M. and Yeganeh, S.H., *Approximation Algorithms for Software Component Selection Problem*, in The 14th Asia-Pacific Software Engineering Conference, pp. 159-166, IEEE Press, ORASUL, 2007

[3] Fox, M.R., Brogan, D.C.G. and Reynolds, P.F., *Approximating component selection*, in Proc. 36th conference on Winter simulation, pp. 429-434, Washington, 2004

[4] Baker, P., Harman, M., Steinhofel, K. and Skaliotis, A., *Search Based Approaches to Component Selection and Prioritization for the Next Release Problem*, in The 22nd IEEE International Conference on Software Maintenance, pp. 176-185, IEEE Press, Washington, 2006

[5] Gesellensetter, L. and Glesner, S., *Only the Best Can Make It: Optimal Component Selection*, Electron. Notes Theor. Comput. Sci, 176, 105-124 (2007)

[6] Kontio, J., *OTSO: A Systematic Process for Reusable Software Component Selection*, Technical report, University of Maryland, 1995

[7] Lozano-Tello, A. and Gómez-Pérez, A., *BAREMO: how to choose the appropriate software component using the analytic hierarchy process*, in The 14th international conference on Software engineering and knowledge engineering, pp. 781-788, ACM, New York, 2002

[8] Alves, C. and Castro, J., *Cre: A systematic method for cots component selection*, in Brazilian Symposium on Software Engineering, IEEE Press, Rio De Janeiro, 2001

[9] Alves, C. and Castro, J., *Pore: Procurement-oriented requirements engineering method for the component based systems engineering development paradigm*, in Int. Conf. Software Eng. CBSE Workshop, IEEE Press, ORASUL, 1999

[10] Mancebo, E. and A. Andrews, A., *A strategy for selecting multiple components*, in 2005 ACM symposium on Applied computing, pp. 1505-1510, ACM, New York, 2005

Babeş-Bolyai University
Department of Computer Science
Kogalniceanu 1
400084, Cluj–Napoca, Romania
E-mail address: avescan@cs.ubbcluj.ro