

# Matching, Ranking, and Selecting Components: A COTS-Aware Requirements Engineering and Software Architecting Approach

Lawrence Chung  
Department of Computer Science  
The University of Texas at Dallas  
[chung@utdallas.edu](mailto:chung@utdallas.edu)

Kendra Cooper  
Department of Computer Science  
The University of Texas at Dallas  
[kcooper@utdallas.edu](mailto:kcooper@utdallas.edu)

## Abstract

*At the heart of a well-disciplined, systematic methodology that explicitly supports the use of commercial off-the-shelf (COTS) components is a clearly defined process for effectively using components that meet the needs of the system under development. In this paper, we present the CARE/SA approach that supports the iterative matching, ranking, and selection of COTS components. The components are represented as an aggregate of its functional and non-functional requirements and architecture.*

## 1. Introduction

The use of commercial off-the-shelf (COTS) components is perceived to significantly shorten development time and cost, while improving quality, in developing large, complex software systems. Ideally, COTS components offer pre-packaged solutions which presumably have already gone through the various time-consuming and costly phases of requirements specification, design, coding, testing, and have been hardened over time. However, the effective use of COTS components requires a well-disciplined systematic methodology that facilitates the exploitation of the benefits of COTS components while guarding against their pitfalls.

This paper presents the COTS-Aware Requirements Engineering and Software Architecting (CARE/SA) Framework, which supports the iterative matching, ranking, and selection of COTS components. COTS components are represented as an aggregate of their functional and non-functional requirements and architecture.

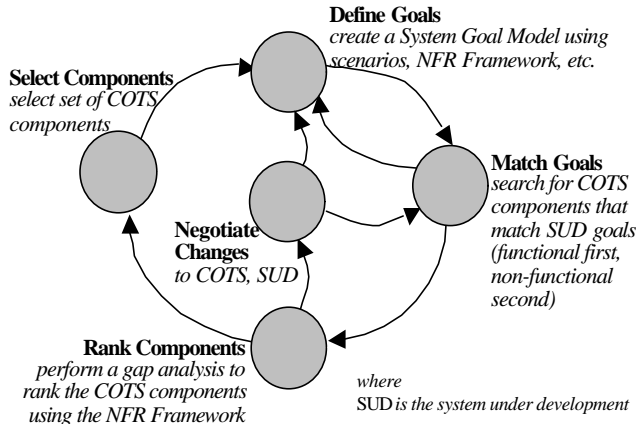
CARE/SA can be viewed as an extension to current methodologies with a systematic approach to represent, match, rank, and select COTS components. The Rational Unified Process (RUP) is an object oriented software engineering technique [10] that uses the Unified Modeling Language (UML). In UML a component, a physical and replaceable part of the system that provides a set of interfaces and typically represents the physical packaging of classes, interfaces, and collaborations [11], is used to represent a COTS component. The Procurement Oriented Requirements Engineering (PORE) technique supports the evaluation and selection of COTS components [12]. PORE's process model identifies four goals in COTS selection: acquiring information from the stakeholders, analyzing the information to determine if it is complete and correct, making the decision about product requirement compliance, and selecting one or

more candidate COTS components. The Model Based Architecting and Software Engineering (MBASE) approach considers four types of models: success, process, product and property [3] and is consistent for use with COTS components [2]. Leveraging the RUP, the EPIC (Evolutionary Process for Integrating COTS Based Systems) simultaneously defines and makes trade-offs among four *spheres of influence*, in building, fielding, and supporting component-based business solutions [1]: the stakeholders' needs and their business processes, the components currently available in the marketplace, the architecture and design of the system, and programmatic and risks.

Section 2 presents how COTS components are modeled as well as the process to match, rank and select components, along with some experiences in using the approach. Conclusions are in Section 3.

## 2. Representing and Evaluating Components

At the heart of an effective COTS-aware methodology are techniques to assist the requirements engineer (*RE*) with the challenging tasks of defining goals, matching, ranking, and selecting potential COTS components, and negotiating changes to the components and/or the system under development, hereafter *SUD* (refer to Figure 1). To be re-usable, components are likely to have a rich set of functional and non-functional capabilities that need to be represented and reasoned about. Given an initial set of goals for a system under development, it is unlikely that there is a simple, one to one mapping from the goals of the *SUD* and the goals (implemented as capabilities) of currently existing COTS components. The *RE* needs to search for matching components, rank them in terms of how well they meet the current *SUD* goals, and select components. In addition, the *RE* needs to iteratively bridge the gap between the currently available components in the



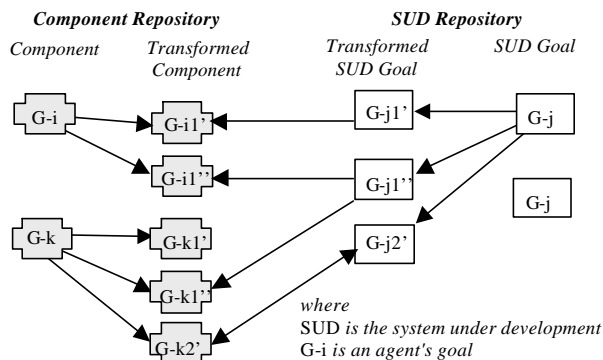
**Figure 1. Overview of the CARE/SA Process.**

repository and the stakeholders' goals for the SUD.

The decisions involved to either negotiate unattained SUD goals or to modify a COTS component are important issues to be considered during the development of COTS-based systems (refer to Figure 2). For example, if a COTS component that provides the necessary functional capabilities is described as high performance and high cost, then the development house may negotiate with a vendor to provide a modified component with moderate performance and moderate cost. Negotiations with the stakeholders to modify, or rewrite, a goal for the SUD may also be possible. If a COTS component cannot be identified at this point, then the RE documents these results. Later in the development process, when the goals are refined into software requirements, the RE can search for suitable COTS components again.

Throughout the CARE/SA process, the decisions and the rationale for making them are documented. For example, when the RE matches, ranks, and selects the components, the reasons, or rationale, for selecting the components as candidates are maintained. When the requirements need to be modified, the RE uses the rationale history as an aid.

The following sections present the CARE/SA approach to modeling COTS components and the activities to match, rank, and select components.



**Figure 2. Negotiating Changes** (Bridging the Gaps Between SUD Requirements and the Capabilities of the COTS Components).

## 2.1 Modeling COTS Components in CARE/SA

The CARE/SA approach represents a COTS component as an aggregation of its requirements (functional and non-functional) and architecture. The representation is used to iteratively match, rank, and select COTS components as the definition of the SUD's requirements and architecture proceeds. The functional and non-functional specifications include the goals, or high-level descriptions of a component's capabilities, such as information found on marketing brochures for products. The description provides enough information for a reader to determine if the product appears to have some potential for use and warrants further investigation. In addition, detailed descriptions of the functional and non-functional characteristics of a component, such as information found on the data sheets for a product are represented. The attributes stored and maintained for a product specification include: type; list of keywords and weights (used for keyword and case based searching); functional overview; domain; vendor; standards compliance; interface type; performance; security; related subcomponents; and lessons learned (e.g., interactions among components - incompatibilities, emergent behavior, etc.). The architecture of the components [14] describes the components, connectors, constraints, patterns, styles, rationale, etc.

## 2.2 Matching, Ranking and Selecting Components within the CARE/SA Process

The CARE/SA process model defines when and how to define the agents, goals, requirements, and architecture, all with respect to using COTS components. Preliminary process definitions have been proposed involving agents [4], goals [6], and software architecture [5].

One of the first steps in CARE/SA is to elicit a set of initial goals. Since goals may be very abstract, the RE may need to decompose them. For example, a softgoal "the system should be scalable" leads the RE to ask the question "scalable in what way?" When interviewed, one stakeholder may intend this goal to mean the system should be scalable to support a higher number of concurrent users. Another may intend this goal to mean supporting additional data in the database. In addition to decomposing the goals, the RE also needs to document the relationships among the goals. A goal-oriented framework is used to characterize the relationships between goals. One such framework is the NFR Framework [7] which provides a set of rankings of the relationships between two (soft)goals: very positive (++), positive (+), negative (-), and very negative (--).

In the NFR Framework, goals (softgoals and hardgoals) are organized into a (soft)goal interdependency graph (SIG), much like the AND/OR trees used in problem-solving [13]. Unlike traditional problem-solving

and planning frameworks, however, goals representing non-functional requirements (NFRs hereafter) can rarely be said to be “satisfied” in a clearcut sense. Instead, different design decisions contribute positively or negatively towards a particular goal. Accordingly, the Framework uses the notion of goal *satisficing* [15] to suggest that generated software is expected to satisfy within acceptable limits, rather than absolutely, NFRs.

The SUD goals (and subgoals) are represented as:

$$SUD-R = SUD-NFR \dot{\cup} SUD-FR$$

where *SUD-R*, *SUD-NFR*, and *SUD-FR* respectively denote requirements, NFRs, and functional requirements for the SUD.

Furthermore,

$$SUD-NFR = \{SUD-NFR-1, SUD-NFR-2, \dots, SUD-NFR-l\}$$

$$SUD-FR = \{SUD-FR-1, SUD-FR-2, \dots, SUD-FR-m\}$$

where each *SUD-NFR-l* is a non-functional requirement for a SUD, and each *SUD-FR-m* is a functional requirement for a SUD (note - there is only one SUD).

Once a collection of goals is defined, the RE analyzes them to identify errors of commission (conflicting, incorrect, and redundant goals) and omission (missing goals). The RE corrects goals, removes redundant goals, adds missing goals, and negotiates conflicting goals. The stakeholders validate the goals to ensure the RE understands their needs and wants. The process to elicit, analyze, correct, and validate goals may be iterative.

When the agents (stakeholders) have no major issues with the goals defined, then the RE begins to match, rank, and select COTS components with the assistance of a Component Engineer (CE) and Software Architect (SA). A CE, responsible for building and maintaining the component repository, is part of the acquirer's organization. The need for this role and activity is based on the understanding that initial data available from a vendor may be incomplete and biased. A CE systematically evaluates a component (on behalf of the acquirer) using empirical studies; this improves the confidence in the results of the matching, ranking, and selecting process. As a project proceeds, more complete data about a component can be gathered as it becomes a more likely match. The SA, also part of the acquirer's organization, may provide additional information of the impact of components on the system architecture.

**Match Components.** Here, the RE begins the process by identifying functional hardgoals that may be candidates for implementing with one or more COTS components. For each candidate, the RE performs a search on the repository that returns functional goal descriptions of the components that match the search criteria. Currently, keyword based search is supported. The keywords used in the COTS component definitions are used to build a glossary of terms that are made available to the RE, who select keywords from this glossary. The RE evaluates the results of the preliminary search and determines which of the components (if any) may be a possible match to a

hardgoal. The RE performs a detailed search on the repository for the components that appear to satisfy the preliminary match; detailed descriptions of the functional and non-functional descriptions are returned. Matching occurs at two levels: (Level 1) requirements and (Level 2) architecture. Here, the focus is on Level 1 matching:

$$COTS-R = \dot{\cup}_i COTS-i-R,$$

where *COTS-R* denotes the set of all the COTS components' requirements and each *COTS-i-R* is the set of requirements for the *i*-th COTS component (e.g. for a specific text editor, etc.)

$$\text{Further, } COTS-i-R = COTS-i-NFR \dot{\cup} COTS-i-FR$$

$$COTS-i-NFR = \dot{\cup}_j COTS-i-NFR-j$$

$$COTS-i-FR = \dot{\cup}_k COTS-i-FR-k$$

each *COTS-i-NFR-j* is a non-functional requirement *j* for a COTS component *i* (e.g., response time performance, etc.) and *COTS-i-FR-k* is a functional requirement *k* for a COTS component *i* (e.g. select text, delete text, or highlight text, etc.).

**Rank Components.** The COTS components that appear to be possible matches are grouped based on the functional requirements into sets including:

$$COTS-i-FR-k-MAKE-SUD-FR-m = \{COTS-i-FR-k | COTS-i-FR-k-MAKE-SUD-FR-m\}$$

$$COTS-i-FR-k-HELP-SUD-FR-m = \{COTS-i-FR-k | COTS-i-FR-k-HELP-SUD-FR-m\}$$

where *MAKE* means an exact match or one with minor, insignificant mismatch, and *HELP* means close match with tolerable mismatch.

The matching should consider if COTS requirements are bigger/smaller in scope, or if they are similar overall, but with a different context or scope.

Subsequently, the NFRs (e.g., performance, authentication, etc.) provided by a COTS architectural component are considered, resulting in sets including:

$$\{ COTS-i-A-a | COTS-i-FR-k-MAKE-SUD-FR-m \dot{\cup} COTS-i-A-a-MAKE-COTS-i-NFR-j \}$$

$$\{ COTS-i-A-a | COTS-i-FR-k-MAKE-SUD-FR-m \dot{\cup} COTS-i-A-a-HELP-COTS-i-NFR-j \}$$

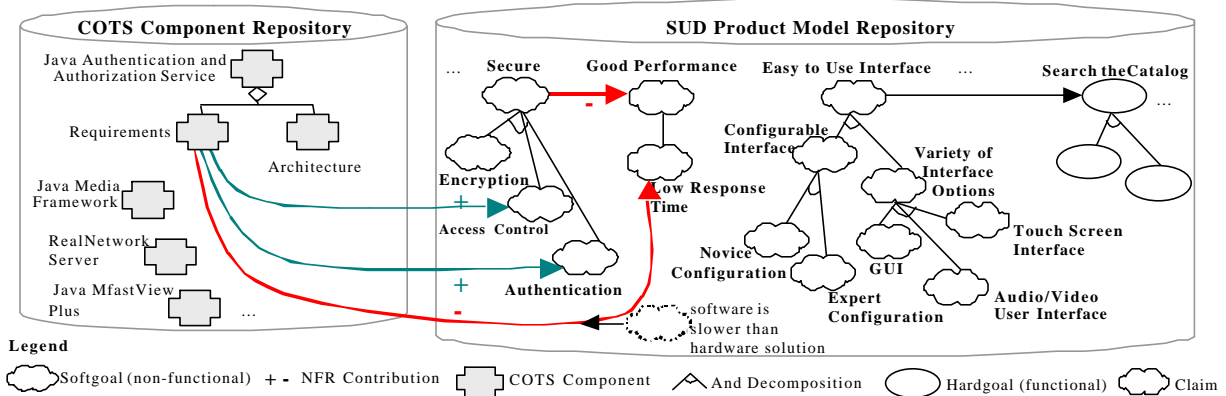
where *COTS-i-A* represents the (whole) architecture for a COTS component *i*, and *COTS-i-A-a* represents an architectural subcomponent (i.e., subsystem). The representation for *COTS-i-A-a* can be further refined as an aggregate of its capabilities, constraints, connections, styles, and patterns as needed.

The components are further grouped, this time based on the contributions between the COTS component NFRs and the SUD NFRs. The resulting groups include:

$$\{ COTS-i-A-a | COTS-i-FR-k-MAKE-SUD-FR-m \dot{\cup} COTS-i-A-a-MAKE-COTS-i-NFR-j \dot{\cup} COTS-i-NFR-j-MAKE-SUD-NFR-l \}$$

$$\{ COTS-i-A-a | COTS-i-FR-k-MAKE-SUD-FR-m \dot{\cup} COTS-i-A-a-HELP-COTS-i-NFR-j \dot{\cup} COTS-i-NFR-j-HELP-SUD-NFR-l \}$$

**Select Components.** The RE examines the results of the ranking and determines which of the components is a close or exact match to the goals. Due to the early stage of development, the RE is unlikely to have enough information to evaluate the component completely for its use in the SUD. For example, a component may support



**Figure 3. Preliminary Softgoal Interdependency Graph with Relationships to COTS Components** (This illustrates how part of the CARE/SA knowledge base is populated for a DLS; this corresponds to Level 1, Goal matching, in the process model).

MPEG 3 but not MPEG 4; the RE may not know at this point if the earlier standard is suitable.

**Experiences.** A Digital Library System (DLS) has been used to validate CARE/SA. Figure 3 represents how part of the CARE/SA knowledge base can be instantiated. On the left, a COTS repository is illustrated with a collection of components that are available in the marketplace. These components (Java media framework, RealNetwork Server, etc.) which play out audio and video stored in different standards (.midi, MPEG3, etc.) have been found by searching the Internet and specified using information from the vendors. This is quite variable, where the missing information about the components needs to be extracted by a CE. On the right, the functional (hardgoals) and non-functional (softgoals) of the SUD are represented in a SIG. Some of the softgoals for a DLS include security and being easy to use. Hardgoals include being able to search the catalog. Here, COTS components that may realize functional and non-functional goals of the SUD are matched, ranked, and selected, while the relationships are defined, and their contributions are evaluated.

### 3. Conclusions

Matching, ranking, and selecting COTS components in the CARE/SA Framework uses the functional and non-functional requirements and the architecture of the components. COTS components are represented as an aggregate of functional and non-functional requirements and architecture - each with its own set of attributes. The approach improves the confidence in the matching, ranking, and selecting process by systematically employing the NFR Framework to qualitatively evaluate the COTS components. Confidence is also improved as the CE, affiliated with the acquirer's organization, uses empirical studies to obtain additional component data. Currently, keyword based searching is supported to match SUD requirements with COTS requirements. This approach has been applied to a Digital Library System.

Future work includes investigating additional search techniques (case based [9], rule based, and trading [8])

and ranking techniques (multi-criteria decision making algorithms such as the analytic hierarchy process). A meta-model for CARE/SA and web-based tool support are also planned.

### References

- [1] Albert, C. and Brownsword, L., *Evolutionary Process for Integrating COTS-Based Systems (EPIC) Building, Fielding, and Supporting Commercial-off-the-Shelf (COTS) Based Solutions*, CMU/SEI-2002-TR-005, 2002.
- [2] Boehm, B. "Requirements that handle IKIWISI, COTS, and Rapid Change", *IEEE Computer*, 33(7), July 2000, pp. 99–102.
- [3] Boehm, B., Port, D., Abi-Antoun, M., and Egyed, A. *Guidelines for the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) deliverables for Model-Based Architecting and Software Engineering (MBASE)*, TR USC-CSE-98-519, USC-Center for Software Engineering.
- [4] Chung, L. and Cooper, K., "Defining Agents in a COTS-Aware Requirements Engineering Approach", *Proc., 7<sup>th</sup> Int. Australian Workshop on Requirements Eng.*, 2002.
- [5] Chung, L. and Cooper, K., "Defining an Architecture with a COTS-Aware Software Engineering Process", *Proc., Int. Council on Systems Eng. Symp.*, 2003, pp. 1219-1228.
- [6] Chung, L. and Cooper, K., "Defining Goals in a COTS-Aware Requirements Engineering Approach", *System Engineering journal*, 7(1), 2004, pp. 61-83.
- [7] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J., *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing, 2000.
- [8] L. Iribarne, J.M. Troya, and A. Vallecillo. "A Trading Service for COTS Components". *The Computer Journal*, 47(3), May 2004.
- [9] Kolodner, J. *Case-Based Reasoning*, Morgan Kaufmann Publishers, San Mateo, CA. 1993.
- [10] Kroll, P. and Kruchten, P., *The Rational Unified Process Made Easy*, Addison-Wesley, 2003.
- [11] Kruchten, P., "Modeling Component Systems with the Unified Modeling Language", *Int. Workshop on Component-Based Software Eng.*, 1998.
- [12] Ncube C. and Maiden N, "Guiding parallel requirements acquisition and COTS software selection", *Proc., IEEE Int. Symp. on Requirements Eng.*, 1999, pp. 133-140.
- [13] Nilsson, N.J., *Problem Solving Methods in Artificial Intelligence*, McGraw Hill, USA, 1971.
- [14] Shaw, M. and Garlan, D., *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.
- [15] Simon, H., *The Science of the Artificial*, Second edition, Cambridge, MA., The MIT Press, 1981