Journal of Universal Computer Science, vol. 14, no. 8 (2008), 1228-1255
submitted: 15/11/07, accepted: 25/3/08, appeared: 28/4/08

Experimenting the Automated Selection of COTS Components Based on Cost and System Requirements

Vittorio Cortellessa
(Dipartimento di Informatica
Università dell'Aquila
Via Vetoio, 1, Coppito (AQ), 67010 Italy
cortelle@di.univaq.it)

Ivica Crnkovic
(Department of Computer Science and Electronics
Mälardalen University
PO Box 883, SE-721 23 Västerås, Sweden
ivica.crnkovic@mdh.se)

Fabrizio Marinelli
(DIIGA, Università Politecnica delle Marche
via Brecce Bianche, 1
60131 Ancona, Italy
marinelli@diiga.univpm.it)

Pasqualina Potena
(Dipartimento di Scienze
Università "G.D'Annunzio"
Viale Pindaro, 42, Pescara, 65127 Italy
potena@sci.unich.it)

Abstract: In a component-based development process the selection of components is an activity that takes place over multiple lifecycle phases that span from requirement specifications through design to implementation and integration. In different phases, different assumptions are valid and different granularity of information is available, which has a consequence that different procedure should be used in the selection process and an automated tool support for an optimized component selection would be very helpful in each phase. In this paper we analyze the assumptions and propose the selection procedure in the requirements phase. The selection criterion is based on cost minimization of the whole system while assuring a certain degree of satisfaction of the system requirements that can be considered before designing the whole architecture. For the selection and optimization procedure we have adopted the DEER (DEcision support for componEnt-based softwaRe) framework, previously developed to be used in the selection process in the design phase. The output of DEER indicates the optimal combination of single COTS (Commercial-Off-The-Shelf) components and assemblies of COTS that satisfy the requirements while minimizing costs. In a case study we illustrate the selection and optimization procedure and an analysis of the model sensitivity to changes in the requirements.

Key Words: COTS selection, optimization model, software requirements
Category: D.2.1

1 Introduction

In a component-based development (CBD) process, one of the crucial and specific activities is component selection. The component selection is an activity in which components are assessed in order to estimate appropriateness of their inclusion into the system. Such a process may result in selecting several candidates or a particular component, or in finding that some desired functionalities cannot be found in existing components and the functionalities should be implemented in a new component. The component selection is not performed only in the implementation phase, when a component can be selected to replace the ones identified in the design phase. If the component selection would be limited to the implementation phase, then it would be difficult to find pre-existing components (in particular COTS (Commercial-Off-The-Shelf) components) that exactly match the software specifications. For this reason, the component selection is considered earlier in the development process. This activity in practice starts in the requirements elicitation phase and then is refined along the development process [Crnkovic et al. 2005]. The selection process will, however, have a different character and different goals in different phases.

One of the main objectives that traverses each phase of a development process is to select the "best" set of (COTS and in-house) components, but with slight difference of goals in each phase. In the requirements phase, the goal is to find appropriate candidates that might be integrated into the system. In the design and implementation phase, particular components are selected and verified/tested in isolation and in combination with other (selected) components. A combination that best fits the goals will be selected. In the maintenance phase, there will be requirements to replace a particular component (or set of components) while keeping other components unchanged. While coming one after the other the phases of a development process, the developers of the system get more information about the features required for the system. In all phases, the choice may have significant consequences on the development costs and the final product quality. Therefore, the tools that support decisions strictly related to meet functional and non-functional requirements, while keeping the costs within a predicted budget, would be very helpful to the software developers' tasks.

Since a pre-selection of components at the requirements phase, in particular related to a combination of functional and non-functional requirements, can sensibly improve the efficiency of the whole process [Gokhale 2004] and decrease the development costs, we have developed the DEER (DEcision support for componEnt-based softwaRe) framework [Cortellessa et al. 2007]. Basing on an optimization model, DEER helps to check if the requirements that do not depend only on the architecture of the system [IBM] are satisfied by a COTS component or an assembly of COTS components. The requirements that do not depend only on the architecture are the ones for which it is possible to provide a first estimation of their satisfaction without the architecture of the system. For example, it is possible to check if an anti-virus tool is available on the market, but, as we say in Section 2.1, it is impossible to estimate the reliability of the whole system without basing on the software architecture. An assembly is meant to be as a set of COTS components that interact with each other. In fact, in [Alves et al. 2005] it is claimed that, depending on the complexity of the application domain and the scope of available products, it is possible to find different COTS solutions ranging from a single, large package or several specific packages that once integrated will provide the desired capabilities. In the COTS selection activity, it is sometimes necessary to decide whether in-house developed software has to be maintained or replaced. DEER could support these decisions by replacing some COTS components by in-house developed components. Besides, to develop a component-based system, Open Source Software (OSS) components could be adopted. DEER could also support the selection of such components by replacing some COTS components by OSS components. DEER could provide the optimal combination of single (COTS, OSS and in-house) components and assemblies of (COTS and/or OSS and/or in-house) components that minimize the costs of construction of the system while satisfying (to a certain extent) the functional and non-functional requirements.

This paper extends the work in [Cortellessa et al. 2007] by providing a more detailed presentation of the DEER framework. In particular, we describe the DEER structure and the practical usage of the DEER framework by applying it to an example. We show how it can support the analysis of model sensitivity to changes in the requirements. Besides, we report the formulation of the optimization model that is generated and solved by DEER. We have chosen to adopt optimization techniques because they resolve some drawbacks of the Analytic Hierarchy Process (AHP) and the Weighted Scoring Method (WSM) that are typically used by the component selection approaches (see Section 6).

This paper is organised as follows. Section 2 gives a background about a component selection process within a product development process. Section 3 provides the formulation of the optimization model that represents the DEER core. Section 4 introduces the DEER structure and underlying mechanisms. Section 5 illustrates the usage of DEER in the development of a mail server. Section 6 introduces related work and discusses the novelty of our contribution. Section 7 presents the conclusions.

2 Background

2.1 Component Selection in a Development Process

Figure 1 shows the component selection process in different phases of a software lifecycle process (i.e. in requirements, design, implementation and maintenance

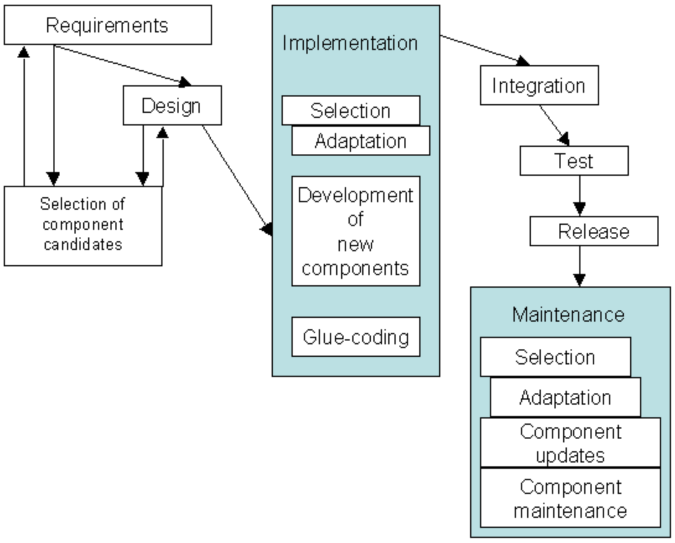

Figure 1: Component-based Waterfall Software product lifecycle.

phase).

Requirements Phase. In the requirements phase, certain components can be selected as candidates for the integration when they meet certain (typically functional) requirements. Since the satisfaction of several (functional and non-functional) requirements depends also on the software architecture, it is obvious that the final selection takes place in a later phase. However, the selection of particular candidates can have influence on the design, or it even may lead to changes and/or refinements of the requirements [Maiden and Ncube 1998]. Also, a pre-selection of components that comply to a particular architectural style may have impact onto the whole system architecture. The estimation of the requirement fulfillment may be somehow inaccurate in the requirement phase. It may be necessary to estimate again the satisfaction of all the requirements when the software architecture will be in place. For example, it is impossible to estimate the reliability of the whole system without basing on the software architecture [Trivedi 2001]. However, a pre-selection of components at requirement phase can sensibly improve the efficiency of the whole process and decrease the development costs. In fact, this is the phase for which we propose a support for components selection.

Design Phase. The design phase of component-based systems follows the same pattern as a design phase of software in general. It starts with a conceptual design providing the system overall architecture, and continues with the detailed design. Architectural components are identified in the system architecture by taking into account existing components. The selected components are not necessarily the same components that may have already been considered in the requirements phase, but they could be other components and assemblies of the existing components. As in the requirements specification process, the trade-off between desired properties and feasible design using the existing components should be analyzed. To achieve this goal, many assumptions should be taken into consideration. For example, it should be decided the component model(s) to be used, which will have impact on the architectural framework as well as on certain system quality properties. The selection process should ensure that appropriate components are selected with respect to system functional and non-functional properties. This requires the verification of the component specification, or the testing of some of the components properties. In addition, it is a well known fact [Wallnau 2002] that even if isolated components correctly work, an assembly of them may fail, due to invisible dependencies and relationships, such as shared data and resources. This implies that assemblies of components should be tested before they are integrated into the system. These efforts should be taken into account when estimating the costs of integration of existing components.

Implementation Phase. In the implementation phase the components should be integrated into the system. The adaptation of components may be required in order to avoid architectural mismatches (such as incompatible interfaces), or to ensure particular properties. There are several known adaptation techniques, such as parameterized interface, wrappers and adapters. Also, a selection of the connectors that connect the components could be performed [Liang and Luqi 2003], for example, connectors could be implemented using the "Off-The-Shelf" middlewares [Dashofy et al. 1999]. The adaptation effort should also be considered when estimating the implementation costs.

Maintenance Phase. A component-based system should be maintained for several reasons [Voas 1998], including frozen functionalities and incompatible upgrades. The former could occur when the vendor of a COTS component stops supporting the component, for example, if a COTS component need periodic updates there could exist some problems. The latter could occur when the vendor of a COTS component updates the component with upgrades that are incompatible with the own system. The paradigm of the maintenance process is similar to the one for the development: find a proper component, test it, adopt it if necessary, and integrate it into the system. In the work by Vidger et al. [Vidger and Dean 2000], the activities of the maintenance phase of a COTS-based system are presented. For example, the component reconfiguration activity consists in replacing, adding, and deleting components within the system. A certain component could be deleted or replaced, for example, if the requirements of the system evolve or if the vendor of the component releases an updated version. Hence the component selection is a peculiar activity of the maintenance phase as well.

2.2 Common steps of a Component Selection Process

As remarked by the work of Mohamed et al. [Mohamed et al. 2007/b], the processes of COTS selection have the following common steps that can be iterative and overlapping.

Step 1. The evaluation criteria are defined using the stakeholders' requirements and constraints.

Step 2. The research of COTS components is accomplished.

Step 3. The identified COTS components are filtered with "must have" requirements in order to define a short set of most promising components which are to be evaluated in more detail.

Step 4. The COTS components identified in the previous step are evaluated.

Step 5. The output of the previous step is analyzed and the COTS component that has the best fitness with the criteria is selected.

DEER supports the fifth step of a process of COTS selection. In fact, upon getting as input a set of COTS components and assemblies and the system requirements, it provides a solution that determines the COTS components to select in order to minimize the costs of construction of the system while satisfying (to a certain extent) the functional and non-functional requirements.

3 Optimization Model Formulation

In this section we introduce the mathematical formulation of the optimization model that represents the core of the DEER tool. The solution of this optimization model determines COTS components to select in order to minimize the cost of construction of the system while assuring a certain degree of satisfaction of the requirements.

3.1 A premise on the requirement satisfaction

In order to quantify the degree of satisfaction of a requirement by a COTS component, it is necessary to adopt a methodology of comparison.
In [Alves et al. 2005], using the outlines of the *goal-oriented requirements engineering* [Lamsweerde 2001] that defines requirements as goals, the authors present a goal-based approach to match the goals of the system and the features of the COTS components. They provide a methodology to derive the *operational goals* that represent the requirements for which it is possible to quantify the degree of satisfaction required. Assuming that an *operational goal* expresses a condition over a quality factor, they define the *satisfaction function* of the *operational goal* g with respect to its quality factor q as follows:

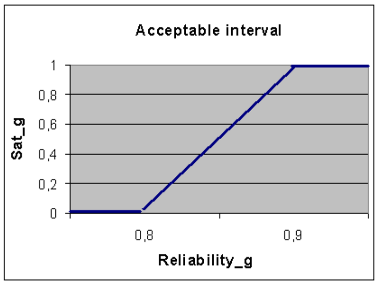

Figure 2: Acceptable Interval for the goal g

$$Sat_g : M \rightarrow [0, 1] \quad (1)$$

where M is the set of values that the quality factor q of g may take. Besides, they quantify how a COTS component satisfies an *operational goal* in the following way. Let x be an element of the set M and $Sat_g(x)$ be the degree of satisfaction of the goal g whit respect to x. Let C be a COTS component available in the market. Upon determining the value of the quality factor q of g in C, represented by C_q, and using the satisfaction function of the goal g, they define the satisfaction degree that the COTS component C meets with the goal g as $Sat_g(C_q)$.

In the following we provide an example of the application of this approach.

Let us assume that we have to develop a system that provides the service *serv*. Let g be the goal "The reliability of the service *serv* should be greater than or equal to 0.9". Let us assume that the acceptable interval for the goal ranges from 0.8 to 0.9 (see Figure 2). Let C be a COTS component that provides the service *serv* and let 0.92 be the value of the reliability guaranteed from the component C. Therefore, the degree of satisfaction that the COTS C meets with the goal g (i.e. $Sat_g(0.92)$) is equal to 1. Then, we can conclude that the COTS C satisfies completely the goal g.

3.2 The Problem Formulation

Let $G = \{g_1, \ldots, g_n\}$ be a set of n *operational goals*, each of which represents either a functional or a non-functional requirement of the system.

Let $C = \{C_1, \ldots, C_{|C|}\}$ be a set of COTS components available on the market.

Let $\mathcal{A} = \{A_1, \ldots, A_{|\mathcal{A}|}\}$ be a family of assemblies of COTS components. To reduce the search space of the model, we assume that each element A_j of $\mathcal{A}$

is a set consisting of one or more COTS components which satisfy at least a functional requirement g_k of the system.

3.2.1 Model Parameters

Let c_i be the purchase cost, typically provided by the vendors, of the i-th COTS component.

Let $\bar{c}_j$ be the cost of the j-th assembly, i.e., the cost of integration and adaptation needed to build the assembly A_j.

The estimate of the cost $\bar{c}_j$ is outside the scope of this paper, however, the adaptation cost c^{adapt} could be estimated in the following way. Consider the assembly A_j $(1 \leq j \leq |\mathcal{A}|)$ and the *operational goal* g_k $(1 \leq k \leq n)$. The cost of adaptation of A_j in order to satisfy g_k can be estimated by using the following expression:

$$c^{adapt} = \max(0, Sat_{g_k} - Sat^j_{g_k}) \cdot \bar{A}_j \qquad (2)$$

where Sat_{g_k} represents the degree of satisfaction required for the goal g_k, $Sat^j_{g_k}$ represents the satisfaction degree with which the assembly A_j meets the *operational goal* g_k and $\bar{A}_j$ is an integer number that depends on the features of the assembly A_j with respect, for example, to the values of the development process parameters (e.g. experience and skills of the developing team).

To estimate Sat_{g_k} and $Sat^j_{g_k}$ the approach presented in [Alves et al. 2005] can be adopted. If an assembly of more COTS components is not available on the market, its quality attributes can be predicted by combining the ones of the single components with the features of the architecture of the assembly [Crnkovic and Larsson 2004] (see, for example, the approach presented in [Yacoub et al. 1999] for estimating the reliability of a component-based system). To build the architecture of an assembly, it is necessary to make some assumptions (e.g. to choose an architectural style) that could be different from the ones who will be adopted for the architecture of the system. In fact, DEER does not support the making of the best architectural decisions, but, as we say in Section 1, our framework can check whether the requirements are satisfied by a COTS component or an assembly of COTS components.

Besides, the cost of adaptation of A_j could be estimated by considering other factors, such as the cost needed to resolve inconsistencies [Nuseibeh et al. 2001] between the specifications of the COTS components that belong to A_j.

For solving a mismatch between a requirement and a COTS component or an assembly of COTS components, different actions are possible. The approach presented in [Mohamed et al. 2007/a] could be adopted, which supports the resolution of mismatches during and after a COTS selection process. It is based on an optimization model that provides the best set of resolution actions needed to resolve a set of mismatches.

With regard to the integration cost, the components that belong to an assembly should support the style of interaction of the assembly architecture. If a COTS component has a different style of interaction, then developers have to introduce glueware to allow correct interactions.
Yakimovich et al. in [Yakimovich et al. 1999] suggest a procedure for estimating the integration cost. They list some architectural styles and outline their features with respect to a set of architectural assumptions. They define a vector of variables, namely the interaction vector, where each variable represents a certain assumption. An interaction vector can be associated either to a single COTS component or to an assembly. To estimate the integration cost they suggest to compare its interaction vector with the interaction vector of the software architecture.

When in-house and OSS components are considered for the construction of the system (Section 1), the parameters of the model could be estimated in a different way. For example, the purchase cost of an in-house instance, as suggested by [Cortellessa et al. 2006], could be intended as the per-day cost of a software developer. It typically depends on the skills and experience required to develop the component. There are well-assessed cost/time models to estimate it (e.g. COCOMO [Boehm 1981]). By contrast, an OSS component is acquired for free. The adaptation and integration costs of an in-house instance should be estimated as the cost needed to adapt the in-house components and that one needed to integrate them with (COTS, OSS and in-house), whereas if the OSS components are considered as black box, the adaptation and integration costs could be estimated as the ones of the COTS components, otherwise they could be estimated by considering other factor, such as the one needed to modify the source code of the components. In fact, while selecting OSS components, as remarked in [Di Giacomo 2005], the evaluation criteria based on factors, e.g. the features of the licence of the component, are often different from the ones used for the COTS components. In [Di Giacomo 2005] the activities needed to acquire, integrate and maintain OSS components are discussed by highlighting their common points and differences with the COTS components.

The relationship between the *operational goals* and the assemblies is described by a matrix $SAT(|\mathcal{A}| \times n)$, where the element $SAT[j,k]$ is equal to 1 if the k-th goal is satisfied by the j-th assembly, and 0 otherwise.

For each non-functional requirement, if the satisfaction degree with which an assembly meets the corresponding goal is greater than or equal to the satisfaction degree required for that goal then the entry that corresponds to the goal and the assembly is set to 1.

Since we assume that each element of the assembly set $\mathcal{A}$ satisfies at least a functional requirement (see Section 3.2), for each row of SAT there exist at least an entry equal to 1. Clearly, in order to obtain a feasible solution, for each non-functional requirement there should exist at least an assembly satisfying it, i.e., for each column of SAT there should exist at least an entry equal to 1.

3.2.2 Model Variables

The variables of our optimization model are:

$$x_i = \begin{cases} 1 \text{ if the } i\text{-th COTS components available} \\ \quad \text{on the market is chosen } (1 \leq i \leq |C|) \\ 0 \text{ otherwise} \end{cases}$$

$$y_j = \begin{cases} 1 \text{ if the } j\text{-th assembly is} \\ \quad \text{chosen } (1 \leq j \leq |\mathcal{A}|) \\ 0 \text{ otherwise} \end{cases}$$

3.2.3 Objective Function and Constraints

The objective function consists in minimizing the total costs, i.e., the sum of the costs of the selected components and assemblies, under the condition that each *operational goal* g_k must be satisfied by at least an assembly. We can formulate the problem as follows:

$$\min \quad \sum_{i=1}^{|C|} c_i x_i + \sum_{j=1}^{|\mathcal{A}|} \bar{c}_j y_j \tag{3}$$

$$\sum_{j=1}^{|\mathcal{A}|} SAT[j,k] y_j \geq 1 \qquad \forall k = 1 \ldots n$$

$$y_j \leq x_i \qquad \forall j = 1 \ldots |\mathcal{A}|, \forall i : C_i \in A_j$$

$$x_i \in \{0,1\} \quad \forall i = 1 \ldots |C|$$

$$y_j \in \{0,1\} \quad \forall j = 1 \ldots |\mathcal{A}|$$

Note that the second set of constraints (i.e. $y_j \leq x_i$) is to guarantee that if an assembly is part of the solution then all COTS components making the assembly will be part of it too.

It is easy to see that the component selection problem here addressed is NP-complete since it boils down to a *set covering* model as soon as purchase costs are set to zero. Another computational difficulty consists in the fact that in principle the size of model (3) grows exponentially in the number of COTS components. From the practical point of view, however, the model can be solved by means of

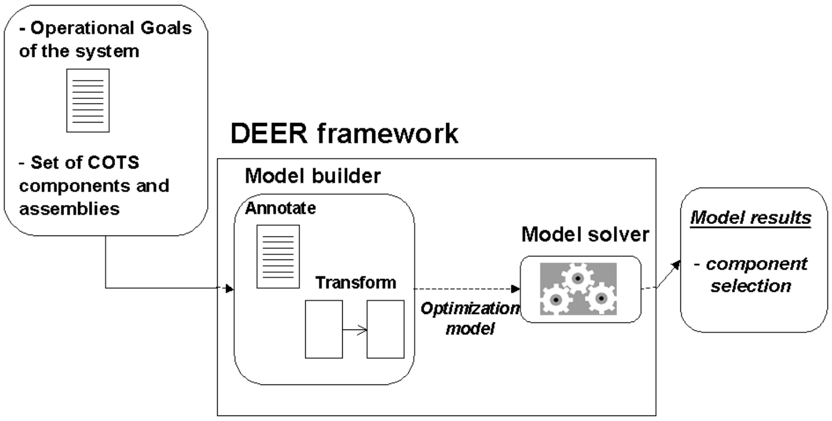

Figure 3: The DEER framework and its environment

standard optimization tools within a reasonable computation time since, quite often in practice, the set A consists of few elements. In those cases where the computation time becomes too big, heuristic approaches based on algorithms for set covering (see cap. 6 in [Reeves 1993]) or more general metaheuristic techniques (e.g. the tabu-search algorithm, [Blum and Roli 2003]) can be used for solving the model.

With regard to the accuracy of the model, the input parameters (i.e. the cost and satisfaction degree with which the assembly A_j meets the operational goal g_k) may be characterized by a not negligible uncertainty (e.g. a range for the costs may be available [Kazman 2001]). The propagation of this uncertainty should be analyzed, but it is outside the scope of this paper. Several methods to perform this type of analysis can be found, e.g. it has been done in [Goseva-Popstojanova and Kamavaram 2002] for a reliability model.

4 The DEER Framework

Figure 3 shows the DEER framework within its working environment.

The input of the framework is represented by: (i) a set of *operational goals*, each of which representing either a functional or a non-functional requirement of the system, (ii) a set of COTS components and assemblies.

The DEER framework is made of two components, which are a model builder and a model solver. The model builder first allows to annotate the *non-functional operational goals* with additional data that represent the optimization model parameters. For each component and/or assembly that satisfies a goal it is possible to enter the satisfaction degree with which it meets the goal (see Section 3.2). Then the model builder transforms the annotated model into an optimization model in the format accepted by the solver. The model solver processes the optimization model received from the builder and produces the results that consist in the selection of COTS components that minimize the costs of development of the system while assuring a certain degree of satisfaction of its functional and non-functional requirements. The optimization model solver that we have adopted is LINGO [LINGO].

However, metaheuristic techniques [Blum and Roli 2003] can be used when the model becomes too big. These algorithms do not assure neither the optimality nor the feasibility (i.e., they could not terminate if no solution exists), but they are very efficient and permit to find optimal or near-optimal solutions. Note that another model solver based on metaheuristics can be easily adopted without essentially changing the overall DEER structure. In fact, only its integration with the model builder would be affected.

In the DEER framework, the model builder has been integrated with the model solver using the same technique adopted in [Cortellessa et al. 2006]. Figure 4 shows the GUI of the model builder. The interface can be partitioned in 4 areas: (i) the *operational goal* area (upper right side of Figure 4), where the *functional operational goals* are shown, and where the *non-functional operational goals* can be selected for annotations; (ii) the *non-functional operational goals* annotation area, where the model parameters related to them can be entered, that are: the required satisfaction degree for the selected goal and, for each assembly that satisfies the selected goal, the satisfaction degree with which the assembly meets it (lower right side of Figure 4); (iii) the COTS annotation area, where the purchase cost of each COTS component can be entered (upper left side of Figure 4); (iv) the assembly annotation area, where the model parameters related to each assembly can be entered (lower left side of Figure 4), that are: its cost (i.e. the cost of integration and adaptation needed to build the assembly), the list of COTS components that define the assembly, and the *functional and non-functional operational goals* satisfied by the assembly. The four ellipses of Figure 4, respectively, from the top to the bottom of the figure, highlight: the button to run the model solver LINGO, the titles of the areas where COTS components and *operational goals* parameters can be entered, and the title of the area where the parameters of the assemblies can be entered.

5 An example: a mail server

In order to show the practical usage of the DEER framework, in this section we apply it to an example. We have considered the "mail server" example used in [Alves et al. 2005], and readers interested to the application details that we do not provide here can refer to [Alves et al. 2005]. Shortly, a bank has to decide if upgrading the own mail server with a new version, or adopt another mail server.

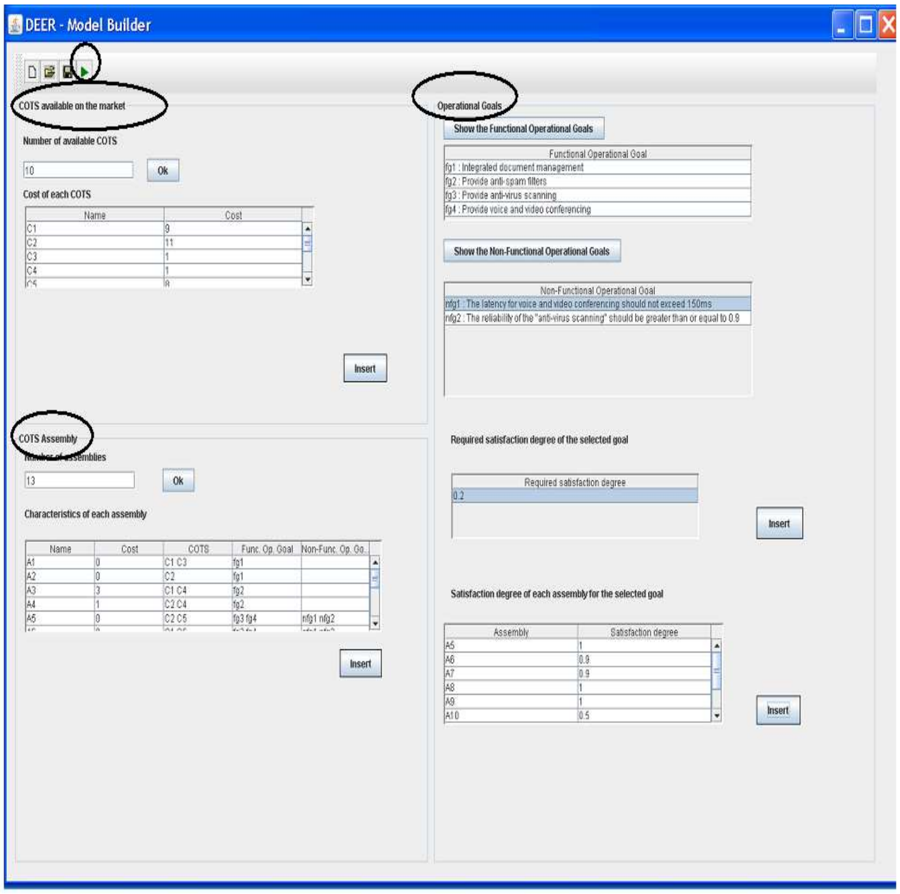

Figure 4: The GUI of the DEER framework model builder.

We have assumed that the mail server has to guarantee the *functional operational goals* listed in Table 1 (where the x-th goal is denoted as fg_x) and the *non-functional operational goals* listed in Table 2 (where the x-th goal is denoted as nfg_x).

Table 3 shows a list of available COTS components along with their buying cost c_i (in KiloEuros, KE). Besides, Table 4 shows the list of available assemblies along with their parameter values. In particular, for each assembly: the second column represents its cost $\bar{c_j}$ (i.e. integration and adaptation cost in KiloEuros, KE); the third column represents the list of COTS components that define the assembly; in the fourth column the *functional operational goals* satisfied by each

	Functional Operational Goal
fg_1	*Integrated document management*
fg_2	*Provide anti-spam filters*
fg_3	*Provide anti-virus scanning*
fg_4	*Provide voice and video conferencing*

Table 1: Mail server: Functional Operational Goals.

	Non-Functional Operational Goal
nfg_1	*The latency for voice and video conferencing should not exceed 150ms*
nfg_2	*The reliability of the anti-virus scanning should be greater than or equal to 0.9*

Table 2: Mail server: Non-Functional Operational Goals.

assembly are listed; in the fifth and sixth columns the satisfaction degrees with which each assembly meets the nfg_1 and nfg_2 goals, respectively, are reported.

Figures 5 and 6 report the acceptable intervals and the satisfaction functions (estimated using the results in [Alves et al. 2005]) of the goals nfg_1 and nfg_2, respectively.

Given the above set of parameters, we have conducted two experiments that differ for a *functional operational goal*.

5.1 First Experiment

In order to show how DEER allows to analyze the cost of construction of the system while varying the targeted satisfaction degree for each requirement, we have solved the optimization model for multiple values of the satisfaction degree required for the nfg_1 and nfg_2 goals. Figure 7 shows the results where each bar represents the minimum cost of the whole system for a given value of the satisfaction degree required for the goal nfg_2 and a given value of the satisfaction degree required for the goal nfg_1. Both the former and the latter span from 0.2 to 1.

Table 5 shows some details of this experiment results and it is organized as follows: each row represents a value of the satisfaction degree of the goal nfg_2 (i.e. the satisfaction degree required for the reliability of the anti-virus scanning), each column represents a value of the satisfaction degree of the goal nfg_1 (i.e. the satisfaction degree required for the latency of the voice and video conferencing). Each entry (row, column) of the table represents the choice of assemblies that DEER suggests for that specific optimization model. The choice is represented as a vector, where each element is the name of an assembly. Beside the vector of assemblies, the cost of the solution is also reported in each entry.

As expected, for the same value of the satisfaction degree required for the latency of the voice and video conferencing, the total cost of the application

COTS name	Cost c_i
C_1	9
C_2	11
C_3	1
C_4	1
C_5	8
C_6	6
C_7	4
C_8	3
C_9	2
C_{10}	1

Table 3: Mail server: costs of available COTS components.

Assembly name	Cost $\bar{c_j}$	COTS names	Func. Op. Goal names	Sat_{nfg_1}	Sat_{nfg_2}
A_1	0	C_1, C_3	fg_1	0	0
A_2	0	C_2	fg_1	0	0
A_3	3	C_1, C_4	fg_2	0	0
A_4	1	C_2, C_4	fg_2	0	0
A_5	0	C_2, C_5	fg_3, fg_4	1	1
A_6	0	C_1, C_6	fg_3, fg_4	0.9	0.9
A_7	0	C_2, C_6	fg_3, fg_4	0.9	0.9
A_8	1	C_1, C_7	fg_3, fg_4	1	0.5
A_9	1	C_2, C_7	fg_3, fg_4	1	0.5
A_{10}	0	C_1, C_8	fg_3, fg_4	0.5	0.3
A_{11}	0	C_1, C_9	fg_3, fg_4	0.3	0.3
A_{12}	0	C_2, C_9	fg_3, fg_4	0.3	0.3
A_{13}	0	C_2, C_{10}	fg_3, fg_4	0.3	0.2

Table 4: Mail server: parameters of the COTS assemblies.

decreases while decreasing the satisfaction degree required for the reliability of the anti-virus scanning. On the other hand, for the same value of the satisfaction degree required for the reliability of the anti-virus scanning the total cost almost always increases (except in two cases) while increasing the satisfaction degree required for the latency of the voice and video conferencing.

The results in Table 5 highlight, in general, that it is necessary to increase the cost of the system (i.e. to utilize more expensive assemblies) in order to better satisfy the system requirements. For example, once fixed the satisfaction degree of the goal nfg_1 to 0.2, DEER suggests different sets of assemblies while varying the satisfaction degree of the goal nfg_2. Nevertheless, in all cases DEER suggests to adopt the assemblies A_2 and A_4, both embedding the component C_2 that largely satisfies nfg_1. The solutions instead vary in terms of the assembly that is necessary to adopt to satisfy the goal nfg_2. While the satisfaction degree of the goal nfg_2 increases, in fact, it is necessary to adopt a more expensive COTS component that better satisfies nfg_2. In fact, if nfg_2 is equal to 0.2 then it is possible to adopt the assembly A_{13} that contains C_{10} that is the cheapest component. If nfg_2 is equal to 0.3 then it is necessary to adopt the assembly

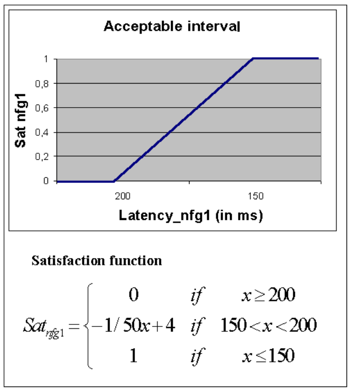

Figure 5: Acceptable interval and satisfaction function of the goal nfg_1.

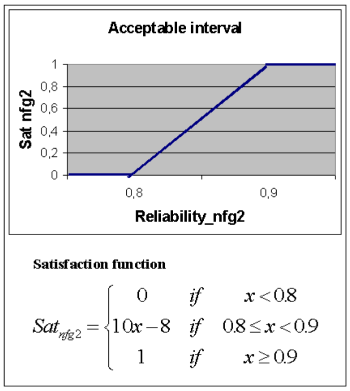

Figure 6: Acceptable interval and satisfaction function of the goal nfg_2.

A_{12} that is more expensive and contains C_9, and so on.

Sometimes, the cost of the system does not increase while raising the satisfaction degree of a requirement. For example, with the satisfaction degree of the goal nfg_2 to 0.5, DEER always suggests the assembly $[A_2, A_4, A_9]$ while varying the satisfaction degree of the goal nfg_1, because the original solution is good enough to satisfy the latter goal with minimum costs.

5.2 Second Experiment

The second experiment shows how the DEER framework reacts to a typical event in the software lifecycle, that is, the introduction of an additional requirement.

In this case, we intend to provide an automated support to tackle the problem of somehow modifying the selection of components in light of an additional re-

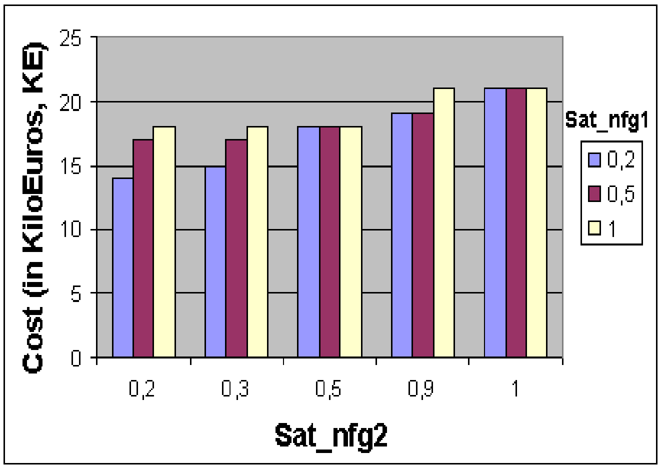

Figure 7: Model solutions for the first experiment.

	Sat_{nfg_1}= 0.2	Sat_{nfg_1}= 0.5	Sat_{nfg_1}= 1
Sat_{nfg_2}= 0.2	$[A_2, A_4, A_{13}]$ Cost = 14	$[A_1, A_3, A_{10}]$ Cost = 17	$[A_2, A_4, A_9]$ Cost = 18
Sat_{nfg_2}= 0.3	$[A_2, A_4, A_{12}]$ Cost = 15	$[A_1, A_3, A_{10}]$ Cost = 17	$[A_2, A_4, A_9]$ Cost = 18
Sat_{nfg_2}= 0.5	$[A_2, A_4, A_9]$ Cost = 18	$[A_2, A_4, A_9]$ Cost = 18	$[A_2, A_4, A_9]$ Cost = 18
Sat_{nfg_2}= 0.9	$[A_2, A_4, A_7]$ Cost = 19	$[A_2, A_4, A_7]$ Cost = 19	$[A_2, A_4, A_5]$ Cost = 21
Sat_{nfg_2}= 1	$[A_2, A_4, A_5]$ Cost = 21	$[A_2, A_4, A_5]$ Cost = 21	$[A_2, A_4, A_5]$ Cost = 21

Table 5: Solution vectors and solution values for the first experiment.

quirements that might not be satisfied (enough) by the current selection. Let us assume that we have to add the *functional operational goal* fg_5, that says: "Provide authentication of users". Table 6 shows the additional parameters necessary for this experiment, that are the satisfaction degrees, for all the assemblies, of the additional operational goal fg_5.

Figure 8 represents two histograms of results of this experiment. Each histogram is obtained while varying the satisfaction degree required for the relia-

Assembly name	Sat_{fg_5}
A_1	0
A_2	0
A_3	0
A_4	0
A_5	1
A_6	0
A_7	0
A_8	0
A_9	0
A_{10}	0
A_{11}	1
A_{12}	1
A_{13}	0

Table 6: Mail server: additional parameters of the assemblies for the second experiment.

bility of the anti-virus scanning and fixing the one of the latency of the voice and video conferencing to 0.2. In the first histogram we do not consider the goal fg_5, while in the second histogram we consider it.

As expected, for a given histogram, the cost of the system increases while increasing the satisfaction degree required for the reliability of the anti-virus scanning. On the other hand, by fixing a value on the x-axis and observing the values on the histograms, the cost of the system almost always increases if the *functional operational goal* fg_5 is considered.

Table 7 illustrates the results of this example and it is organized as follows: each row represents a value of the satisfaction degree of the goal nfg_2 (i.e. the reliability of the anti-virus scanning); the first column represents the case of the satisfaction degree of the goal nfg_1 (i.e. the latency of the voice and video conferencing) set to 0.2 and the set of the *operational goals* without fg_5, whereas the second column represents the same case with the functional goal fg_5. Hence, in each entry (row, column) we represent the choice of assemblies that DEER suggests for each specific case (i.e. optimization model). The choice is represented as a vector, where each element is the name of an assembly. Beside the vector of assemblies, the cost of the solution is also reported in each entry.

Looking at the results of Table 7, we can make the following considerations. Sometimes, it is necessary to increase the budget allocated for the system construction in order to meet an additional requirement. For example, once fixed the satisfaction degree of the goals nfg_1 and nfg_2 to 0.2, DEER suggests the assemblies $[A_2, A_4, A_{13}]$ if fg_5 is not considered, and the assemblies $[A_2, A_4, A_{12}]$ in the other case. In fact, although the former is a cheaper solution, the assembly A_{13} does not satisfy the *functional operational goal* fg_5.

In some other cases, instead, DEER reveals that it is not necessary to increase the cost of the system to meet an additional requirement. For example, if the

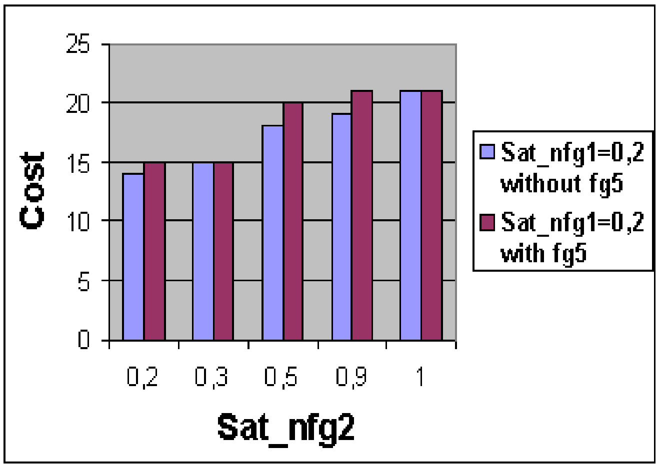

Figure 8: Model solutions for the second experiment.

	$Sat_{nfg_1} = 0.2$ without fg_5	$Sat_{nfg_1} = 0.2$ with fg_5
$Sat_{nfg_2} = 0.2$	$[A_2, A_4, A_{13}]$ Cost = 14	$[A_2, A_4, A_{12}]$ Cost = 15
$Sat_{nfg_2} = 0.3$	$[A_2, A_4, A_{12}]$ Cost = 15	$[A_2, A_4, A_{12}]$ Cost = 15
$Sat_{nfg_2} = 0.5$	$[A_2, A_4, A_9]$ Cost = 18	$[A_2, A_4, A_9, A_{12}]$ Cost = 20
$Sat_{nfg_2} = 0.9$	$[A_2, A_4, A_7]$ Cost = 19	$[A_2, A_4, A_5]$ Cost = 21
$Sat_{nfg_2} = 1$	$[A_2, A_4, A_5]$ Cost = 21	$[A_2, A_4, A_5]$ Cost = 21

Table 7: Solution vectors and solution values for the second experiment.

satisfaction degree of the goal nfg_2 is equal to 0.3, DEER suggests the same assembly with and without the *functional operational goal* fg_5.

Although the above considerations seem to be intuitive, note that they could not be quantified without the support of an optimization model like ours.

6 Related Work

Several interesting approaches have been introduced for supporting the activity of selection of COTS components (see, for example,[Ncube and Maiden 1999], [Chung et al. 2004] and [Kotonya and Hutchinson 2004]).

A quite extensive list of these approaches can be found in [Ruhe 2003], [Mohamed et al. 2007/b] and [Navarrete et al. 2005].
As remarked in [Andrews et al. 2005], some approaches are based on the architecture of the system, whereas other ones support the selection of COTS given a set of requirements, so such approaches can be used also in the requirements phase, in which the component selection activity is related to the definition of the (functional and non-functional) requirements (see Section 1).

However, all these approaches basically provide guidelines to select the components. Only a few of these approaches, some of which are detailed below, support automation for this task.

In [Grau et al. 2004] the DesCOTS (Description, evaluation and selection of COTS components) system is presented. DesCOTS is composed by four tools that collaborate with each other. It is designed for supporting only quality requirements, and the component selection criterion is based on quality models.

In [Krystkowiak et al. 2003] the OPAL tool is presented, which supports the definition of the requirements and the selection of the COTS components.

Both DesCOTS and OPAL do not consider the possibility that a requirement can be satisfied by a COTS assembly, whereas we introduce this possibility here.

The DEER framework is based on an optimization model. The optimization techniques resolve some drawbacks of the Analytic Hierarchy Process (AHP) and the Weighted Scoring Method (WSM) that are typically used by the component selection approaches. In fact, "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."[Neubauer and Stummer 2007]

In [Neubauer and Stummer 2007] it is presented an approach that can be embedded in a process of COTS selection. They suggest to solve an optimization model that, through a multi-objective optimization [Censor 1977], provides a set of COTS solutions. This so-called Pareto solution of the model is the one that maximizes a set of objectives (e.g. functionality, usability) under some constraints, such as on the resources available (e.g. maintenance cost). Besides, they suggest how to analyze and explore the space of solutions in order to find the preferred solution. The approach does not consider the possibility that a requirement can be satisfied by a COTS assembly, whereas we introduce this possibility here.

Few other experiences have been based on optimization problems to automatize the development of a component-based system. For example, in our previous work [Cortellessa et al. 2006] an optimization model is generated and solved within CODER (Cost Optimization under DElivery and Reliability constraints), a framework that supports "build-or-buy" decisions in selecting components. Getting as input value the software architecture and the possible scenarios of the system, CODER suggests the best selection for each component of the system, either one of the available COTS products or an in-house developed one. The selection is based on minimizing the costs of construction of the system under constraints on delivery time and reliability of the whole system. In addition, for each in-house developed component CODER suggests the amount of testing to perform in order to achieve the required level of reliability. CODER supports the selection of (COTS and in-house) component in the design phase, whereas DEER for the requirements phase. While CODER suggests the best assembly of components (i.e. the best software architecture), DEER selects a set of COTS (or assembly of more COTS). Since in the requirements phase the architecture of the system has not yet been built, DEER does not suggest how to make the best architectural decisions, whereas it helps to check if the requirements that do not depend only on the architecture of the system [IBM] are satisfied by a COTS component or an assembly of more COTS components.

Based on the dynamic monitoring of a system, in [Mikic-Rakic et al. 2004] an optimization model has been suggested for the best allocation of the components on the hardware hosts while maximizing the availability of the system under some constraints, such as the allocation of two components on the same host.

In [Grunske 2006] an optimization model is formulated to allow the best architectural decisions, while maximizing the satisfaction of the quality attributes of the system using multi-objective optimization [Censor 1977] under some constraints, such as budget limitations.

In [Mohamed et al. 2007/a] an approach based on an optimization model supports the resolution of mismatches between system requirements and COTS components during and after a COTS selection process. The model solution provides the best set of actions needed to resolve a set of mismatches.

The construction of COTS components and assemblies is outside the scope of this paper. However the approach presented in [Sjachyn and Beus-Dukic 2006] could be adopted to identify and classify the components available on the market. Besides, users need to determine aspects of components, such as functionality, limitations, pre- and post-conditions in order to decide which component is appropriate [Andrews et al. 2002]. Voas [Voas 1999] suggested an approach involving black-box testing to determine quality attribute, and interface perturbation analysis (IPA) and operational system testing to determine the impact of using a certain component in a system. Sometimes the COTS components do not provide a rich documentation that allows to understand their features, especially in relation to domain requirements. In fact, whereas some requirements (e.g., specific type of hardware, timing and performance constraints, security) can be found in product descriptions, the majority is not readily offered by vendors and has to be specifically requested (e.g., compliance to a domain standard), researched (e.g., popularity of a component in a specific domain), or tested (e.g., interoperability with other COTS) [Beus-Dukic 2000]. In literature there are many approaches that provide the outlines for defining the specification of a component, but they do not sufficiently provide the information needed for the selection and the reuse of components. In [Geisterfer and Ghosh 2006] this problem is faced by analyzing some possible approaches for component specification while taking into account the needs of the process of selection and reuse of components. In addition, they provide some recommendations to address these issues. In order to understand the features of a COTS component, it can be also adopted a comprehension process (e.g. [Andrews et al. 2005], [Ghosh et al. 2005]). A comprehension process of COTS components provides the outlines for developing the comprehension model of a component. While for the developers of the system it is difficult to understand the behavior of the components, at the same time for the end-user it is difficult to express the requirements. For instance, the non-functional requirements imposed on COTS component by end-user organizations are not easy to elicit and quantify [Beus-Dukic 2000].

The COTS components and the assemblies of COTS available on the market typically are developed for different domains of application. The knowledge of the domain of the application (e.g. operating system required) could be used [Sjachyn and Beus-Dukic 2006] to define filters useful for optimizing their research.

Besides, constraints on the resources available could be very helpful, for example, they are used in the approach presented in [Neubauer and Stummer 2007]. To reduce the search space of the model, constraints claiming the compatibility between components could be also formulated. In fact, the "compatibility check is useful to reduce the set of candidate components chosen to build a system, working as a selection criterion, and eliminating incompatible components" [Sillitti 2003].

The following major aspects characterize the novelty of our approach with respect to the existing literature:

– DEER could be adopted to automatize whatever approach supporting the activity of selection of COTS components (e.g. [Ncube and Maiden 1999], [Chung et al. 2004], [Kotonya and Hutchinson 2004]). In particular, it could support the fifth step of a process of COST selection (see Section 2.1).

– DEER is not tied to any particular component-based development process.

– DEER supports the study of the cost of construction of the system while adding, replacing and deleting requirements. The solution of the framework determines the COTS components to select in order to minimize the cost of construction of the system while assuring the satisfaction of the functional and non-functional requirements by single COTS or assemblies of COTS components.

– DEER allows to analyze how the cost of construction of the system varies while varying the targeted satisfaction degree for each requirement. In fact, by solving the optimization model with respect to different values of the satisfaction degree of the non-functional requirements, it is possible to draw curves that describe this trend (see Section 5.1).

– Sometimes the exact satisfaction degree value with which a COTS component or an assembly of COTS meet a non-functional requirement is not available, whereas it is possible to get a range over which the satisfaction degree value may lie. For example, it may be more feasible to predict the range over which component reliability may lie for a given cost based on prior experience rather than predicting the exact reliability value [Gokhale 2007]. If ranges over which the satisfaction degree with which the COTS component or the assemblies of COTS meet the requirements are available, DEER helps to address such a problem. In fact, by solving the optimization model with respect to different values with which the COTS component or the assemblies of COTS meet the requirements, it is possible to analyze the cost of construction of the system.

– The output of DEER can be an input of the design phase. In fact, the designers of the software architecture, basing on the set of assemblies provided by the framework, can start designing the non-COTS components of the system (if any) and, mostly, the necessary gluecode to adapt and assemble them.

7 Conclusions and Future Directions

We have presented DEER, a framework that supports the selection of COTS components in the requirements phase of a component-based development process. The solution of the framework determines the COTS component to select in order to minimize the cost of construction of the system while assuring the satisfaction of the requirements, which does not depend only on the architecture of the system [IBM]. A requirement can be satisfied by a single COTS component or an assembly of several COTS components. We have applied DEER to an example. We have conduced two experiments. The first one showed how accurately DEER provides support for automated analysis of the cost of construction of the system while varying the satisfaction degree targeted for requirements. The second one showed how DEER allows to analyze the sensitivity of the cost of the system upon introducing an additional requirement.

Experiments show that the use of a tool is very helpful to make decisions during the activity of component selection in the requirements phase. By resolving the optimization model for different parameters it is possible to analyze the cost of the system with respect to different scenarios (e.g. different values for the satisfaction degree targeted for requirements, or introduction of an additional requirement).

We are investigating several future directions, for example, we intend to enhance DEER in order to support the resolution of conflicts between requirements by assigning priorities to requirements and introducing the risk that considers the degree of uncertainty associated with the selection of a certain assembly of COTS components (like in [Mahmood and Lai 2006]). Besides, we intend to deal with the dependencies that may exist between the requirements. In fact, a functional requirement could be tied to a set of non-functional requirements. For example, a mail server (see Section 5) could require an anti-virus with a certain level of reliability and availability. So, DEER could provide an assembly of COTS that provide an anti-virus with the required characteristics. Therefore, the DEER framework could be constrained to suggest the assembly of COTS that satisfies both the functional and the related non-functional requirements.

As we remarked in Section 1, DEER could be used to decide whether in-house developed software has to be maintained or replaced by some COTS components. We plan to enhance DEER in order to better support the possibility to develop a component in-house by taking into account their typical characteristics. For example, in DEER we assume the cost of a COTS component as an input value, whereas for in-house components we intend to introduce a development cost model (e.g. COCOMO [Boehm 1981]). Also, the cost of a COTS component can be refined into direct costs, maintenance costs and return of investment, which can in total decrease the initial cost.

Besides, we intend to refine the cost of an assembly of COTS (i.e. adaptation and integration cost) by expressing it as a function of the satisfaction degree with which an assembly satisfies a goal. The rationale behind this direction is that an adaptation cost very likely depends on how well the acquired COTS satisfies the requirement(s) for which it has been selected. For example, we want to study the applicability of the optimization model by introducing the relationship that we have provided for the adaptation cost (see Section 3.2.1). Since this relationship makes the model integer non-linear it would be interesting to study its scalability while adding new COTS components.

With regard to the parameters estimation we also intend to study techniques for the estimation of the satisfaction factor of a COTS components (e.g. how to estimate the reliability of a software component). Following this issue we also intend to study the sensitivity of the model result with respect to the accuracy of parameters estimation.

In order to address computation effort problems that may raise, we intend to investigate the use of metaheuristics techniques [Blum and Roli 2003] for solving the model (e.g. the tabu search algorithm). Furthermore, we intend to enhance DEER by introducing a multi-objective optimization [Censor 1977], as the approach presented in [Neubauer and Stummer 2007] suggests, to provide the configuration of components that minimizes, for example, both the cost of construction of the system and some non-functional requirements of the system.

We also intend to enhance DEER for allowing to specify ranges for model parameters and to experiment the model and the DEER tool on larger size systems.

Finally, as a long term goal, we work on the automatization of the process of selection of (COTS and in-house) components in all the phases of a component-based development process, as we have discussed in Section 2.1. In particular, we are designing an integrated tool, based on several optimization models (such as the ones in CODER, DEER and others), that may assist software designers during the whole software component lifecycle.

Acknowledgements

Authors would like to thank the reviewers for their excellent comments that have helped to improve the quality of the paper.
This work was partially supported by the Swedish Foundation for Strategic Research via the strategic research center PROGRESS, and partially supported by the European Community (6th Framework Programme) via the PLASTIC project.

References

[Alves et al. 2005] Alves, C., Franch, X., Carvallo, J.P., Finkelstein, A.: "Using Goals and Quality Models to Support the Matching Analysis During COTS Selection", *Proc. of ICCBSS 2005*, LNCS 3412, 146-156, 2005.

[Andrews et al. 2002] Andrews, A., Ghosh, S., Choi, E.: "A Model for Understanding Software Components", *Proc. of 18th IEEE International Conference on Software Maintenance (ICSM'02)*, 2002.

[Andrews et al. 2005] Andrews, A., Stefik, A., Picone, N., Ghosh, S.: "A COTS Component Comprehension Process", *Proc. of the 13th International Workshop on Program Comprehension (IWPC'05)*,2005.

[Beus-Dukic 2000] Beus-Dukic, L.: "Non-Functional Requirements for COTS Software Components",*Proc. of the 2nd Workshop on COTS Software*, 4-5 June 2000, Limerick, Ireland.

[Blum and Roli 2003] Blum C., Roli A.: "Metaheuristics in Combinator Optimization: Overview and Conceptual Comparison", *ACM Computing Surveys*,vol.35, n.3, 2003.

[Boehm 1981] Boehm, B.: "Software Engineering Economics", Prentice-Hall, 1981.

[Censor 1977] Censor, Y.: "Pareto Optimality in Multiobjective Problems", Appl. Math. Optimiz., vol. 4, 41-59, 1977.

[Chung et al. 2004] Chung, L.: Cooper, K., Courtney, S. "COTS-Aware Requirements Engineering: The CARE Process", *Proc. of RECOTS 2004*, September 2004, Kyoto, Japan.

[Cortellessa et al. 2006] Cortellessa, V., Marinelli, F., Potena, P.: "Automated Selection of Software Components Based on Cost/Reliability Tradeoff", *Proc. of EWSA06*, LNCS 4344, 66-81, 2006.

[Cortellessa et al. 2007] Cortellessa, V., Crnkovic, I., Marinelli, F., Potena, P.: "Driving the Selection of COTS Components on the Basis of System Requirements", (short paper) *Proc. of 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007)*, Atlanta, Georgia, USA, November 5-9, 2007.

[Crnkovic et al. 2005] Crnkovic, I., Chaudron, M., Larsson, S.:"Component-based Development Process and Component Lifecycle", *Proc. of 27th International Conference Information Technology Interfaces (ITI), IEEE*, Cavtat, Croatia, June, 2005.

[Crnkovic and Larsson 2004] Crnkovic, I., Larsson, M.: "Classification of quality attributes for predictability in component-based systems"*Proc. of Workshop on Architecting Dependable Systems*, IEEE, June, 2004.

[Dashofy et al. 1999] Dashofy, E. M., Medvidovic, N., Taylor, R. N.: "Using Off-the-Shelf Middleware to Implement Connectors in Distributed Software Architectures" *Proc. of ICSE 1999*, Los Angeles, CA, May 16-22, 1999.

[Di Giacomo 2005] Di Giacomo, P.: "COTS and Open Source Software Components: Are They Really Different on the Battlefield?" *Proc. of ICCBSS 2005*, LNCS 3412, 301-310, 2005.

[Geisterfer and Ghosh 2006] Geisterfer, CJ., M., Ghosh, S.: "Software Component Specification: A Study in Perspective of Component Selection and Reuse, *Proc. of the Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS 2006)*, 2006.

[Ghosh et al. 2005] Ghosh, S., Kelly, J., L., Shankar, R., P.: "Enabling the Selection of COTS Components"*Proc. of ICCBSS 2005*, LNCS 3412, 122-131, 2005.

[Gokhale 2007] Gokhale, S. S.:"Architecture-Based Software Reliability Analysis: Overview and Limitations"*IEEE Transactions on dependable and secure computing*, vol. 4, n. 1, January/March 2007.

[Gokhale 2004] Gokhale, S.S., Wong, W.E, Horgan, J.R., Trivedi, K.S.: "An analytical approach to architecture-based software performance and reliability prediction", *Performance Evaluation*, vol. 58 (2004), 391-412.

[Goseva-Popstojanova and Kamavaram 2002] Goseva-Popstojanova, K., Kamavaram, S.: "Uncertainty Analysis of Software Reliability Based on Method of Moments" *FastAbstract ISSRE 2002*.

[Grau et al. 2004] Grau, G., Carvallo, J. P., Franch, X., Carme, Q.: "DesCOTS: A Software System for Selecting COTS Components"*Proc. of EUROMICRO 2004)*, 2004.

[Grunske 2006] Grunske, L.: "Identifying "Good" Architectural Design Alternatives with Multi-Objective Optimization Strategies" *Proc. of ICSE06*, May 20-28, 2006, Shanghai, China.

[Kazman 2001] Kazman, R., Asundi, J., Klein, M.: "Quantifying the Costs and Benefits of Architectural Decisions" *Proc. of 23rd International Conference on Software Engineering (ICSE'01)*, 2001.

[Kotonya and Hutchinson 2004] Kotonya, G., Hutchinson, J.: "Viewpoints for Specifying Component-Based Systems" *Proc. of CBSE 2004*, LNCS 3054, Springer 2004, Edinburgh, UK, May 24-25, 2004.

[Krystkowiak et al. 2003] Krystkowiak, M., Bucciarelli, B., Dubois, E.: "COTS Selection for SMEs: a report on a case study and on a supporting tool" *Proc. of RECOTS03*, 2003.

[Lamsweerde 2001] Lamsweerde, A.: "Goal-Oriented Requirements Engineering: A Guided Tour", *Invited mini-tutorial paper, appared in Proc. of RE 2001*, 249-263, Toronto, August 2001.

[Liang and Luqi 2003] Liang, S. X., Luqi, J. P.: "Perspective-based Architectural Approach for Dependable Systems" *Proc. of ICSE 2003, Workshop on Software Architectures for Dependable Systems*, Portland, Oregon - USA, 3 May, 2003.

[Mahmood and Lai 2006] Mahmood, S., Lai, R.: "Analyzing Component Based System Specification" *Proc. of AWRE 2006*, Adelaide, Australia, 2006.

[Maiden and Ncube 1998] Maiden, N. A., Ncube, C.: "Acquiring COTS Software Selection Requirements", *IEEE Software* Volume 15, Issue 2, Pages: 46 56, March 1998.

[Mikic-Rakic et al. 2004] Mikic-Rakic, M., Malek, S., Beckman, N., Medvidovic, N.: "A Tailorable Environment for Assessing the Quality of Deployment Architectures in Highly Distributed Settings" *Proc. of 2nd International Working Conference on Component Deployment (CD 2004)*, Edinburgh, UK, May 20-21, 2004.

[Mohamed et al. 2007/a] Mohamed, A., Ruhe, G., Eberlein, A.:"MiHOS: an approach to support handling the mismatches between system requirements and COTS products" *Requirements Engineering, Computer Science*, 2007.

[Mohamed et al. 2007/b] Mohamed, A., Ruhe, G., Eberlein, A.: "COTS Selection: Past, Present, and Future" *Proc. of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)* , 2007.

[Navarrete et al. 2005] Navarrete, F., Botella, P., Franch, X.: "How Agile COTS Selection Methods are (and can be)?" *Proc. of 31st EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA05)*, 2005.

[Ncube and Maiden 1999] Ncube, C., Maiden, N. A. M.: "PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm" *Proc. of International Workshop on Component-Based Software Engineering*, 1999.

[Neubauer and Stummer 2007] Neubauer, T., Stummer, C.: "Interactive Decision Support for Multiobjective COTS Selection" *Proc. of 40th Hawaii International Conference on System Sciences*, 2007.

[Nuseibeh et al. 2001] Nuseibeh, B. A., Easterbrook, S. M., Russo, A.: "Making Inconsistency Respectable in Software Development" *Journal of Systems and Software*, 58(2), 171-180, 2001.

[Reeves 1993] Reeves, C.R.: *Modern heuristic techniques for combinatorial problems* John Wiley & Sons, Inc. New York, NY, USA.

[Ruhe 2003] Ruhe, G.: "Intelligent Support for Selection of COTS Products" *Web, Web-Services, and Database Systems 2002*, LNCS 2593, 34-45, 2003.

[Sillitti 2003] Sillitti, A., Granatella, G., Predonzani, P., Vernazza, T., Succi, G.: "Dealing with Software Components Compatibility" *Proc. of the 7th World Multi-Conference on Systemics, Cybernetics and Informatics*, USA, 2003.

[Sjachyn and Beus-Dukic 2006] Sjachyn, M., Beus-Dukic L.:"Semantic Component Selection - SemaCS" *Proc. of the Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems (ICCBSS 2006)*, 2006.

[Trivedi 2001] Trivedi, K.: "Probability and Statistics with Reliability, Queuing, and Computer Science Applications", J. Wiley and S., 2001.

[Vidger and Dean 2000] Vidger, M.R., Dean, J.: "Maintaining a COTS-Based System" *Proc. of The NATO Information Systems Technology Panel Symposium on Commercial Off-the-shelf Products in Defence Applications*, Brussels, Belgium, April 3-5, 2000.

[Voas 1998] Voas, J.: "Maintaining Component-Based Systems" *IEEE Software*, July/August 1998.

[Voas 1999] Voas, J.: "Certifying Software for High-Assurance Environments". *IEEE Software*, 16(4):48-54, 1999.

[Wallnau 2002] Wallnau, K.: "Dispelling the Myth of Component Evaluation" *in Ivica Crnkovic and Magnus Larsson (editors), "Building Reliable Component-Based Software Systems"*, Artech House Publisher, 2002.

[Yacoub et al. 1999] Yacoub, S., Cukic, B., Ammar, H.: "Scenario-Based Reliability Analysis of Component-Based Software", *Proc. of the 10th International Symposium on Software Reliability Engineering (ISSRE'99)"*, 1999,22-31.

[Yakimovich et al. 1999] Yakimovich, D., Bieman, J. M., Basili, V. R.: "Software architecture classification for estimating the cost of COTS integration.", *Proc. of ICSE 1999*, pages 296-302, June 15-21, 1999.

[IBM] Home Page of the article "Capturing Architectural Requirements", available at http://www-128.ibm.com/developerworks/rational/library/4706.html.

[LINGO] Lindo's Home Page, available at http://www.lindo.com.