Towards a Model-based COTS-aware Requirements Engineering Process

Lawrence Chung
Dept. of Computer Science
The University of Texas at Dallas
P.O. Box 830688
Richardson, TX, USA
chung@utdallas.edu

Kendra Cooper
Dept. of Computer Science
The University of Texas at Dallas
P.O. Box 830688
Richardson, TX, USA
kcooper@utdallas.edu

Abstract

The goals of developing systems better, faster, and cheaper continue to drive software engineering practitioners and researchers to investigate software engineering methodologies. In requirements engineering, the focus has been on modeling the software engineering process and products for systems that are being built from scratch. As the size and complexity of systems continues to grow the use of commercial off the shelf (COTS) components is being viewed as a solution. Effective use of COTS components, however, requires a systematic approach that provides both a set of concepts for modeling the subject matter and a set of guidelines for using such concepts.

In this paper, we present a preliminary version of a goal and agent oriented software engineering process model that explicitly addresses the use of COTS components. More specifically, we present a model for a COTS-Aware Requirements Engineering (CARE) process and illustrate it using a Digital Library System.

1 Introduction

Models are used to embody abstract ideas in a concrete representation. In a model, the ideas are more easily communicated, reviewed, and revised. The properties we seek in a model include that the model be complete, consistent, correct, and concise. We also recognize that a model needs both an ontology and a methodology. An ontology is a formal description of entities and their properties; it forms a shared terminology for the objects of interest in the domain, along with definitions for the terms. A methodology describes how the entities are related to one another.

In requirements engineering, models are used to represent both the methodologies and the requirements specifications. Our work is focused on creating and modeling a requirements engineering approach that explicitly supports the use of commercial off the shelf (COTS) components. Our approach is called the COTS-Aware Requirements Engineering (CARE) approach. It is both goal oriented (e.g., see [1,2,3]) and agent oriented (e.g., see [4]).

There are two models that describe the CARE approach: a process model and a product model. The process model is the focus of this work. Here we present the process model in the i* framework [4] and demonstrate its use with a Digital Library System application.

The development of the product model for the CARE approach is underway. We recognize that a CARE approach needs to be knowledge-based, simply because without the knowledge of the COTS components, they cannot be used. For describing the COTS components in the knowledge base (KB), the conceptual notations of MBASE [5,6], RUP [7], and ACRE-PORE [8] are adopted at least for the functional aspect. As a result, there are three types of essential concepts, or ontology, namely, object oriented (OO), agent oriented (AO) and goal oriented (GO).

This paper is organized as follows. Following the introduction, we present the issues that motivate the CARE research in Section 2. The process model for the CARE approach is presented in Section 3. To illustrate the application of the CARE approach, an example illustration is presented in Section 4. Conclusions and future work are in Section 5.

2 Why CARE?

In developing an approach that supports the use of COTS components, one of the first steps is to define the desired characteristics of an ideal, CARE approach. We consider characteristics along four, related lines. Firstly, the approach should encourage the development of systems that the customers are going to find useful. Secondly, the approach should be process driven, in order to support the systematic development of a system. Thirdly, the approach should be knowledge based in order to effectively use the COTS components. Lastly, the scope of the approach needs to consider the impact of using COTS over the software development lifecycle. When all four characteristics are considered, a CARE approach is expected to be feasible to use.

2.1 Customer Satisficing

A CARE approach needs to support developing a system that satifies the customer. In reality, however, a customer is not likely to be 100% satisfied with a system. The term *customer satificing* is used to describe that a customer is satisfied enough with the system to use it.

When using a standard software engineering methodology, the requirements are gathered from stakeholders (*native* requirements) and are intended to describe a system that is going to be "good enough".

When developing systems with the goal of maximizing the use of COTS components, it is vital to understand how to specify requirements that strike the optimum balance between describing the desired user functionality and the available COTS products. We call the requirements of the COTS components *foreign* requirements.

In order to satisfice the customer, the CARE approach needs to consider a number of charactersitics including the flexibility of the requirements, matching and selection support, and a goal driven process.

Flexibility of Native Requirements. In a traditional software development lifecycle, the native requirements are gathered, analyzed, revised and baselined. However, in CARE, there may be few, or no, COTS components to meet the native requirements. In order to use COTS components, the customer needs to be flexible, hence their requirements too, and forgo some of their initial requirements.

Matching and Selecting Components. Bridging the gap between native requirements and the set of foreign requirements is not a unidirectional but a bi-directional process: moving either from the native system requirements to the foreign component requirements set or the other way around. Since there can be more than one way to bridge the gap, a CARE approach needs to encourage the exploration of alternatives (matching native to foreign requirements) and provide guidance as to the best selection among the alternatives.

Goal Driven. A CARE approach needs to be goal driven. Rather than begin with the traditional documentation of requirements, a goal oriented approach begins by eliciting the customer's goals. Goals describe very high level objectives for the system as a whole. For example, the customer may state that they would like a "usable and flexible" system. It is important to note that the goals may be met by a combination of human activities and the system developed. When requirements are elicited, they need to be evaluated against the goals of the system. The requirements engineer needs to ask how each requirement supports accomplishing one or more of the goals.

2.2 Method (or Process) Based

A CARE approach needs to have a clearly defined process that provides the requirements engineer with a set of guidelines and heuristics for using the COTS components. The process needs to offer a collection of methods that address bridging the native and foreign requirements, matching, and selecting the COTS components. When re-writing requirements in the bridging process, the process needs to be flexible enough to allow for both correctness-preserving refinements and inconsistent rewriting. In addition, the process should not be overly complicated for it to be accepted for use.

2.3 Knowledge-Based

A CARE approach needs to be knowledge-based, simply because without the knowledge of the COTS components, they cannot be used. In a knowledge based approach, there are three types of essential concepts, or ontology, namely OO,

AO, and GO. Using OO, knowledge of software systems, both foreign and native, is captured for their functional requirements. Using AO and GO, coupled with scenario analysis and descriptions of experiences, knowledge of the enterprise or context is captured. Knowledge of system-wide properties is represented and reasoned about as knowledge of non-functional requirements.

2.4 Scope of a CARE Approach

A CARE approach should consider the impact on the system architecture, the long term maintenance of the system, and the system cost.

Impact on the System Architecture. The use of COTS products also impacts the architecture, or high-level design, phase of the software development lifecycle. In a standard software development lifecycle the architecture is developed to fullfill the native requirements of the system without regard to the availability of COTS components. Using this approach, there may be few or no available products that fit within the chosen architecture [9]. To maximize the use of COTS components, the architects need to consider the distinction between the foreign and the native requirements when developing the architecture.

Impact on Initial System Cost. In comparison to a approach that does not consider the use of COTS products, a CARE approach may use more time in the requirements analysis phase. When considering COTS, the engineer needs to search through the COTS specifications to match and select among the components. The benefits of using a CARE approach are expected to appear near the end of the software development lifecycle, in the detailed design, implementation, and unit testing phases. As a result, the evaluation of the CARE approach needs to be measured with respect to the overall development lifecycle effort.

Impact on the System Maintenance. The impact of using COTS components on the system maintenance may also need to be considered. For systems that are being developed with the expectation of going into a maintenance phase, the complexity of the interfaces of the components and the probability of the vendor being in business in five years may be considered. This is not an issue when a goal is to prototype a system, for example, that is only intended to demonstrate a concept and then be disposed of.

3 Process Model

The scope of the process described here is restricted to the requirements phase of the software development lifecycle. The description does not, for example, consider the design, implementation, testing, and maintenance phases.

3.1 Ontology

The ontology of the process model includes actors, goals, resources, dependencies, decompositions, and means-end relationships. The definitions of these entities are found in the i* modeling framework [4].

The actors in the CARE process model are the customer and the requirements engineer (RE). In this description of the model, the customer is contracting the development of a large scale system and is involved in the development work. Variations of this model can be developed in the future to consider high volume, shrink-wrap product development.

The customer's overall goal is to receive a system that satisfices them. Subgoals include receiving product development artifacts (product goals, system requirements, and software requirements) and product planning artifacts (quality plan, test plan, schedule, etc.). The customer's softgoals include receiving a system that is delivered on schedule, within budget, and with high quality.

The RE depends on the customer to validate the system as it is being developed and provide (ideally) complete and correct information.

3.2 Methodology

The CARE Process methodology is being developed using a set of examples for a Digital Library System [10]. In this work, the process is described in the i* notation (refer to Figures 1-5). In i*, the concepts of the model are embedded into the conceptual modeling language Telos [20]. As a result, i* provides an extensible, object-oriented representational framework with classification, generalization, aggregation, attribution, and time.

The scope of the strategic dependency diagram (refer to Figure 1) focuses on the requirements engineering phase of the software development

lifecycle. As the CARE approach is extended to consider other phases, additional actors, goals, and softgoals are going to be added to the model.

In Figure 2, the RE actor is decomposed using the Strategic Rationale Model. The model describes how the RE can accomplish the goals of creating the baselined system goals and the baselined system requirements. Currently, our process model does not extend to creating the baselined software requirements.

The task of creating the Baselined System Goals (refer to Figure 3) can be decomposed into a set of five subtasks (Elicit Initial Goals, Analyze Goals, Correct Goals, Validate Goals, Baseline Goals). The RE, an intentional agent, decides when to use each of the subtasks to accomplish the goal of Creating the Baselined System Goals.

The subtask of Defining the COTS Requirements (refer to Figure 4, 5) can be decomposed into a set of nine subtasks. Each of these is briefly described below.

Select Candidate Requirements. The RE evaluates each system requirement and determines if it is a candidate for implementing with one or more COTS components.

Preliminary Search. The RE performs a search on the repository that returns high level descriptions of the components that match the search criteria.

Preliminary Match. The RE evaluates the results of the preliminary search and determines which of the components may be a possible match to the functionality needed.

Detailed Search. The RE performs a search on the repository for the components that satisfy the preliminary match. Detailed specifications of these components are returned.

Detailed Match. The RE evaluates the results of the detailed search and determines which of the components are a close or exact match to the functionality needed.

Select Component. The RE selects one or more components that are an exact match for the functionality needed.

Request to Change Component. The RE sends a request to the vendor for a change to a component

Request to Change Requirement. The RE sends a request to a stakeholder for a change to a requirement.

Maintain Components. The RE adds, updates, or deletes the detailed specifications of the components in the repository.

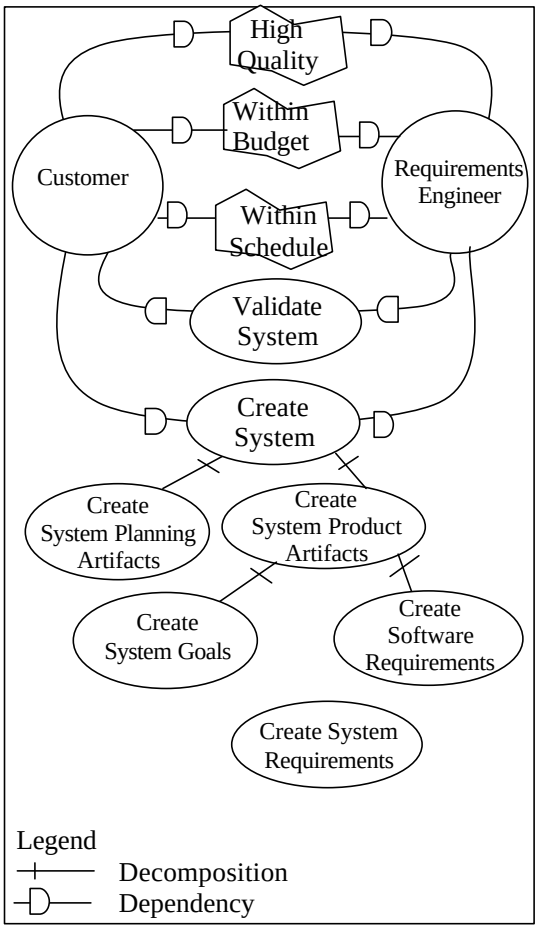

Figure 1. Strategic Dependency Model

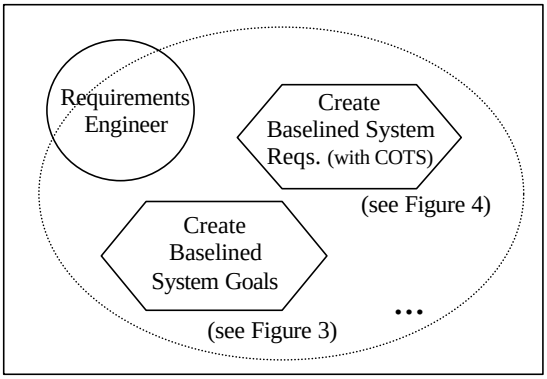

Figure 2. Strategic Rationale Model (RE)

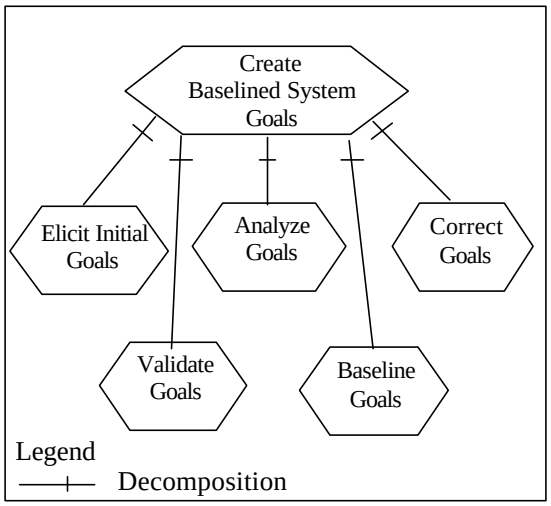

Figure 3. Create Baselined System Goals

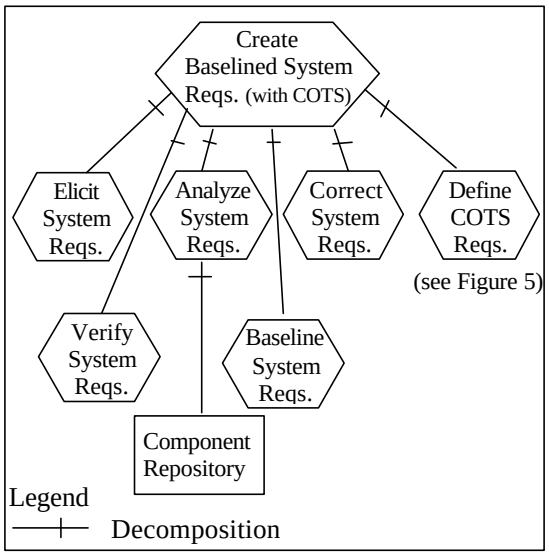

Figure 4. Create Baselined System Requirements (with COTS)

4. Illustration

The CARE approach is being developed iteratively by working through a set of examples in a Digital Library System. The initial CARE process is based on the example used to develop the system requirements (with COTS) for the database component in the system [10]. In this work, we present another example in order to validate the initial process. The example is for a communication component in the system.

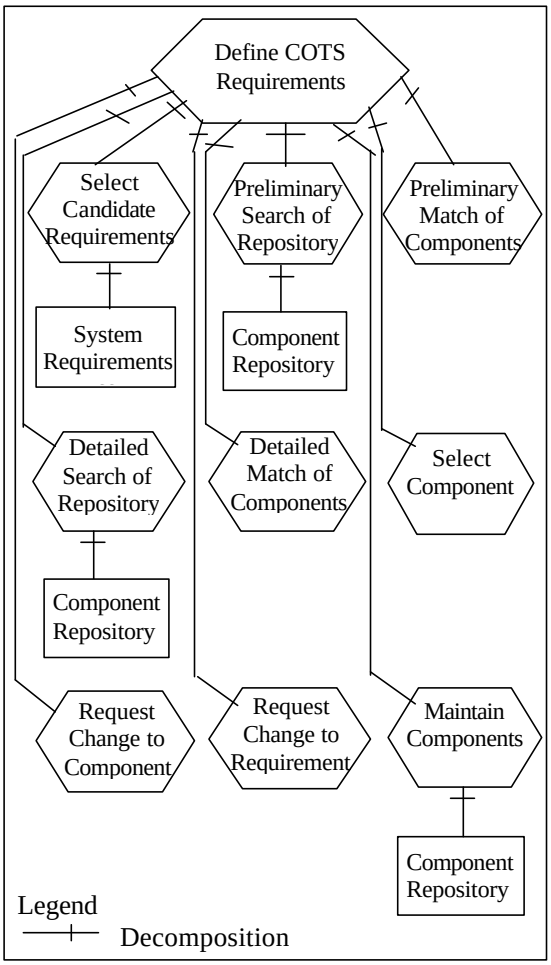

Figure 5. Define COTS Requirements

4.1 Create Baselined Set of Enterprise Goals

The goal of describing the Baselined Enterprise Goals is workable because the RE has a routine to use. In this example, the routine to describe the set of Baselined Enterprise Goals is composed of the following tasks (refer to Figure 3):

Elicit Initial Set of Goals
Analyze Goals
Correct Goals
Baseline Goals

The RE's routine is described below.

Elicit Initial Set of Enterprise Goals. Here, the digital library (DL) refers to the digital library organization as a whole. It may consist of people, software, hardware, and interfaces to external

agents. The digital library system (DLS) refers to the goals that need to be met by the software system. The initial set of goals are:

1. The DLS should be delivered with high quality, on time, and within budget
2. The DL should comply with current standards
3. The DL should be easy to use
4. The DL should have high availability
5. The DL should have fast performance
6. The users should be able to access a large number of diverse objects
7. The users should be able to search, browse, and retrieve objects quickly and efficiently
8. The librarians should be able to maintain the library quickly and efficiently
9. The librarians should be able to provide reference support quickly and efficiently
10. The administrators should be able to maintain users' accounts quickly and efficiently
11. The DL should be scaleable
12. The DL should be secure
13. The DL should be inexpensive to operate

In this example, we select two, initial goals to refine into system level requirements (with COTS) using our CARE process:

1. The DLS should be delivered with high quality, on time, and within budget
2. DL should comply with current standards

The first goal drives the use of COTS components. The use of COTS is viewed as a solution to the problem of how to develop systems that are increasingly complex with high quality, on time, and within budget. This goal needs to be refined in a quality plan, project schedule, and project budget and are not considered in the refinement process for technical requirements. The second goal is considered in the analysis task.

Analyze Goals. The second goal may be achieved by people, hardware, or software within the organization (i.e., the library) or by external systems (described by interfaces). In the CARE work, we focus on the software system. Here, the RE interprets the meaning of the goal to be that the customer would like the system to adhere to the latest technical standards that are applicable to a DLS. The analysis of this goal determines which standards (de-facto or international) are relevant to the DLS. Domain experts and publicly available documents describing currently deployed digital library systems are sources of information. The initial list of standards includes:

2a. ANSI/NISO Z39.50-1995 [11]
2b. Dublin Core [12]
2c. XML [13]
2d. EAD [14]
2e. MARC [15]
2f. AACR2 [16]

Correct Goals. Domain experts validate the identified standards and note that the following standards are missing:

2g. ebXML [17]
2h. OEBPS [18]

After investigating these standards and understanding why they need to be included, the RE corrects the problem and adds the standards to the refined goals.

Baseline Goals. After analysis and correction, the RE determines that the refinement step is complete and the goals are ready to be baselined. A heuristic the RE may apply to determine if the refinement task is complete is this: if further refinement results in a statement that can be tested (from a test engineer's perspective), then the goal is completely refined. When the goals are completely refined, the RE places the set of goals under configuration management.

4.2 Create Baselined Set of System Requirements (with COTS)

The goal of describing the set of Baselined system requirements (with COTS) is workable because the RE has a routine to use. In this example, the routine to describe the set of Baselined system requirements is composed of the following tasks (refer to Figures 4 and 5):

Elicit Initial Set of System Requirements
Select Requirement as Candidate for COTS
Preliminary Search for Components
Preliminary Matching for Components
Detailed Search for Components
Detailed Matching for Components
Select Component
Correct System Requirements
Baseline System Requirements

This routine is described below.

Elicit Initial Set of System Requirements. For this example, we focus on the baselined goal of

complying to the ANSI/NISO Z39.50-1995 standard. This standard specifies a client/server based protocol for Information Retrieval across the Internet. Like many international standards, the Z39.50-1995 is quite large. It specifies procedures and structures for a client to search a database provided by a server, retrieve database records identified by a search, scan a term list, and sort a result set. Access control, resource control, extended services, and a "help" facility are also supported. The protocol addresses communication between corresponding information retrieval applications, the client and server [4].

As an initial system requirement, the description of the goal is re-used as the system requirement:

2a1. ANSI/NISO Z39.50-1995

For standards that are going to be implemented using COTS components, it may not be necessary to identify and describe precisely which parts of the standards are needed because a component that complies with the standard is expected to meet the entire standard. Therefore, at this point it is important for the RE to determine if the requirement is a candidate for COTS component. If it is a candidate, then the requirement can be left at this level. Otherwise, a detailed statement of compliance to the standard is needed.

Select Candidate System Requirements. The RE selects the system requirement

2a1. ANSI/NISO Z39.50-1995

as a candidate for COTS component. This is an international, communication protocol standard for the Internet and is likely to have COTS components available.

Preliminary Search. The RE performs a preliminary search in the repository using the keyword Z39.50. The results of the search include the following components:

1. Follett Software Company, Follett Software Z39.50 version 2
2. Sunstone Systems, Sunstone Z39.50 version 2
3. Blue Angel, Metastar Gateway component Z39.50 version 3
4. Blue Angel, Metastar Server component, Z39.50 version 3.

Preliminary Match. The RE reviews the results of the preliminary search and determines that the Blue Angel components are preliminary matches. The components by Sunstone and Follett only support version 2 of the standard and are not considered any further.

Detailed Search. The RE performs a detailed search on the preliminary matches. The following information is returned:

Blue Angel Metastar Server component is compliant with Z39.50 version 3.
Platforms: Windows NT and UNIX platforms.
Performance: N/A
Scaleability: N/A

Blue Angel Metastar Gateway component is compliant with Z39.50 version 3.
Platforms: Windows NT and UNIX platforms.
Performance: N/A
Scaleability: N/A
Notes: integrated with the Altavista and Fulcrum search engines; provides Z39.50 client

Detailed Match. The RE performs a detailed match. Based on the information in the repository, the Blue Angel components match the requirement because they provide the Z39.50 version 3 client and server capabilities.

Select Component. Based on the information in the repository, the RE selects the Blue Angel Gateway and Server components to provide the Z39.50 client and server functionality.

Correct System Requirements. Due to the selection of the COTS component, the RE updates the system requirements with a new requirement: The ANSI/NISO Z39.50-1995 shall be implemented using a COTS component.

Baseline System Requirements. In this task, the RE places the corrected system requirements under configuration management.

5. Conclusions and Future Work

We have presented a preliminary process model for a requirements engineering approach that explicitly supports the use of COTS components. The ontology and methodology for the model is described and an example is given to demonstrate the applicability of the model to a Digital Library System. We recognize that the model needs to be refined to address complex issues in the use of COTS components including: the coupling of non-

functional requirements for a component (e.g., memory, performance tradeoffs), incompatibility of COTS components, and bridging the gap between customer requirements and the capabilities available in COTS components [3,19].

There are a number of important aspects of the research to investigate. Our next step is to refine the process by working through a more complex illustration in the Digital Library System. The next illustration will show conflicting goals and conflicting requirements, and use some portions of the process model that have not been used in this example (e.g., Request a Change to a COTS Component, Request a Change to a Requirement, Maintain the Components in the Repository).

As the process model is refined and validated, the product model for the CARE approach also needs to be developed. Defining the product model which would allow the consistent use of a "mixed bag" of the various OO, AO, and GO ontologies and methodologies for CARE is expected to be challenging.

Our CARE approach currently addresses three of the four characteristics of an ideal, CARE approach. Our CARE approach supports developing systems that satisfice the customer, is method based, and is knowledge based. In the future, we will extend the approach to consider the impact over the software development lifecycle.

References

1. Annie I. Antón and Colin Potts, "The Use of Goals to Surface Requirements for Evolving Systems", *International Conference on Software Engineering*, Kyoto, Japan, pp. 157-166, 19-25 April 1998.
2. A. van Lamsweerde and E. Letier, "Handling Obstacles in Goal-Oriented Requirements Engineering", *IEEE Transactions on Software Engineering*, Special Issue on Exception Handling, Vol. 26, September 2000.
3. L. Chung, B. Nixon, E. Yu and J. Mylopoulos, Non-Functional Requirements in Software Engineering, Kluwer Academic Publishing, 2000.
4. E. Yu, "Modelling Strategic Relationships For Process Reengineering", DKBS-TR-94-6, The University of Toronto, Canada, December 1994.
5. B. Boehm, D. Port, M. Abi-Antoun, and A. Egyed, "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.
6. H. In and E. Flores-Mendoza, "Initial Design of the "Plug-n-Analze" Framework for Architecture Tradeoff Analysis", COMPSAC99, Phoenix, Arizona, USA, October, 1999.
7. I. Jacobson, G. Booch, and J. Rumbaugh, The Unified Software Development Process, Addison Wesley Longman, Inc., USA, 1999.
8. C. Ncube and N. Maiden, "Guiding parallel requirements acquisition and COTS software selection", *Proceedings of the IEEE International Symposium on Requirements Engineering* 1999, pp. 133-140.
9. L. Brownsword, T. Oberndorf, and C.A. Sledge, "Developing new processes for COTS-based systems", *IEEE Software*, Volume: 17 Issue:4, July-Aug. 2000 pp. 48 –55.
10. L. Chung, K. Cooper, and D.T. Huynh, "COTS-Aware Requirements Engineering Technique", submitted to ICCBSS '01.
11. ANSI/NISO Z39.50-1995 Information Retrieval (Z39.50): Application Service Definition and Protocol Specification, Z39.50 Maintenance Agency, July 1995. (ISO 23950)
12. S. Weibel and J. Kunze, "Dublin Core Metadata for Resource Discovery", IETF RFC 2413, September 1998.
13. World Wide Web Consortium (W3C), "Extensible Markup Language (XML), Recommendation 6, October 2000.
14. Encoded Archival Description Document Type Definition (EAD DTD), version 1.0, Network Development and MARC Standards Office, August 1998.
15. Machine Readable Cataloging (MARC) 21, Format for Bibliographic Data, update No. 1, Network Development and MARC Standards Office, October 2000.
16. Anglo-American Cataloguing Rules, Second Edition, Canadian Library Association, Library Association Publishing, American Library Association, 1999.
17. Electronic Business XML (ebXML), ebXML Technical Architecture Specification version 1.04, The United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) and The Organisation for the Advancement of Structured Information Standards (OASIS), February 2001.
18. Open eBook Publication Structure (OEBPS) specification, version 1.0.1, July 2001.
19. J. Mylopoulos, L. Chung, and B. Nixon, "Representing and using nonfunctional requirements: a process-oriented approach" *IEEE Transactions on Software Engineering*, Volume 18, Issue 6, June 1992, pp. 483-497.
20. J. Mylopoulos, A. Borgida, M. Jarke, M. Koubarakis, "Telos: Representing Knowledge about Information Systems", ACM Trans. Info. Sys., 8 (4), pp. 325-362, October 1990.