

COTS-Aware Requirements Engineering and Software Architecting¹

Lawrence Chung

Department of Computer Science
The University of Texas at Dallas
chung@utdallas.edu

Kendra Cooper

Department of Computer Science
The University of Texas at Dallas
kcooper@utdallas.edu

Abstract

At the heart of a well-disciplined, systematic methodology that explicitly supports the use of 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 which supports the iterative matching, ranking, and selection of COTS components, using a representation of COTS components as an aggregate of their functional and non-functional requirements and architecture. The approach is illustrated using a Digital Library System example.

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 that 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 technical, business, and legal pitfalls [2],[14],[22].

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, using a representation of COTS components as an aggregate of their requirements and architecture.

The CARE/SA Framework can be viewed as an extension to current methodologies with a systematic approach to match, rank, and select COTS components. The Rational Unified Process (RUP) is an object oriented software engineering technique [12] which is based on four phases (transition, construction, elaboration, inception), and uses the unified modeling language (UML). In UML, a COTS component is represented as 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 [13]. The Procurement Oriented Requirements Engineering (PORE) technique supports the evaluation and selection of COTS components [16]. 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 [4] and is consistent for use with COTS components [3]. The Evolutionary Process for Integrating COTS Based Systems (EPIC) framework has been presented to meet the challenges of building, fielding, and supporting component-based business solutions [1]. While leveraging the RUP, EPIC simultaneously defines and makes trade-offs among four *spheres of influence*: 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 and the CARE/SA process definition. An example illustrating the application of part of the process is presented in Section 3. Conclusions are in Section 4.

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

¹ This is an extended and improved version of [8]; this extension considers both functional and non-functional requirements as candidates for the matching, ranking, and selection process.

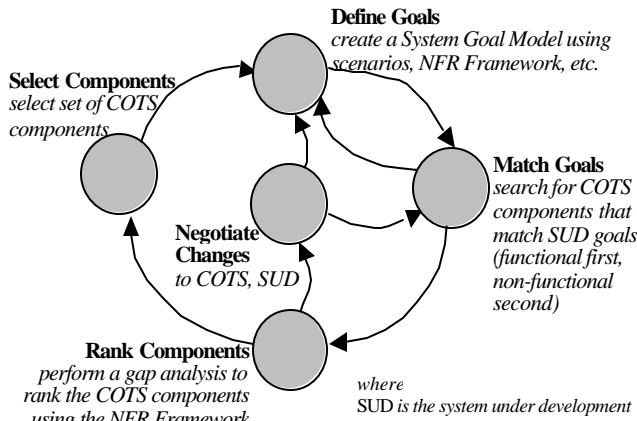


Figure 1. Overview of the CARE/SA Process.

(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 repository and the stakeholders' goals for the SUD.

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.

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 [19] describes the components, connectors, constraints, patterns, styles, rationale, etc.

2.2 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 [5], goals [7], and software architecture [6]. Here, we describe the five activities identified in Figure 1: Define Goals, Match Components, Rank Components, Select Components, and Negotiate Changes.

Define Goals. 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 scaleable" leads the RE to ask the question "scaleable in what way?" When interviewed, one stakeholder may intend this goal to mean the system should be scaleable 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 [9] 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 [17]. 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* [20] 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).

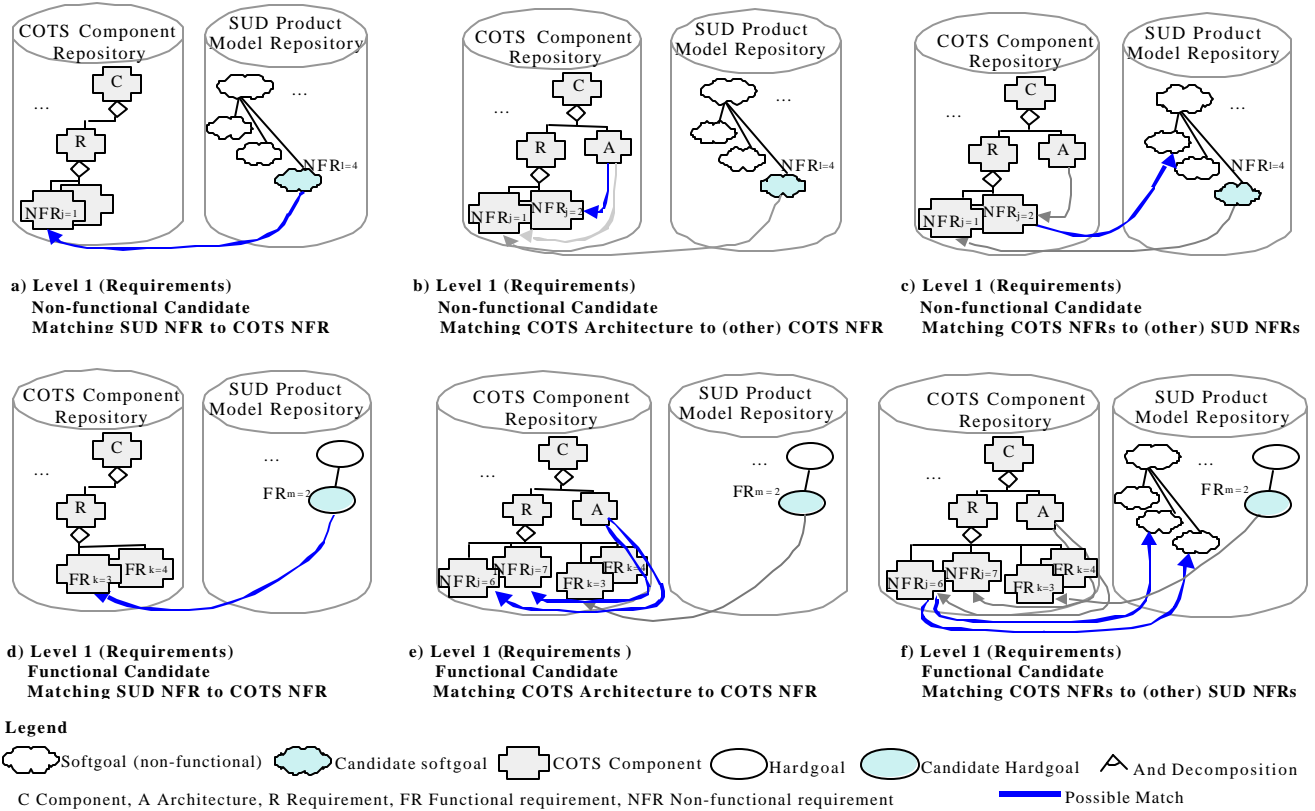


Figure 2. Matching, Ranking, and Selecting COTS Components. (There are two cases to consider. The first case has a non-functional candidate goal, as represented in a), b) and c). The second case has a functional candidate goal, represented in d), e), and f).

The SUD architecture is represented as:

$$SUD-Arch = \hat{E}_a SUD-A-a$$

where each $SUD-A-a$ is subcomponent (i.e., subsystem) that is an aggregate of its capabilities, connections, constraints, architectural patterns, architectural styles, and rationale.

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 goals 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 overviews 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 goal. 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.

The COTS component requirements are represented as:

$$COTS-R = \hat{E}_i COTS-i-R,$$

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.)

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.).

The COTS component architecture is represented as²:

$COTS-i-A = \dot{\cup}_a COTS-i-A-a$

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). Each subcomponent is an aggregate of its capabilities, connections, and constraints.

As shown in Figure 1, matching occurs iteratively as the software is developed. The initial matching is at the requirements level (Level 1); subsequent matching is at the architecture level (Level 2). After each search, a set of zero or more components is identified as potential matches.

The matches are described in two cases (refer to Figure 2). The first case has a non-functional goal as the candidate; the second case has a functional goal as the candidate. After matching at the requirements level (Level 1), the two cases are (Figure 2-a and 2-c, respectively):

Case 1. Level 1 Non-functional Match

$\{COTS-i \mid COTS-i-NFR-j \text{ matches } SUD-NFR-l\}$

Case 2. Level 1 Functional Match

$\{COTS-i \mid COTS-i-FR-k \text{ matches } SUD-FR-m\}$

Rank Components. The COTS components that appear to be potential matches are grouped into sets containing COTS components with varying degrees of "satisficing" SUD requirements. After ranking at the requirements level (Level 1), the sets would include (in the order of decreasing degree of satisficing):

For Case 1: Let NFR-l' represent a NFR such as authentication. Then, the sets would include:

$COTS-i-NFR-j-MAKE-SUD-NFR-l' =$

$\{COTS-i-NFR-l \mid COTS-i-NFR-j-MAKE-SUD-NFR-l'\}$

$COTS-i-NFR-j-HELP-SUD-NFR-l' =$

$\{COTS-i-NFR-l \mid COTS-i-NFR-j-HELP-SUD-NFR-l'\}$

For Case 2:

$COTS-i-FR-k-MAKE-SUD-FR-m =$

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

$COTS-i-FR-k-HELP-SUD-FR-m =$

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

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

² Note that the architectural styles, architectural patterns, and rationale for the COTS component are unlikely to be available, and are not included in the representation.

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., response time performance, authentication, etc.) provided by a COTS component are considered, resulting in sets including:

For Case 1:

$\{COTS-i-A-a \mid COTS-i-NFR-j-MAKE-SUD-NFR-l' \dot{\subset} COTS-i-A-a-MAKE-COTS-i-NFR-j\}$

$\{COTS-i-A-a \mid COTS-i-NFR-j-MAKE-SUD-NFR-l' \dot{\subset} COTS-i-A-a-HELP-COTS-i-NFR-j\}$

For Case 2:

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

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

The components are further grouped, this time based on the contributions between the COTS component NFRs and the SUD NFRs. For Case 1, this implies the consideration of other NFRs (NFR-l) interacting with the NFR of concern (NFR-l'). The resulting groups include:

For Case 1:

$\{COTS-i-A-a \mid COTS-i-NFR-j-MAKE-SUD-NFR-l' \dot{\subset} COTS-i-A-a-MAKE-COTS-i-NFR-j \dot{\subset} COTS-i-NFR-j-MAKE-SUD-NFR-l'\}$

$\{COTS-i-A-a \mid COTS-i-NFR-j-MAKE-SUD-NFR-l' \dot{\subset} COTS-i-A-a-HELP-COTS-i-NFR-j \cap COTS-i-NFR-j-HELP-SUD-NFR-l'\}$

For Case 2:

$\{COTS-i-A-a \mid COTS-i-FR-k-MAKE-SUD-FR-m \dot{\subset} COTS-i-A-a-MAKE-COTS-i-NFR-j \dot{\subset} COTS-i-NFR-j-MAKE-SUD-NFR-l'\}$

$\{COTS-i-A-a \mid COTS-i-FR-k-MAKE-SUD-FR-m \dot{\subset} COTS-i-A-a-MAKE-COTS-i-NFR-j \dot{\subset} COTS-i-NFR-j-HELP-SUD-NFR-l'\}$

Negotiate Changes. 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 3). 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

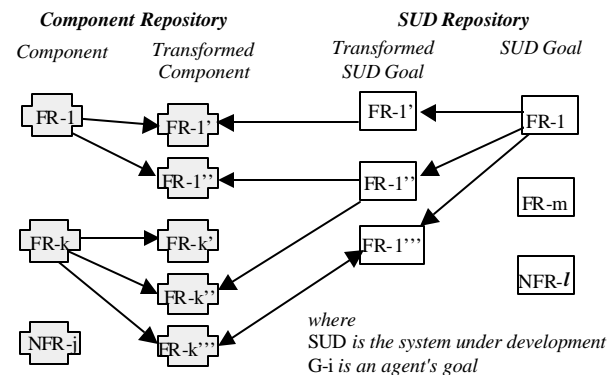


Figure 3. Negotiating Changes (Bridging the Gaps Between SUD Requirements and the Capabilities of the COTS Components).

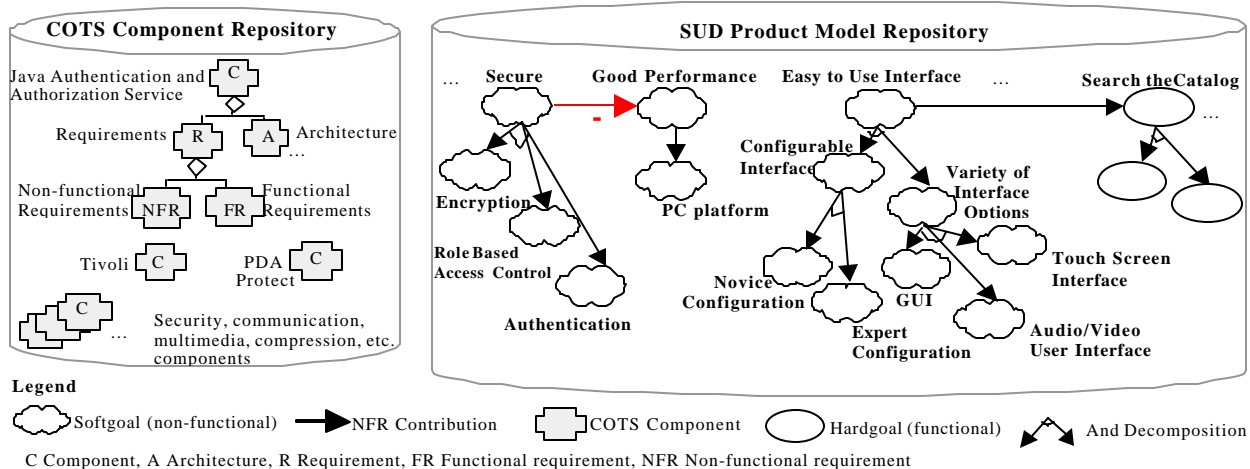


Figure 4. Preliminary Softgoal Interdependency Graph and COTS Component Repository (This illustrates how part of the CARE/SA knowledge base is populated for a DLS before the components are matched, ranked and selected).

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.

3. Illustration

A Digital Library System (DLS) has been used to validate the CARE/SA approach. Figure 4 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 currently available in the marketplace. These components have been found by searching the Internet and specified using information available from the vendors. This is quite variable, where the missing information about the components needs to be obtained by a component engineer by contacting the vendor or empirically evaluating the component. 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. As the CARE approach is applied, COTS components that may realize functional and non-functional goals of the SUD are identified using the matching, ranking, and selection process. The relationships between the COTS components in the repository and the goals for the SUD are explicitly defined, and their contributions are evaluated.

3.1 Model Components

Of interest in this example are components that provide access and authentication capabilities. The CE

does an Internet search and identifies a collection of components including IBM's Tivoli Access Manager for e-business [21], Sun's Java authentication and authorization service (JAAS) [10], and Transaction Security, Inc.'s PDA Protect [18]. The functional and non-functional goals available from the product information posted about the products are extracted; the components are added to the repository. For example, part of the JAAS component in the repository is described informally below:

```

Unique Identifier: C_021
Type:Software
Name:Java authentication and authorization service (JAAS)
Keyword: [security, authentication, authorization, access control] //these are instances of the class keyword
Overview: "The JAAS enables developers to authenticate users and enforce access controls upon those users. JAAS can be used to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet."
Vendor: "Sun Microsystems Inc."
VendorEvaluation: "Sun Microsystems, Inc. is a leader in the Java language specification and tool support"
Webpage: "java.sun.com/products/jaas/overview.html"
Version Number: "1.0"
Interface: [API, binary]
Environment: "Java 2 SDK, v1.3"
Domain: [distributed, non-distributed]
Standards Compliance: "implements a Java version of the standard Pluggable Authentication Module (PAM) framework."
// Keyword instances
security
  name: "security"
  weight: 5
authentication
  name: "authentication"

```

```

weight: 4
...
access control
  name: "access control"
  weight:4

```

The keywords and their weights are used in a keyword based search to order the results presented to the RE.

The components are represented formally in the repository using the Telos [15] notation.

3.2 Match Goals

The RE searches for components in the repository that have goals that may match the goals of the SUD. The RE searches the repository for components to provide security capabilities including authentication and access control. The RE does a keyword-based search using "security", "authorization", "access control". Preliminary information, including the names and the overviews of the three components in the repository that match are returned. Based on this information, two of the components appear to be possible matches; one appears to be specific for personal data assistants (PDA):

First component in search results:

Name:Java authentication and authorization service (JAAS)

Overview: The JAAS enables developers to authenticate users and enforce access controls upon those users with APIs. JAAS can be used to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet.

Second component in search results:

Name:Tivoli Access Manager for e-business

Overview: The Tivoli Access Manager for e-business integrates with e-business applications. It provides

authentication and authorization APIs and integrates with application platforms such as J2EE™.

Third component in search results:

Name: PDA Protect

Overview: The PDA-Protect is a remote authentication technology based upon the submission of a secret sign from a personal data assistant (PDA) device. It is used to control the release of the password to enable the device at power-up or whenever a password is required. The user does not need to remember or enter a password each time it is used; a record of a password may be stored in a secure location for access.

The RE requests additional information about the components to determine which may be possible matches by performing another search on the repository. Details about the components are returned. For example, a significant attribute of the PDA Protect component is the required operating system is the Pocket PC 2002, an operating system for mobile, personal data assistants. The Tivoli and the JAAS components, however, do not have operating system restrictions, as they are Java based.

3.3 Rank and Select Components

Using the NFR Framework, the components are ranked by the RE. The PDA Protect component is placed into the "breaks" group of components, because the DLS is not intended to be a mobile, PDA based application. At this point, however, the Tivoli component and the JAAS component appear to be possible matches, and are placed into the "helps" group.

Given the current understanding of the SUD and the components, the JAAS and the Tivoli components are both selected as possible components (refer to Figure 5). In subsequent iterations, based on the detailed

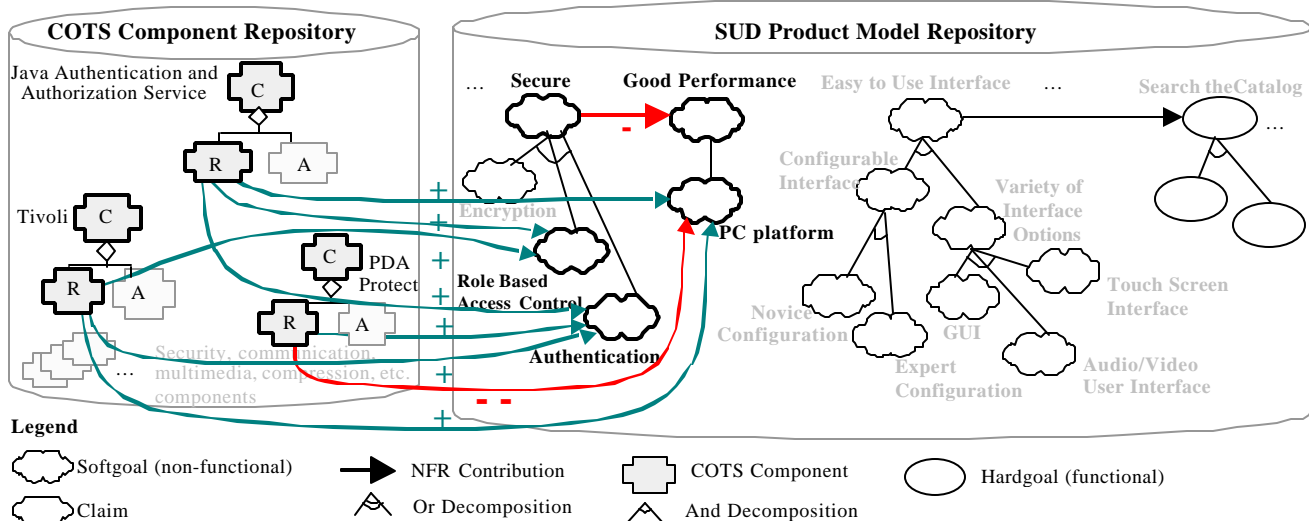


Figure 5. Results of Matching, Ranking, and Selecting COTS components (This illustrates how part of the CARE/SA knowledge base is populated for a DLS after the Goals of the SUD are matched, ranked and selected with respect to the Goals of the components).

requirements and architecture of the SUD and the components, these two components are re-evaluated. As there are two possible components identified, changes to the COTS components or the SUD are not deemed necessary at this time; negotiations to change them are not performed.

4. Conclusions

Here we have presented the CARE/SA Framework, an ongoing work for supporting the matching, ranking, and selection of COTS components, during the development of software/system architectures. The Framework represents a COTS component as an aggregate of its requirements, both functional- and non-functional requirements, and its architecture - each with its own set of attributes. The levels are necessary to understand and use the requirements of the COTS components in supporting the elicitation, specification and validation of the system-under-development (SUD) requirements, as well as the design of the SUD architecture. Matching SUD requirements to COTS components can be done using keyword or case based reasoning. This approach has been applied to a Digital Library System. There are several lines of future research, including the investigation of more heuristics for matching, ranking and selection of COTS components, a meta-model for the CARE/SA process and product, and web-based tool support.

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] Basili, V.R. and Boehm, B., "COTS-based systems top 10 list", *Computer*, 34(5), May 2001, pp. 91-95.
- [3] Boehm, B. "Requirements that handle IKIWISI, COTS, and Rapid Change", *IEEE Computer*, 33(7), July 2000, pp. 99-102.
- [4] 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.
- [5] Chung, L. and Cooper, K., "Defining Agents in a COTS-Aware Requirements Engineering Approach", Proc., 7th Int. Australian Workshop on Requirements Eng., 2002.
- [6] 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.
- [7] Chung, L. and Cooper, K., "Defining Goals in a COTS-Aware Requirements Engineering Approach", *System Engineering journal*, 7(1), 2004, pp. 61-83.
- [8] Chung, L. and Cooper, K., "Matching, Ranking, and Selecting Components: A COTS-Aware Requirements Engineering and Software Architecting Approach", Int. Workshop on Models and Processes for the Evaluation of COTS Components, co-located with ICSE 2004, to appear.
- [9] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J., *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing, 2000.
- [10] Java authentication and authorization service product description, available at: <http://java.sun.com/products/jaas/overview.html>
- [11] Kolodner, J. *Case-Based Reasoning*, Morgan Kaufmann Publishers, San Mateo, CA. 1993.
- [12] Kroll, P. and Kruchten, P., *The Rational Unified Process Made Easy*, Addison-Wesley, 2003.
- [13] Kruchten, P., "Modeling Component Systems with the Unified Modeling Language", *Int. Workshop on Component-Based Software Eng.*, 1998.
- [14] Maiden, N. and Ncube, C., "Acquiring COTS Software Selection Requirements", *IEEE Software*, March/April 1998, pp. 46-56.
- [15] Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M., "Telos: Representing Knowledge about Information Systems", *ACM Transactions on Information Systems* 8 (4), October 1990, pp. 325-362
- [16] Ncube C. and Maiden N., "Guiding parallel requirements acquisition and COTS software selection", *Proc., IEEE Int. Symp. on Requirements Eng.*, 1999, pp. 133-140.
- [17] Nilsson, N.J., *Problem Solving Methods in Artificial Intelligence*, McGraw Hill, USA, 1971.
- [18] PDA Protect product description, available at: http://whitepapers.informationweek.com/detail/PROD/1072086990_744.html&src=iw
- [19] Shaw, M. and Garlan, D., *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.
- [20] Simon, H., *The Science of the Artificial*, Second edition, Cambridge, MA., The MIT Press, 1981.
- [21] Tivoli product description, available at http://www-306.ibm.com/software/tivoli/products/access_mgr-e-bus/
- [22] Voas, J., "The Challenges of Using COTS Software in Component Based Development", *IEEE Computer*, June 1998, pp. 44-45.