

COTS-Aware Requirements Engineering: The CARE Process

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

Sam Courtney

Rational software
IBM Software Group
scourtney@us.ibm.com

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 (COTS-Aware Requirements Engineering) Framework, a process-oriented framework 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. We also present a study of a Home Appliance Control System, with emphasis in the security aspect of the system.

1. Introduction

At the heart of an effective COTS-aware 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, hereafter *SUD*. This paper presents the *CARE* (COTS-Aware Requirements Engineering) Framework, a process-oriented framework which assists 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 *SUD*. 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 *SUD*, 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 repository and the stakeholders' goals for the *SUD*. Throughout the *CARE* process, the decisions and the rationale for matching, ranking, and selecting the components as candidates are maintained, for future evolution of the system.

The *CARE* Framework can be viewed as an extension to current methodologies with a systematic approach to match, rank, and select COTS components. While [7] applies the framework to a digital library system, this paper, among other things, applies the framework to a home appliance control system. The Rational Unified Process (*RUP*) is an object oriented software engineering technique [10] which is based on four phases (transition, construction, elaboration, and 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 [11]. Procurement Oriented Requirements Engineering (*PORE*) supports the evaluation and selection of COTS components [13]. *PORE*'s process model addresses 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. Model Based Architecting and Software Engineering (*MBASE*) considers success, process, product and property models [3] and is consistent for use with COTS components. Evolutionary Process for Integrating COTS Based Systems (*EPIC*) aims to build, field, and support component-based business and military solutions [1]. Leveraging the *RUP*, *EPIC* simultaneously defines and makes trade-offs among: the stakeholders' needs and processes, the components available in the marketplace, the architecture and design of the system, and programmatic and risks.

Section 2 presents an overview of the *CARE* framework. Section 3 describes an application of the *CARE* process. Conclusions are in Section 4.

2. Overview of the CARE Framework

The *CARE* approach represents a COTS component as an aggregation of its requirements (functional and non-functional) and architectural design. 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 – for determining if the product appears to have some potential for use and warrants further investigation. The attributes stored for a product specification include: list of

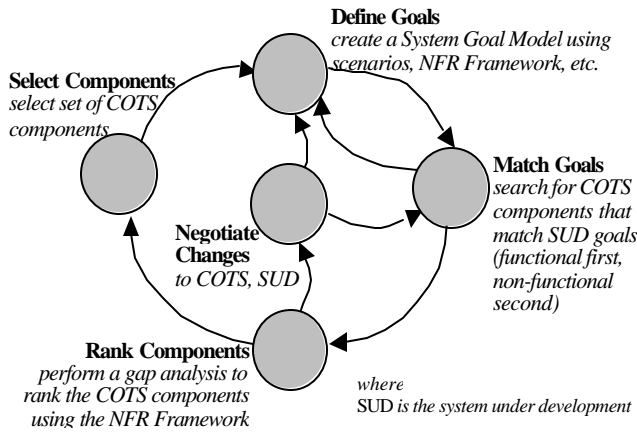


Figure 1. Overview of the CARE Process.

keywords and weights (for keyword and case based searching); functional overview; domain; vendor; standards compliance; performance; security; subcomponents; and lessons learned (e.g., components incompatibilities). The architecture of the COTS components describes the components, connectors, constraints, patterns, styles, etc.[16].

The CARE 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]. 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 is to elicit a set of initial goals. Since goals may be abstract, the RE may need to refine them. For example, a softgoal "the system should be easy to use" leads the RE to ask the question "easy to use in what way?" When interviewed, one stakeholder may intend this goal to mean the system should support a variety of peripheral devices such as PDAs and Smartphones with different user interfaces (stylus, keypad, etc.). Another may mean having convenient remote and local access.

The RE also needs to document the relationships among the goals. A goal-oriented framework is used to characterize such relationships. One such framework is the NFR Framework [8] which provides a set of rankings of the relationships between two (soft)goals: very positive (++), positive (+), negative (-), and very negative (--). In this Framework, goals (softgoals and hardgoals) are organized into a (soft)goal interdependency graph (SIG). The Framework uses the notion of goal *satisficing* to suggest that generated software is "good enough" to meet non-functional requirements (NFRs).

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.

Match Components. The RE identifies 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. 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.

As in Figure 1, matching occurs iteratively. The initial matching is at the requirements level; subsequent matching at the architecture level. After each search, a set of components is identified as potential matches in two cases (refer to Figure 2). The first case has a non-functional goal as the candidate; the second case a functional goal.

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, the sets would include (in the order of decreasing degree of satisficing) sets with the following contributions: make, help, neutral, hurt, and break. Here, *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 provided by a COTS component are considered. The components are further grouped, this time based on the contributions from the COTS component NFRs to the SUD NFRs.

Negotiate Changes. The decisions involved to either negotiate unattained SUD goals or to modify a COTS component are important. 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 requirements, the RE can search for suitable COTS components again.

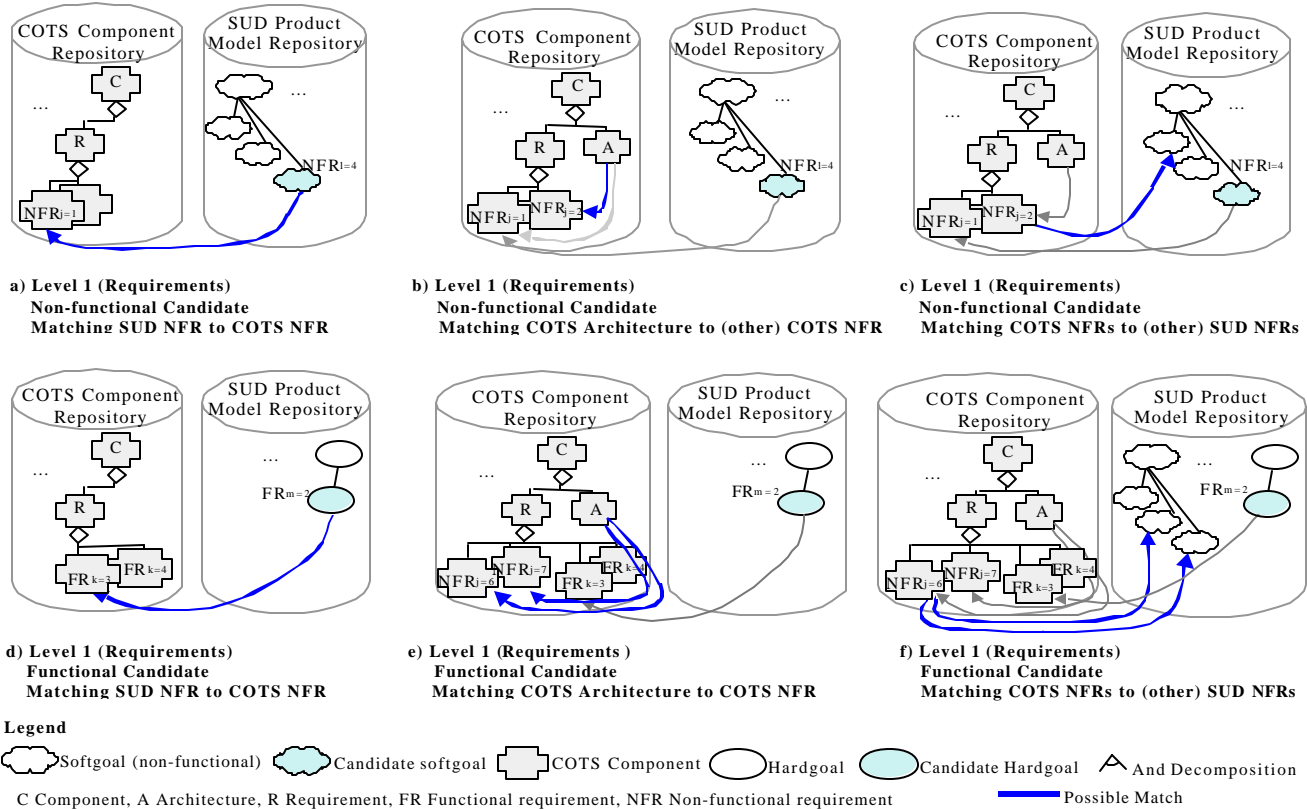


Figure 2. Matching, Ranking, and Selecting COTS Components. (There are two cases to consider. The first case has a non-functional candidate goal, as represented in a), b) and c). The second case has a functional candidate goal, represented in d), e), and f).

3. Study: Home Appliance Control System

Figure 3 shows how part of the CARE knowledge base can be used for HACS. On the left, a COTS repository contains a collection of components, that are currently available in the marketplace, for a variety of security capabilities, including Sun's Java authentication and authorization service (JAAS) [9]. While 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 and the SUD goals are explicitly defined, and their contributions are evaluated.

Define Goals. The RE elicits and defines functional and non-functional goals of the HACS: The system shall be used via a remote system such as a mobile phone, to control home appliances such as air conditioner and garage door opener; The system shall provide security, ease of use, etc. (a SIG on the right side of Figure 3).

Match Goals. The RE searches for components in the repository whose goals may match the SUD goal. The RE begins with a keyword-based search using "security" and "authentication". Preliminary information, including the name, vendor, and overview for the matching components are returned. Some of the matches are:

- Bluefire Mobile Firewall/ Bluefire Mobile Firewall Plus [2], a component for mobile and wireless applications, providing firewall protection, intrusion prevention, integrity

management, enforced authentication, FIPS 140-2 validated encryption and enterprise security management features.

- JAAS [9], a Java SDK 1.4 package that provides authentication and access controls. Java applications, applets, beans, or servlets are supported.
- PDA Protect [14], a component that provides convenient authentication capabilities for Pocket PC 2002 devices.
- Tivoli Access Manager for e-business [17], a component that provides authentication and authorization APIs and integrates with application platforms such as J2EE™.

One of the matching components is a PDA firewall. A firewall had not been considered up to this point; this brings up a number of questions to ask the stakeholders:

- Is a PDA firewall needed (e.g., support data integrity)?
- How does this affect the other SUD requirements?
- How does this affect the selection of COTS components?
- How does this affect the initial concept of the architecture?
- ...

These questions re-initiate the activities to define the goals, as well as match, rank and select components.

Rank and Select Components. Using the NFR Framework, the components are ranked by the RE into sets that have make, help, neutral, hurt, or break contributions. The Tivoli and JAAS components are placed into the "breaks" group of components, because the HACS is not intended to be an information system.

The information about the supported operating

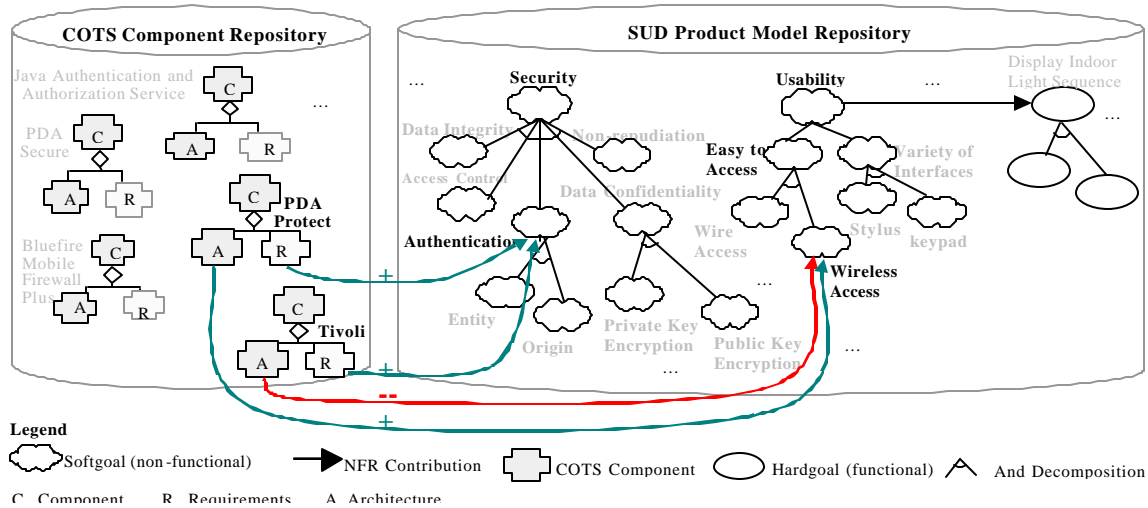


Figure 3. Applying the CARE Approach to a HACS Example. This partial example illustrates the COTS components on the left, SUD artifacts on the right, and relationships between the components and the SUD artifacts.

systems for the PDAsecure and PDA Protect do not include Pocket PC 2003. As a result, the components are ranked as "help" for the Pocket PC 2002 NFR and "neutral" for the Pocket PC 2003 NFR. The RE needs to request additional information from the CE to clarify this. For example, the CE could determine if a version of the components is under development or planned to support Pocket PC 2003.

Given the current understanding of the SUD and the components, the PDAsecure and PDA Protect components are both selected as possible components. In subsequent iterations, based on the detailed requirements and architecture of the SUD and the components, these two components are re-evaluated.

4. Conclusions

Here we have presented the CARE Framework, an ongoing work for supporting the matching, ranking, and selecting 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. Here, the Framework has been applied to a Home Appliance Control 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 process and product, web-based tool support, search techniques to consider more domain characteristics. We also plan to investigate how to consider user operational processes and cope with interactions among architectural constituents.

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] Bluefire Mobile Firewall Plus, product description: http://www.bluefiresecurity.com/mobile_firewall_plus.php

[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., 7th 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. and Cooper, K., "COTS-Aware Requirements Engineering and Software Architecting", Int. Workshop on Systems and Software Architecting, 2004, Las Vegas, NE.

[8] Chung, L., Nixon, B., Yu, E., and Mylopoulos, J., *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishing, 2000.

[9] Java authentication and authorization service product description: <http://java.sun.com/products/jaas/overview.html>

[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] Mylopoulos, J., Borgida, A., Jarke, M., and Koubarakis, M., "Telos: Representing Knowledge about Information Systems", *ACM TOIS*: 8 (4), Oct. 1990, pp. 325-362

[13] Ncube C. and Maiden N., "Guiding parallel requirements acquisition and COTS software selection", *Proc., IEEE Int. Symp. on Requirements Eng.*, 1999, pp. 133-140.

[14] PDA Protect product description: http://whitepapers.informationweek.com/detail/PROD/1072086990_744.html&src=iw

[15] PDAsecure product description: <http://www.pcgardiantechologies.com/PDAsecure/index.html>

[16] Shaw, M. and Garlan, D., *Software Architecture: Perspectives on an Emerging Discipline*, Prentice-Hall, 1996.

[17] Tivoli product description available at: <http://www-306.ibm.com/software/tivoli/products/access-mgr-e-bus>