Interactive Decision Support for Multiobjective COTS Selection

Thomas Neubauer
Secure Business Austria - Security Research
Email: neubauer@securityresearch.at

Christian Stummer
School of Business, Economics, and Statistics
University of Vienna, Austria
Email: christian.stummer@univie.ac.at

Abstract— **In the past decade component-based software engineering (CBSE) has gained considerable attention from both industry and the scientific community. Obviously, selecting the "best" combination of commercial off-the-shelf (COTS) components plays a critical role in CBSE. This task becomes demanding, as multiple objectives and constraints have to be taken into account. So far, the selection process has been tackled by such traditional techniques as the Weighted Scoring Method or the Analytical Hierarchy Process.**
This paper introduces a decision support approach that more properly addresses that challenge. The approach involves first determining (feasible) Pareto-efficient alternatives, then allowing decision makers to interactively explore the solution space until they find the most appealing solution. It not only works without extensive *a priori* preference information (such as criteria weights), it can also be easily integrated into existing COTS selection frameworks.

I. Introduction

Component-based software engineering (CBSE) plays a major role in today's software engineering research and practice [1]. It differs from traditional approaches in that it uses commercial off-the-shelf (COTS) components[1] as the basis for developing new software products. CBSE can lower the level of complexity by breaking down a large project into more manageable smaller pieces with the aim of reassembling them again later-on in order to achieve the desired solution [3].

Literature lists several factors that drive the popularity of CBSE. These include the increasing industrial competition for the delivery and provision of high quality and reliable software within short time-frames, as well as the demand for larger and more complex projects, which often cannot be accomplished by a single software development organization that has to start from scratch [4]. Furthermore, software reuse is recognized as one of the most efficient means for reducing the number of bugs (cf. corresponding empirical studies by [4], [5]) if development costs have to be kept at a comparatively low level. And finally, CBSE is stimulated by a growing number of technologies such as CORBA, JavaBeans/EJB, DCOM/ActiveX or .Net as well as by the availability of various component configuration and deployment tools [6].

On the other hand, COTS systems need to be customized with respect to individual requirements and, as a consequence, the proper selection of COTS components is critical for subsequent phases of the software development life cycle. Inefficient decisions not only affect the correctness, quality and dependability of a COTS-based application, they may also cause substantial cost increases with respect to the development process and/or future maintenance activities [1], [7], [8]. Moreover, customers will be dissatisfied if they feel that their preferences have not been given adequate consideration [9].

Several frameworks for decision support in COTS evaluation and selection have been proposed (e.g., OTSO [2]). As multiple objectives (such as costs, compatibility, ease of installation, etc.) are involved more often than not, authors propose the use of multicriteria decision-making techniques (cf. [4], [7], [9], [10], [11] and others) and typically recommend either the Analytic Hierarchy Process (AHP) or the Weighted Scoring Method (WSM) to achieve this end. However, both methods come with serious drawbacks such as the combinatorial explosion of the number of pair-wise comparisons necessary when applying the AHP, the need of extensive *a priori* preference information such as weights for the WSM, or the highly problematic assumption of linear utility functions in both cases. Further criticism is directed towards the lack of interactive decision support [4]; this criticism stems from the proposition that the decision maker should not be confronted with a single "optimized" solution, but rather should be invited to analyze and explore different scenarios and, thus, to participate in and to control the COTS selection process.

We propose a two-phase decision support approach to address the reservations and demands outlined above. The first phase involves determining COTS solution alternatives that are both feasible with respect to given constraints (such as resource limitations or dependencies between two or more components) and Pareto-efficient with respect to a number of objectives that have been identified as the most relevant ones for a given decision setting. In the second phase, decision makers are supported in interactively exploring the determined solution space until they find their individually "best" solution, i.e. that particular set of components that promises a mix of objective values that best fits their expectations. The integration of this approach into an existing COTS selection frameworks is straightforward and allows software architects as well as software project managers to investigate various scenarios and, thus, to learn about the characteristics of the

[1]Typically, 'COTS' is used to refer to software packages that have been developed or are suitable for reuse. Following Kontio [2] we use this term for all off-the-shelf software, regardless of its origin (commercial, open source, or in-house).

1530-1605/07 $20.00 © 2007 IEEE

underlying problem.

The remainder of this paper is organized as follows: after offering an overview of related literature and a discussion of the motivations behind our decision support approach in Section II, we provide a detailed step-by-step description of the new approach in Section III and then finish with conclusions and an outlook on further research.

II. Background

A COTS component is a piece of software that can be reused in software projects to become a part of other software products [12]. Given this definition, it follows that a COTS component is characterized by (i) not being tailored exclusively for a specific project and (ii) not having its further development controlled with regard to provided features and their evolution. Note that open source products have not be considered COTS products [13] until fairly recently, when the fact that both commercially available and open source components are treated the same way in practical applications has become generally recognized [12]. Further note that CBSE aims at providing complex solutions in a more efficient and effective way (e.g., within less time and/or for lower costs), but there is a serious threat of underestimating technical risks associated with the selection, evaluation, and integration of these software components, which more often than not has resulted in considerable delays as well as budget overruns for development or maintenance [14]. Although typical tasks in a CBSE framework are pretty much alike [6], [14], [15] (for a typical example cf. Section III-A), there is, so far, no generally accepted COTS selection method [1]. According to Andrews [6], frameworks can be grouped into architecture-driven and requirements-driven approaches; the first group is represented by OTSO [2], BAREMO [16], or STACE [17], while SCARLET [18] as the successor of PORE [7] and CRE [19] are of the latter type.

These approaches have in common that they posit the need to value COTS solution alternatives. As one-dimensional (usually financial) indices such as the return on investment, net present value, internal rate of return or the length of the payback period have failed to properly encompass all relevant aspects of IT investments, COTS frameworks regularly recommend AHP (Analytic Hierarchy Process) or WSM (Weighted Scoring Method) for this purpose. Both techniques assume linear utility functions and need the assignment of weights to each criterion standing for their relative importance for the decision. WSM requires decision makers to (i) have these criteria weights that perfectly represent their preferences at their fingertips without having seen any alternative solutions and (ii) to be willing to freely provide them to the system in order to calculate utility values for each component and/or each COTS solution. In contrast, the AHP is based on a hierarchical structure for consolidating information regarding alternatives; the weights are obtained through pair-wise comparisons of requirement statements that are converted afterwards to normalized rankings using their eigenvalues. While these results are supposed to be much more consistent and reliable compared with those obtained through WSM, applying the AHP method can easily result in numerous pairwise comparisons that have to be performed by the decision makers. Maiden [7] illustrates this fact by means of a case study of a project with about 130 requirements: because it would have required an estimated 42,000+ individual paired comparison scores, applying AHP was impossible in this and similar cases due to the time constraint involved. Note that this drawback may be diminished to some extent through sequentially reducing the number of COTS candidates as implemented in CAP [20].

The evaluation methods outlined above aim at characterizing COTS candidates through a single index such as the return on investment or an utility value gathered by means of WSM or AHP. Using such functional approaches is highly problematic, as they are based on the availability of a proper utility function that adequately represent the decision maker's preferences. This is unrealistic in many practical situations, because the acquisition of precise preferential information such as importance weights, substitution ratios and/or various thresholds on particular criteria is typically impossible. Reducing the number of required parameters by allowing linear utility functions only (as in the case of WSM and AHP) makes things even worse as this implies a constant "exchange" rate between, for example, reliability, security, or maintenance costs regardless of the current levels of achievement in these objectives. Moreover, total scores tend to mask individual attributes that may represent particular strengths or weaknesses — as a consequence, a high score on one attribute/dimension may offset poor performance on another [9]. For this reason, researchers and practitioners agree that traditional evaluation methods are inadequate for many cases of COTS selection [21], [22] and that stakeholders need better support for efficiently coping with a wide spectrum of potential resources, benefits and risks in selecting the most appropriate set of components. To this end, relational solution approaches from multi-criteria decision analysis come into play as they are less rigorous and do not rely as heavily on the availability of quantitative preference data.

III. The Decision Support Approach

Our approach offers a modern alternative to the WSM and AHP approaches described above. Among its salient features is that it does not require extensive *a priori* preference information and/or numerous pairwise comparisons, but rather allows the decision makers to specify their preferences gradually. To this end, solutions on the "efficient frontier" need to be identified before the actual decision making session, as all of these solution alternatives may be relevant given the absence of *a priori* information on preferences. Note that a COTS solution is considered to be "Pareto-efficient" or "nondominated" if no other solution performing at least as well in terms of all objectives and better in at least one objective exists. In the second phase of the approach, the decision maker's search effort is interactively guided towards the individually preferred solution. Before describing both phases in detail in the following section, we outline a typical COTS selection

process and indicate where our approach could be linked into the COTS selection framework.

A. COTS Selection Framework

This section describes OTSO (Off-The-Shelf Option), a representative COTS selection framework introduced by Kontio [2], that may very well serve as a basis for our interactive decision support approach. The OTSO process consists of five steps:

1) Define Criteria: In a first step, a hierarchical set of criteria is derived from reuse goals that are affected by factors such as a organization's infrastructure, application architecture and design, application requirements, project objectives and constraints, or the availability of software libraries. Establishing the criteria for COTS selection is a critical task in CBSE and one that must be performed individually for each decision instance; for a more detailed discussion on how to decompose criteria (e.g., by using GQM [23]) see [2]. More often than not, authors recommend using an experience base for refining and formulating the reuse goals. However, note that especially functional requirements are typically unique for each application. The identified evaluation criteria can be roughly grouped into four families [2]:

- Functional criteria are determined by the underlying business processes. The fulfillment of these criteria has high priority in order to secure the coverage of most or possibly even all requirements for the COTS system.
- Quality criteria comprise non-functional requirements concerning the quality of the COTS candidates, such as the defect rate, performance, usability or security. Further criteria may refer to maximum internal cohesion or minimum external coupling.
- Strategic criteria cover many non-technical issues such as product costs, available time, acceleration of business processes or employees and their qualifications.
- Domain and architecture criteria play a role if a reuse candidate is influenced by the application domain or the software architecture.

2) Searching: This step aims at identifying COTS components. The following list contains examples that have been classified with respect to their architectural level [24]:

- Operating systems: AIX, FreeBSD, Linux, Microsoft Windows Server, QNX, Solaris.
- Middleware: BEA Weblogic, IBM Websphere, OpenORB, Oracle Application Server, Visibroker and additional components such as CRM, DMS, ERP, SCM, SRM available from various vendors.
- Databases: Hypersonic SQL, Microsoft Access, Microsoft SQL Server, MySQL, Oracle DB, PostgreSQL.
- Support Software: Agent++, Apache Tomcat, Hypertext Preprocessor, IBM Java Runtime Environment, Intel COPS Client, Log4J, Telia BER Coder, WebMacro, Xerces, XMLDB.

3) Screening: The objective of the screening process is to filter components that should be evaluated in greater detail.

4) Evaluation: The evaluation process strives for (i) consolidating and evaluating the pre-selected alternatives with respect to the defined criteria and (ii) documenting these results.

5) Analysis: In the analysis phase, decision makers decide on the "best" COTS solution alternative for the application under consideration. For this purpose, the OTSO process recommends the AHP method, starting with determining the importance of criteria on each level through pairwise comparisons, then checking the consistency of rankings and revise them if necessary, and finally, evaluating alternative COTS solutions and presenting the recommendation.

B. Determining Efficient COTS Solutions

Our decision support approach is designed for the final phase of the framework described above. To this end, it substitutes the AHP approach and provides decision makers with "true" multiobjective decision support. The first task in our approach lies in determining efficient COTS solution alternatives — a task that, technically, constitutes a multiobjective combinatorial optimization (MOCO; for a survey cf. [25]) problem. Solving this problem involves identifying Pareto-efficient combinations of components where the binary variables $x_i \in \{0, 1\}$ indicate whether or not a component i is selected for the COTS product ($x_i = 1$ if so, and $x_i = 0$ otherwise). Alternatively, we also may opt for integer decision variables (i.e., $x_i \in \{1, \ldots, n_i\}$) in case of n_i mutually exclusive components (e.g., if one and only one operating system should be selected); if so, the value of the decision variable directly describes this selection (e.g., $x_1 = 3$ may refer to Linux). No matter what form has been selected, a COTS solution can be represented by a vector $x = (x_1, \ldots, x_N)$, with N being the number of components or necessary choices between components, respectively. The MOCO problem is to maximize K objectives (such as functionality, usability or supplier capability)

$$\text{maximize} \quad u_k(x) \qquad \text{for } k = 1, \ldots, K. \tag{1}$$

Note that objective functions referring to criteria that naturally should be minimized (e.g., costs) can easily be transformed by simply multiplying the underlying objective values with (-1). Further note that functions $u_k(x)$ may take any form (linear, non-linear, etc.) and also may have points of discontinuity in them as long as they are defined for all (feasible) COTS alternatives x. And finally, we recognize that finding proper functions for criteria such as the expected defect rate of a given COTS solution may be tricky: however, this difficulty also holds true for all other decision support approaches to the same degree. We will not address that problem in this paper, as doing so would go beyond the paper's scope, which focuses on the analysis phase – and not the evaluation phase – of the COTS framework. Yet, what can be done anyway is to take into account interdependencies between two or more COTS components (such as synergy or cannibalism effects) that affect the utility functions; refer to [26] for a detailed discussion and a corresponding example from the field of research and development project portfolio selection.

Any procedure applied in this phase is meant to identify (or, at least, provide an approximation of) the set of all COTS solutions that are Pareto-efficient (i.e., there is no other solution with equally good or better values in all K objectives and a strictly better value in at least one objective). Of course, all solutions taken into consideration have to be feasible with respect to two sets of constraints.

The first set relates to limited resources (e.g., development costs or maintenance costs). For binary decision variables x_i constraints may be formulated simply as

$$\sum_i r_{iq} x_i \leq R_q \qquad \text{for } q = 1, \ldots, Q, \tag{2}$$

where r_{iq} represents the amount of resources of type q required by component i and R_q stands for the maximum available amount of resources. Corresponding terms have to be added in the event of synergy or cannibalism effects that impact the total resource consumption.

The second set ensures that at most a maximum – or at least a minimum – number of components from given subsets (e.g., from each architectural level) is included in feasible solutions. For instance, a constraint may require that at least one operating system (referring to the corresponding components having assigned indices 1 to 6) must be selected and, thus, takes the form

$$\sum_{i=1}^{6} x_i \geq 1. \tag{3}$$

Depending on its size (i.e., the number of COTS components, the number of objectives as well as the form and number of interdependencies between components), the MOCO problem may be solved exactly through complete enumeration or by means of a metaheuristic procedure. Note that this type of problem is NP-difficult, i.e., the search space of potential solutions doubles with each additional component candidate. As a rule of thumb, the problem may be solved exactly within reasonable computation time as long as the number of components is somewhat below 40, although complete enumeration may be worthwhile even for higher numbers of components given that some interdependencies (e.g., exactly one alternative from each architectural level) reduce the numerical complexity. In other circumstances, complete enumeration may become too time-consuming and metaheuristic procedures should come into play. We recommend the Non-Sorting Genetic Algorithm-II, Pareto-Ant Colony Optimization or MOCO-versions of Tabu Search and Variable Neighborhood Search for that purpose (cf. [27], [28] for applications and comparisons of their performances; a survey is provided in [29]). While there is no guarantee that these procedures will actually find all efficient solutions, they usually identify the vast majority at a fraction of the runtime required by complete enumeration.

C. Interactive Exploration of Solution Space

In the second phase, the decision maker needs support in finally determining the COTS solution that best fits his/her notions out of the possibly several thousand Pareto-efficient alternatives identified in the first phase. We will propose two approaches for the problem at hand. The first is based on interactive modifications of lower and upper bounds for one or more objectives. To this end, the decision support system (DSS) starts with displaying K bars (cf. Fig. 1).

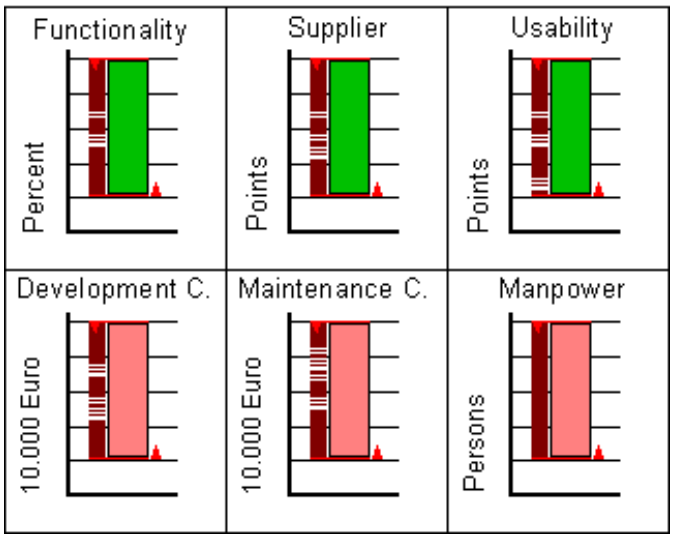

Fig. 1. Status of the DSS at the beginning

For each objective (cf. Fig. 2) the system provides information on what can be achieved by (i) the efficient COTS solutions from solution space and (ii) the subset of solutions that have remained after the decision maker has entered some aspiration levels.

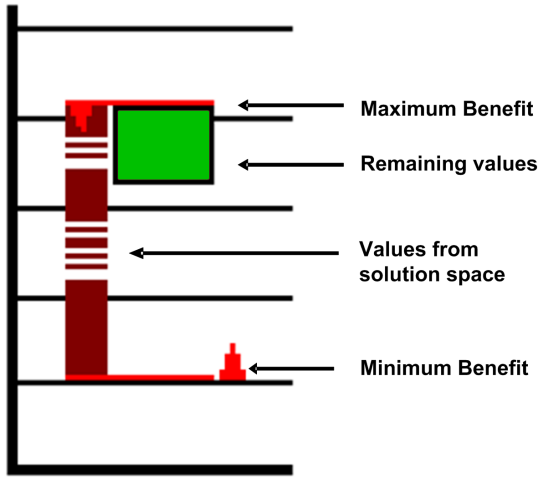

Fig. 2. Subwindow details

The first-mentioned values are represented through small dark marks on the left side that may visually grow together to vertical blocks. As objective values are independent from each other (at least to some degree), the resulting optical "segmentation" differs between the objectives. The broader, colored bars on the right side indicate the span of objective values achievable by solutions that fulfill all aspiration levels

set so far. Technically, these bars should be segmented according to the aforementioned dark marks. However, for the sake of maximum visibility they are displayed as solid bars. Two moveable horizontal lines with small triangles at one side represent lower and upper bounds. After their initialization with minimum and maximum objective values taken from the alternative COTS products in solution space, they are intended for restricting the set of remaining solutions in a step-by-step manner (e.g., by raising the minimum bound in one of the objectives) or for expanding it (e.g., by once again relaxing some bounds) according to the decision makers preferences. In all cases, the system provides immediate feedback about the consequences of such choices in terms of intervals for values achievable with the remaining alternatives. Let us illustrate this by reducing the maximum allowance for developing costs (cf. Fig. 3).

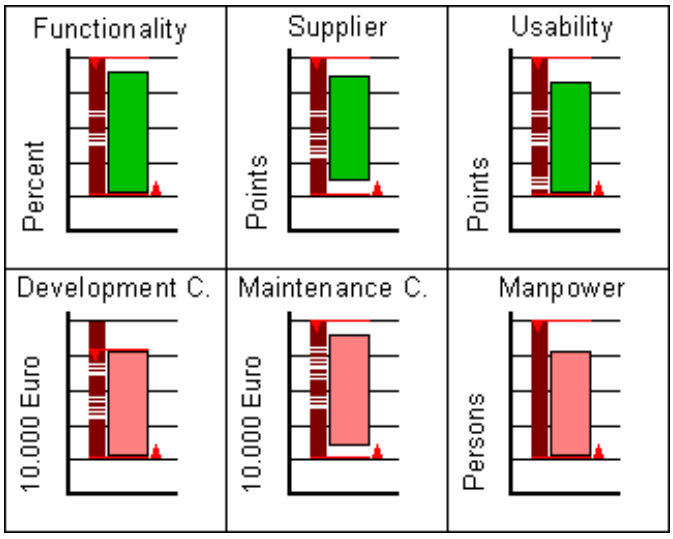

Fig. 3. Status of the DSS after one setting

Because this setting has filtered primarily those COTS solutions that come with relatively high development costs (and, on average, somewhat higher need for manpower) but low maintenance costs and high functionality, the options in the other objectives have been reduced as well and the position and size of the "flying" bars representing the updated span for the achievable objective values have changed accordingly. Raising the minimum value for functionality narrows the set of remaining alternatives even further since particularly many low-price alternatives drop out (cf. Fig 4).

In further iterations, the decision maker continues playing with minimum and maximum bounds and by doing so learns about the consequences of his/her decisions and, thus, will get a much better "feeling" for the problem in terms of what can be achieved in some objectives at what "price" in terms of opportunity costs in other objectives. After several cycles of restricting and once again expanding the opportunity set, the decision maker will finally end up with a COTS solution alternative that offers an individually satisfying compromise between the relevant objectives. Note that he/she does not

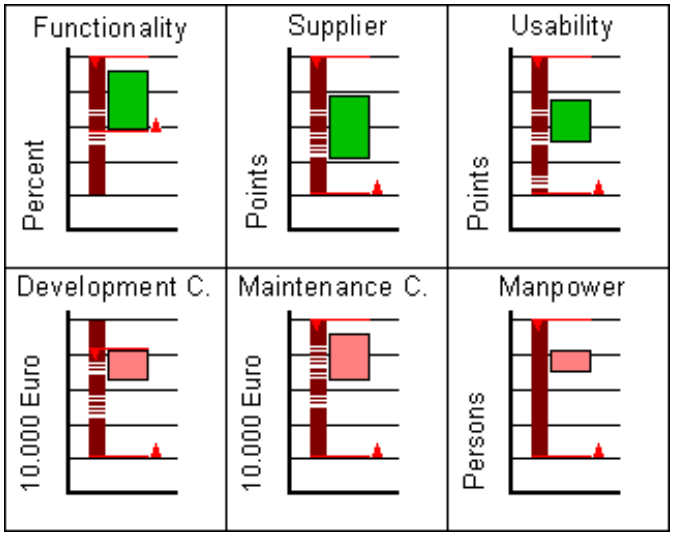

Fig. 4. Status of the DSS after two settings

need to explicitly specify weights for objectives or to state how much one solution is better than another during any stage of the whole procedure. Instead, ample information on the specific COTS selection problem is provided to him/her and the DSS ensures that the final solution will be an optimal (i.e., Pareto-efficient) one with no other feasible solution available that is "better" from an objective point of view.

The second approach for interactive decision support in multiobjective COTS selection differs from the first one insofar as the decision maker in each of several iterations chooses his/her favorite candidate from a list of about seven alternatives – which, during the whole process, become increasingly less diverse – and by doing so, implicitly provides the system with information on his/her preferences. The corresponding stepwise pruning of the solution space can be implemented by means of a cluster analysis procedure that makes it possible to partition a large set of solutions into groups of relatively homogenous elements as well as to represent each group by a single characteristic solution. Procedurally, the decision maker in each iteration receives information about the objective values of the COTS solutions representing their cluster. Through pair-wise comparisons and/or simply by looking at the objective patterns, he/she selects the alternative that best meets his/her implicit preferences and, thus, determines the region in objective space that should be explored in greater detail. A further decision cycle starts by splitting the selected group in ever smaller (sub-) sets and asking once again for the decision maker's preferred representative; this approach continues until a single solution remains. Psychological constraints limit a person's effective overview to about seven alternatives. For this reason, it is critical to fix the maximum number of clusters in each iteration in advance; this can be accomplished with the commonly used (non-hierarchical) k-means clustering.

Given this constraint, the user interface provides information on objective values for no more than seven solutions (cf. Fig. 5 for an example with four such cluster representatives).

Note that decision makers may feel that this representation is counterintuitive because lower bars are preferred to higher ones in case of objectives referring to resource consumptions (such as development costs (DC)), whereas it is the other way round in case of functionality (FU). An alternative way of symbolizing costs would therefore be to fix the corresponding bars on the top and let them hang down or to use completely different, more holistic, designs such as web diagrams (for an example cf. [30]).

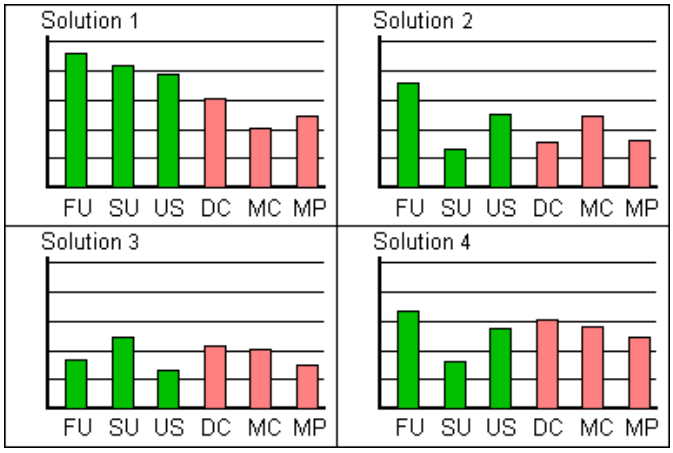

Fig. 5. Selection of cluster representatives

Once the decision maker has clicked on his/her favorite alternative, another clustering iteration starts within the set of solutions that has been represented by the selected solution and representatives of the identified new (sub-) clusters will be displayed, and so forth. Although the clusters will usually somewhat differ in their size, on average only the seventh part of alternatives will remain after each decision, thus ensuring that just a few iterations (e.g., around four and certainly less than ten for a given set of about 2,000 efficient alternatives) are necessary when applied in practice.

Both procedures outlined above have merits and, thus, there is no general recommendation which one to use. The first method provides the user with ample feedback on remaining alternatives in any of the objectives and offers the opportunity to rapidly move in objective space. Even if the decision maker may not always be fully aware of the implications when reducing the solution space, it will be a matter of a few clicks to make sure to not have missed some attractive solution alternatives (and users of the system should be encouraged to do so). This comes at the price of relatively low guidance by the system which may result in some more iterations than absolutely necessary. However, the system is designed for supporting the decision maker to learn about the characteristics of the specific COTS selection problem at hand and maybe even about his/her own preferences. In many cases that should be worth the "investment" of some extra iterations. In contrast, exploring the solution space by means of the clustering approach is much more streamlined and will guide the decision maker to a reasonable result within a few decisions. On the other hand, it becomes time-consuming to reach distant areas in objective space once some iterations have passed already.

IV. Conclusions and Further Research

Decision makers in CBSE have to select the most appropriate combination of COTS components from what is typically a large set of candidates. This task becomes demanding not only because of the possibly high number of components grouped on several architectural levels or the various interdependencies between them, but also because decision makers pursue multiple, often opposing, objectives and they are neither able nor willing to explicitly provide their "true" (aggregating) utility function to some DSS in advance. This paper has introduced a two-phase decision support approach that may be used to upgrade existing COTS frameworks like OTSO and provide them with means to more properly address the challenge of supporting decision makers in their COTS selection problem. By applying our approach, software architects and project managers can avoid some of the serious problems that are inherent to traditional procedures like the AHP or the WSM. So far, the described methodology is implemented in a customizable decision support system called Odin [31] that is about to be applied for COTS selection in the IT department of a major public utility company.

The most important limitation of our approach is found on its dependence on both the availability of proper procedures for deriving objective values (e.g., for the reliability of a COTS solution given that reliability is of relevance for the decision) and the quality of input data. This limitation is not unique for our approach and holds for all other approaches as well, but nevertheless confines the applicability of formal decision support approaches to cases with the necessary information available. It is worth noting that our practical experiences have shown that decision makers often do not keep these kinds of information.

Further research is needed in several directions. The first concerns procedures aiming at generating better data during the preceding phases (e.g., by implementing some workshop-based process dedicated for this purpose; for an example cf. [32]). In the analysis phase itself, an explicit consideration of uncertainty caused by changing system requirements, evolving of COTS products, and the ability of the selection team (cf. [4], [8]) seems to be a promising field to work on. In this respect, introducing coefficients that are random variables with approximated probability distributions could constitute an initial step. Thus, our mathematical programming model will need to incorporate the concept of stochastic dominance, which can be achieved by supplementing percentiles as additional objectives (see [33] for an example). This extension should not only make it possible to take into consideration point estimates or expected values, but should also allow for the uncertain outcomes that are typical for complex environments, as in the case of COTS selection. Next, it may be worthwhile to allow fuzzy input data (for a survey on fuzzy multiple attribute decision making see [34]). Then ideas from group

decision making and negotiation analysis might be integrated in order to elicit preferences from multiple stakeholders and to propose proper (e.g., fair) compromises. And finally, we will continue conducting experiments in order to more thoroughly test alternative user-interface designs with respect to their acceptance in real-life decision environments.

ACKNOWLEDGMENT

This work was performed at Secure Business Austria, a competence center that is funded by the Austrian Federal Ministry of Economics and Labor (BMWA) as well as by the provincial government of Vienna.

REFERENCES

[1] G. Ruhe, "Intelligent support for selection of COTS products," in *Revised Papers from the Workshop on Web, Web-Services, and Database Systems (NODe 2002)*. Springer LNCS 2593, 2002, pp. 34–45.

[2] J. Kontio, "OTSO: a systematic process for reusable software component selection," Institute for Advanced Computer Studies and Department of Computer Science, University of Maryland, Tech. Rep., 1995.

[3] J. Cheesman and J. Daniels, *UML Components: A Simple Process for Specifying Component-Based Software*. Addison-Wesley, 2001.

[4] C. Alves and A. Finkelstein, "Investigating conflicts in COTS decision-making," *International Journal of Software Engineering and Knowledge Engineering*, vol. 13, no. 5, pp. 473–495, 2003.

[5] V. R. Basili, L. C. Briand, and W. L. Melo, "How reuse influences productivity in object-oriented systems," *Communications of the ACM*, vol. 39, no. 10, pp. 104–116, 1996.

[6] A. A. Andrews, A. Stefik, N. Picone, and S. Ghosh, "A COTS component comprehension process," in *Proceedings of the 13th International Workshop on Program Comprehension (IWPC05)*. IEEE, 2005, pp. 135–144.

[7] N. A. Maiden and C. Ncube, "Acquiring COTS software selection requirements," *IEEE Software*, vol. 15, no. 2, pp. 46–56, 1998.

[8] G. Ruhe, "Software engineering decision support: a new paradigm for learning software organizations," in *Proceedings of the 4th International Workshop on Advances in Learning Software Organizations (LSO 2002)*. Springer LNCS 2640, 2003, pp. 104–113.

[9] C. Ncube and J. C. Dean, "The limitations of current decision-making techniques in the procurement of COTS software components," in *Proceedings of the First International Conference on COTS-Based Software Systems (ICCBSS 2002)*. Springer LNCS 2255, 2002, pp. 176–187.

[10] F. Navarrete, P. Botella, and X. Franch, "How agile COTS selection methods are (and can be)?" in *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA05)*. IEEE, 2005, pp. 160–167.

[11] T. Wanyama and B. Homayoun, "Towards providing decision support for COTS selection," in *Proceedings of the 18th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE 2005)*. IEEE, 2005, pp. 908–911.

[12] M. Torchiano and M. Morisio, "Overlooked aspects of COTS-based development," *IEEE Software*, vol. 21, no. 2, pp. 88–93, 2004.

[13] G. Lawton, "Open source security: opportunity or oxymoron?" *Computer*, vol. 35, no. 3, pp. 18–21, 2002.

[14] V. Tran, D.-B. Liu, and B. Hummel, "Component-based systems development: challenges and lessons learned," in *Software Technology and Engineering Practice*. IEEE, 1997, pp. 452–462.

[15] A. W. Brown and K. C. Wallnau, "The current state of CBSE," *IEEE Software*, vol. 15, no. 5, pp. 37–46, 1998.

[16] A. Lozano-Tello and A. Gomez-Perez, "BAREMO: how to choose the appropriate software component using the Analytic Hierarchy Process," in *Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering*, ACM Proceedings 27, 2002, pp. 781–788.

[17] D. Kunda, "STACE: social technical approach to COTS software evaluation," in *Component-Based Software Quality: Methods and Techniques*, Springer LNCS 2693, 2003, pp. 64–84.

[18] N. Maiden, H. Kim, and C. Ncube, "Rethinking process guidance for selecting software components," in *Proceedings of the First International Conference on COTS-Based Software Systems (ICCBSS 2002)*, Springer LNCS 2255, 2002, pp. 151–164.

[19] C. Alves and J. Castro, "CRE: a systematic method for COTS components selection," in *Proceedings of the XV Brazilian Symposium on Software Engineering*, 2001.

[20] M. A. Ochs, D. Pfahl, G. Chrobok-Diening, and B. Nothhelfer-Kolb, "A COTS acquisition process: definition and application experience," in *Proceedings of the 11th ESCOM Conference*, Shaker, 2000, pp. 335–343.

[21] M. Martinsons, R. Davidson, and D. Tse, "The balanced scorecard: a foundation for the strategic management of information systems," *Decision Support Systems Journal*, vol. 25, no. 1, pp. 71–78, 1998.

[22] S. D. Ryan and M. S. Gates, "Inclusion of social sub-system issues in IT-investment decisions: an empiricial assessment," *Information Resources Management Journal*, vol. 17, no. 1, pp. 1–18, 2004.

[23] V. Basili and H. Rombach, "Tailoring the software process to project goals and environments," in *Proceedings of the 9th International Conference on Software Engineering*. IEEE, 1987, pp. 345–357.

[24] M. Morisio and M. Torchiano, "Definition and classification of COTS: a proposal," in *Proceedings of the First International Conference on COTS-Based Software Systems (ICCBSS 2002)*. Springer LNCS 2255, 2002, pp. 165–175.

[25] M. Ehrgott and X. Gandibleux, "A survey and annotated bibliography of multiobjective combinatorial optimization," *OR Spectrum*, vol. 22, no. 4, pp. 425–460, 2000.

[26] C. Stummer and K. Heidenberger, "Interactive R&D portfolio analysis with project interdependencies and time profiles of multiple objectives," *IEEE Transactions on Engineering Management*, vol. 50, no. 2, pp. 175–183, 2003.

[27] C. Stummer and M. Sun, "New multiobjective metaheuristic solution procedures for capital investment planning," *Journal of Heuristics*, vol. 11, no. 3, pp. 183–199, 2005.

[28] K. Doerner, W. Gutjahr, R. Hartl, C. Strauss, and C. Stummer, "Nature-inspired metaheuristics for multiobjective activity crashing," *Omega*, forthcoming.

[29] M. Ehrgott and X. Gandibleux, "Approximative solution methods for multiobjective combinatorial optimization," *Top*, vol. 12, no. 1, pp. 1–63, 2004.

[30] E. Kasanen, R. Östermark, and M. Zeleny, "Gestalt system of holistic graphics: new management support view of MCDM," *Computers & Operations Research*, vol. 18, no. 2, pp. 233–239, 1991.

[31] T. Neubauer, "Odin: a tool for interactive decision support in the presence of multiple objectives [in German]," *Technical Report*, Information & Software Engineering Group, Technical University of Vienna, 2006.

[32] T. Neubauer, C. Stummer, and E. Weippl, "Workshop-based multiobjective security safeguard selection," in *Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06)*. IEEE, 2006, pp. 366–373.

[33] A. Medaglia, S. Graves, and J. Ringuest, "A multiobjective evolutionary approach for linearly constrained project selection under uncertainty," *European Journal of Operational Research*, forthcoming.

[34] S.-J. Chen and C.-L. Hwang, *Fuzzy Multiple Attribute Decision Making: Methods and Applications*. Springer, 1992.