Requirements Engineering for COTS Selection

Carina Frota Alves[1], Fernanda M. R. de Alencar[2], Jaelson F. B. Castro[1♠]

[1]Universidade Federal de Pernambuco, Centro de Informática,
Av. Prof. Luiz Freire, s/n – Cidade Universitária
CEP 50732-970 Recife, Pernambuco
E-mail: {cfa, jbc}@di.ufpe.br

[2] UFPE - Universidade Federal de Pernambuco
CT - Departamento de Eletrônica e Sistemas
Rua Acadêmico Hélio Ramos, s/n - Cidade Universitária
Recife - PE - CEP: 50.740-530
E-mail: fmra@npd.ufpe.br

Abstract

There is growing interest in the notion of software development through the planned integration of COTS (Commercial Off-The-Shelf) products. The potential advantages of this integration-centric approach are shorter development time and reduced cost. Often a COTS based development process consists of an evaluation, selection, adaptation, integration, and evolution of components obtained from external vendors. However, most methods focus on system adaptation and integration but neglect the processes of evaluation and selection of COTS. This paper introduces a COTS-Based Requirements Engineering Model that focuses on non-functional requirements to assist the processes of evaluation and selection of COTS products.

Key Words: COTS development process, non-functional requirements, product evaluation and selection.

1. Introduction

The construction of software products through the planned and deliberate reuse of previously constructed components has long been heralded (Sommerville 1996) as one of the key challenges in moving software production forward to cost-efficient, planned, engineering discipline. More recently, the idea of reuse-centred software development is becoming know as Component-Based Software Engineering (CBSE) were these components are often COTS (Commercial Off-The-Shelf) products.

The interest in COTS is based on a long history of work in modular systems, structures design and most recently in object-oriented systems. Component-based development has many potential advantages such as shorter time to market, lower prices, and higher quality software solutions. This engineering approach emphasizes the acquisition and integration of reusable COTS products over development from scratch.

The nature of COTS suggests that the model of component-based software development should be different from the conventional development model. It has

resulted in a significant shift away from the development-centric toward a procurement-centric approach. In general, existing software engineering models, such as Waterfall and Spiral models do not address the extensive process and cost associated with the identification, evaluation, selection, and integration of reusable software components in COTS development. As a result, implementing these systems using such models often leads to unrealistic project planning. In this way, many models has been proposed to address the various aspects of the component based development process that have been largely ignored in conventional development models (Tran and Liu 1997).

It looks very promising to use components in order to improve productivity and quality of software development. However, the use of COTS software introduces new problems and risks, including difficulty in selecting suitable components and insufficient requirements analysis. This work discusses some of the problems and challenges raised during the phases of evaluation and selection of COTS products as well as investigates the importance of requirements engineering to obtain a consistent and mature COTS process model. We present the CRE (COTS based on Requirements Engineering) Model, it focuses on non-functional requirements to assist the processes of evaluation and selection of COTS products.

The paper is organized as follows. Section 2 provides a description of essential COTS-based development activities. Section 3 describes some challenges in COTS selection activity. Next section shows the NFR Framework which is an approach to representing non-functional requirements. Section 4 presents the CRE Method and in particular, its life-cycle processes. Finally, section 5 concludes our discussion and shows future work.

2. Essential COTS-based Development Activities

Most COTS life-cycle models encountered in literature consider the activities of identification, selection, integration, and adaptation, as part of the development process to construct systems based on COTS products (Tran and Liu 1997), (Wallnau, Carney and Pollak 1998), see figure 1.We describe briefly each of them following.

COTS market evaluation. This phase includes gathering the overall system requirements, identifying and classifying COTS product into product sets, and prioritizing them for the subsequent selection. The COTS candidates may come from a variety of vendors and hence a process of investigation into the properties and qualities of the COTS is required. These properties include component functionality (i.e. what services are provided), aspects of a component's interface such as the use of standards. Normally, it is also reasonable to discover non-technical aspects such as, vendor reputation and maturity. Evaluation is a difficult and ill-defined process, usually available product information is difficult to analyze and; in some cases, difficult to obtain. There are two extreme visions to evaluate COTS products, the first one and simpler is called *superficial*. This kind of evaluation process consists of buying the components that apparently are suitable to use. This approach ensures minimal cost to the evaluation effort by eliminating products set that failed a

particular evaluation stage and selecting the first one that passes all evaluation stages. The second one is more detailed and is known as *exhaustive.* For the exhaustive approach, all COTS candidates are evaluated in detail through all identified stages. The goal of exhaustive evaluation is to ensure that an optimal product set will be selected for the final integration at the cost of additional evaluation time and resources. In situations where a high level of uncertainty and a large number of candidates exist, the first option offers less risks. Other product evaluation approaches fall between these two extremes.

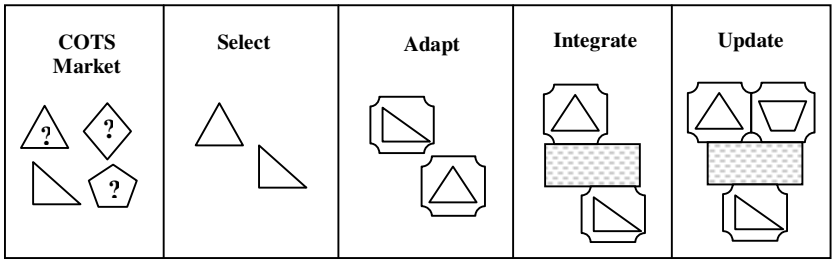

Fig. 1 – Activities of the component-based development process

Component selection. Selecting an appropriate product typically requires trade-off analysis. As a result, establishing the criteria for products selection is a very important task in COTS development. Some criteria are concerned with the product vendors. Others reflect the limitation of time and resources to support evaluation effort, such as the deadline associated with the final product selection decision, and the resources available for the evaluation activity. In COTS-based development, the selection process must occur early in the life cycle. COTS evaluation and selection become a critical part of the early analysis process rather than a peripheral activity within the later design process which occur in conventional development. If an unsuitable component is selected much effort will be necessary to adapt and integrate it into the actual system. A complete selection process includes far more simply considering the desired functionality. To make use of a product, one must also understand non-functional aspects, such as performance, reliability, flexibility, etc; as well as the implicit assumptions made by the product about the operating environment.

Component adaptation. This activity includes the development of all necessary software adapters and enhancements to the selected COTS. Components adaptation should be based on rules that ensure conflicts minimization among components. Normally, scripts are written as a buffer between user request and component actions. However, fault injection techniques can also be used to identify unacceptable behavior exhibited by a COTS component. A common approach is component wrapping, to avoid robustness and dependability problems in COTS software. Wrappers can filter the component's inputs, outputs or both (MacGraw and Viega 1998). This approach disallows COTS software from exhibiting undesired functionality by placing a software barrier around the components; limiting what it can do.

Component integration. The integration phase encompasses all development efforts required to interconnect different selected COTS products into a single integrated

system. This phase also consists of the development of other system's parts that were not supported by commercially available COTS products and testing the final system. Since the efforts often require in-house development, a conventional development approach such as Waterfall or Spiral model should be deployed (Tran and Liu 1997).

Component update. At a first glance, component-based systems may appear to be relatively easy to evolve and upgrade since component's paradigm is based on reuse and change. For example, to repair an error, an updated component may replace a defective one. However, this practice is not so simple. Component replacements is often a time-consuming and arduous task. Without careful planning, a change to one component can have extensive unforeseen repercussions on many others components. In order to mitigate these problems, this activity must be well-supported as an essential activity through appropriate definition of component interfaces and controlled interaction among components.

3. Challenges in COTS Selection

It has been argued that a well defined selection process is the cornerstone for any effective COTS development process (Ncube and Maiden 1999), (Fox and Lantner 1997). However, the COTS selection process is prone to some potentially problems. The decisions made at this point will be critical and will have a considerable impact on project success or failure. Most projects are often assigned under schedule pressure and necessitate quick decision make, usually in the face of unavoidable uncertainty.

The selection of suitable COTS products is often a non-trivial task and requires careful consideration of multiple criteria (Fox and Lantner 1997). We have identified four main dimensions that should be considered during the selection phase, see figure 2.

- **Domain coverage** – The components have to provide all or part of the required capabilities necessaries to meet core essential customer requirements, where non-functional requirements play a critical role during the assessment process. In some cases, extra new components need to be develop to meet the shortfalls.

- **Time restriction** – Software companies normally operate with very rigid development schedule, on which their competitiveness depends. Selection is a time consuming activity, a considerable amount of time is necessary to search and screen all potential COTS candidates.

- **Costs rating** – The available budget is a very important variable. The expenses when selecting a particular COTS will be influenced by factors such as: acquisition license, cost of support, adaptation expenses, and maintenance prices. Aoyama et al. (1998) provides an economic model for estimating the cost of COTS-based system development.

- **Vendor guaranties** – An important aspect to consider during selection activity is verify the technical support provided by the vendor. Some issues

need to be taken into account, for example: vendor reputation and maturity, number and kind of the applications that already use de COTS, clauses characteristics of the maintenance licenses.

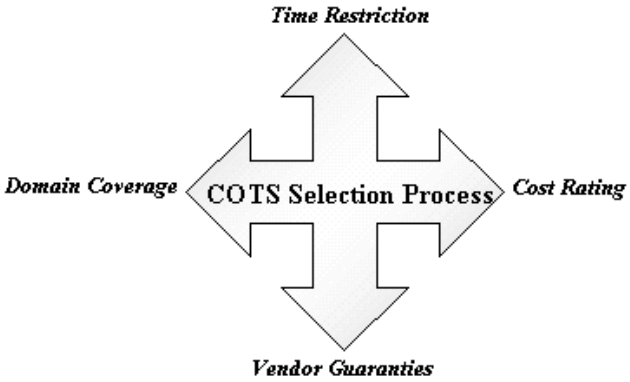

Fig. 2 – Main dimensions of COTS selection

Considering the dimensions presented above, we observe that quality aspects such as: dependability, reliability and robustness are important issues during the selection of COTS products. Unfortunately, most methods do not define these properties adequately during the early phases of development process. Normally, they are deeply examined only in the phases of adaptation and integration. We claim that the NFR framework (Chung, Nixon, Yu and Mylopoulos 2000) is well suitable for representing non-functional requirements. Next section provides a brief description of this approach.

4. Representing Non-Functional Requirements

The NFR Framework is a process-oriented approach where non-functional requirements are explicitly represented as goals to be achieved during the process of system development. Each goal will be decomposed into satisficing goal represented by a graph structure inspired by the and/or trees used in problem solving (Chung, Nixon, Yu and Mylopoulos 2000).

One fundamental premise of this approach is that non-functional requirements have the property of potentially interacting with each other, in conflict or synergy (Chung and Nixon 1995). We explain how this property may be used to systematically guide selection among COTS products. In using the NFR Framework, one constructs an initial goal interdependency graph by identifying the main non-functional requirements that the particular system under development should meet. By treating these high level requirements as goals to be achieved, we can decompose them into more specific subgoals which together satisfice the higher level goals. Goals contribute, positively or negatively, to fulfilling other goals. The NFR framework

includes a stage for Knowledge Acquisition and Application of the framework (Chung and Nixon 1995).

Knowledge acquisition – consists of two activities:

- Acquisition of knowledge specific to NFRs – This activity encodes knowledge about the particular type of requirement into a catalogue, which contains a terminology for the quality requirements, a list of generic techniques, and their tradeoffs and interactions. Normally, system developers can access existing catalogues but they can be extended to deal with additional or more refined concepts.
- Acquisition of domain knowledge – During this activity, the developer acquires and uses information about the domain in which the COTS products will be used. This includes items such as functional requirements, organizational priorities and existing systems.

Application of the NFR-Framework

- Identification of NFR-related concepts – This phase includes identification of important NFR goals and an initial estimate of what is important, critical, etc. It also includes identification of design rationale and recording the relevant arguments used.
- Linking NFR-related concepts – The developer starts with an initial set of goals, and then refine them into other NFR goals. To satisfice NFR goals, the developer considers design alternatives, called satisficing goals, along with their tradeoffs, refines them, makes selections and justifies them by recording design rationale. Throughout this process, the impact of each design decision is propagated towards top-level NFR goals.

These steps are not necessarily sequential, and one may also iterate over them many times during the design process. In the next section, we present the CRE Model and describe how the NFR Framework can help the developer to adequately represent non-functional requirements which drives the selection of suitable COTS candidates.

5. The CRE (COTS-Based Requirements Engineering) Model

The success of a Component-Based development process largely depends on the appropriate selection of COTS software components that meet core customer requirements (Ncube and Maiden 1999). However, the activities of COTS evaluation and selection have received little attention in the literature. Most research in Component-based systems are interested on adaptation and integration processes.

A consensus seems to be emerging in the CBSE community that the COTS development process should be an iterative activity of requirements engineering and COTS processes of evaluation, selection, adaptation, integration and update (Tran and Liu 1997), (Fox and Lantner 1997). Most COTS approaches treat quality aspects only

during the later phases of development. This attitude increases the risks of COTS failure and the costs of the final system. However, Chung (1995) emphasises that non-functional requirements should be addressed as early as possible in the systems lifecycle.

To remedy the problems caused because of later definition of non-functional requirements, a more disciplined approach is needed for improving our ability to understand the high level system constraints and the rationales behind COTS products choices. The CRE Model facilitates a systematic, well-defined and requirements-driven COTS selection process. A key issue supported by this model is the definition and analysis of non-functional requirements during the phases of COTS evaluation and selection.

The selection of COTS products is made by rejection. The products that do not meet customer requirements are rejected and removed from the candidate list. As the candidate list decreases, the number and detail of customer requirements increases. The result of this process is an iterative activity of requirements acquisition that enables the selection of COTS products and this selection process also gives information about user requirements, see figure 3 (Ncube and Maiden 1999).

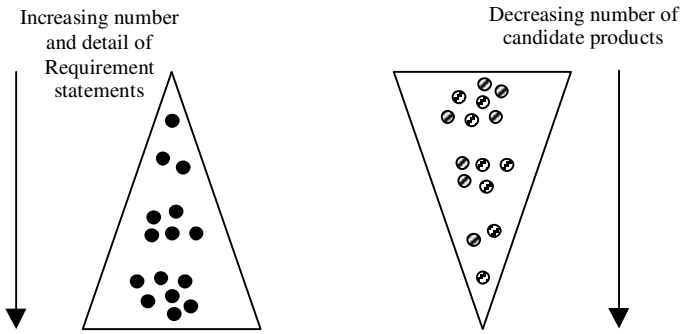

Fig. 3 – Overview of the PORE's iterative process

Within this iterative process, we propose a new model, called CRE that emphasizes quality requirements as a way to enrich the selection process of COTS products. This model has four iterative phases: Product Identification, Product Description, Requirements Acquisition, and Product Acceptance, see figure 4. In the sequence, we describe each phase individually.

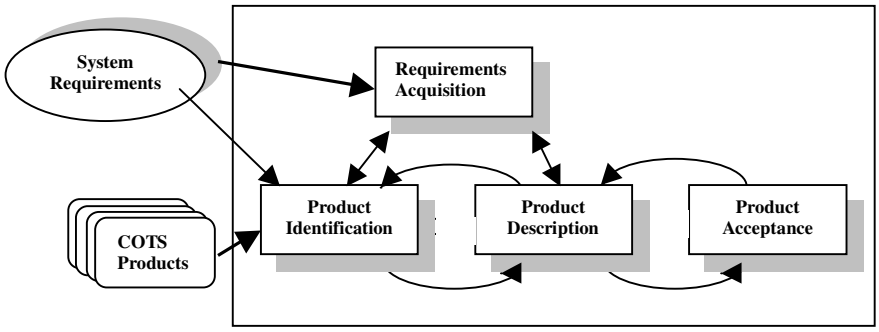

Fig 4 – The CRE Model

5.1 Product Identification Phase

The primary objective of this phase is to identify and find all suitable and potential COTS candidates. This phase is driven by the evaluation criteria which takes as input high level requirements, such as services and limitations under which the component should operate and any revised requirements that are part of the feedback mechanism. At a first stage, the evaluation criteria does not need to be very detailed or formally defined but it is necessary to be unambiguous. Therefore, the identification phase can be initiated as soon as the main features of the required component have been defined.

Card Sorts is a simple and useful technique for acquiring high level requirements that are used as basis for the definition of the evaluation criteria. When using this technique, the requirements engineer writes candidate product names on cards and asks the stakeholders to use the cards to sort the products into categories. Criteria for these sorts, such as "the system must be secure", indicate customer requirements that discriminate between products. Discriminating requirements then provide a starting point for more thorough requirements acquisition using other techniques. We then describe some steps that should be taken during this phase.

1. COTS candidates identification - identify products that could meet the evaluation criteria, several sources can be used for identifying products available in the market, such as: in-house reuse libraries, Internet, magazines, and Agora (Robert, Scott and Kurt 1998), a Web search engine to finding components in the software marketplace, this tool creates an indexed, worldwide database of software products classified by component type (e.g., JavaBeans or ActiveX control);

2. Product information/clarification - create a repository with relevant product information obtained above;

3. List of COTS candidates - generate a list of candidate COTS products.

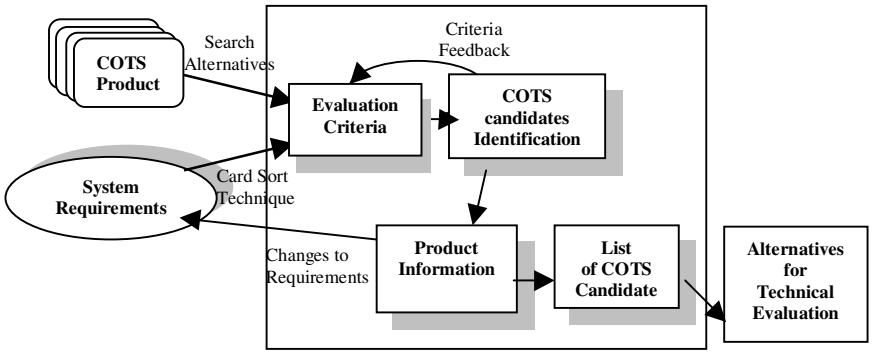

Fig. 5 – The Product Identification Phase

It is quite possible that among the COTS alternatives, some extra functionality (not initially considered) may be available. Some of these new requirements, upon a careful consideration, might indeed be required. This is an important feedback mechanism that can be used to enhance the development process and user satisfaction. Figure 5 describes this phase. It is important to note that there is not an optimal evaluation criteria. It will depends on each domain and environment.

The evaluation criteria should includes at least functional requirements. Although, in such cases the evaluation criteria often is not detailed enough to contribute as a basis for systematic technical evaluation. Furthermore, we observe that at this moment the requirements statements are still poorly described, specially non-functional requirements because they are usually very difficult to be quantified by customers. In this way, they need to be refined and formalized before initiating the technical evaluation. The following phase attempts to describe these requirements in adequate detail.

5.2 Requirements Acquisition Phase

Requirements acquisition must be an iterative and simultaneous process with product identification and product description, see figure 4. The product specification provided by the vendor usually gives a good description of product's functionality. In this way, they can be considered first during the evaluation process. However, quality attributes such as reliability, security and performance are important issues and are not described in details. Hence, it is necessary to use methods to clarify and refine these requirements. Indeed, we propose the use of the NFR Framework (Chung, Nixon, Yu and Mylopoulos 2000) to assist the decision making process. Figure 6 describes the activities addressed during the requirements acquisition phase. It starts with system requirements and COTS alternatives, then the NFR Framework is used in order to clarify non-functional requirements. Disputed requirements are negotiated and prioritized to identify critical requirements and to help the decision making process. Therefore, some products that do not meet the requirements are eliminated

from the candidate list. Finally, the result of this phase is an updated list of COTS candidates.

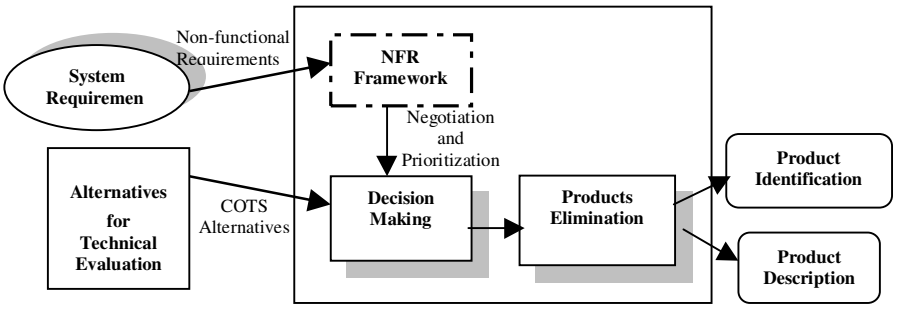

Fig. 6 – The Requirements Acquisition Phase

For the sake of illustration, consider a case study related to the development of an e-commerce application based on COTS products. Functional requirements include providing catalogue browsing facilities, search engines, on-line ordering, etc. Moreover, this application also needs to provide user-friendly access, security of information and good performance. After posing these non-functional requirement as goals to satisfice, the developer attempts to decompose them, as shown in figure 7 (Chung, Nixon, Yu and Mylopoulos 2000).

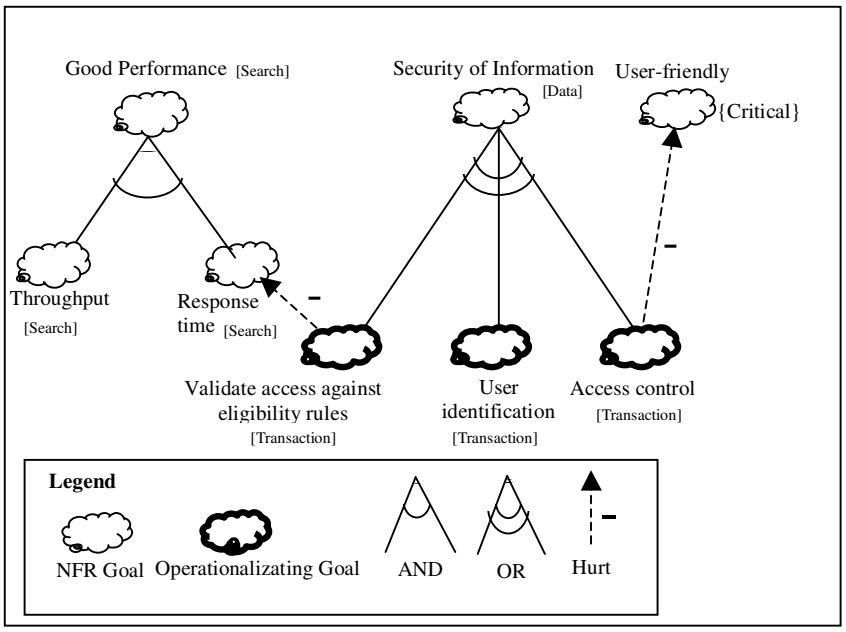

Fig. 7 – Decomposition of non-functional requirements using the NFR Framework

The refinement process continues until the developer considers that the possible solutions for the target system are sufficiently detailed, and that no other alternatives need be considered. The goal *security of information* is decomposed into the subgoals *validate access against eligibility*, *user identification* and *access control* through OR type of contribution (i.e. only one of these goals needs to be met for the overall *secure information* goal to be achieved). While the goal *good performance* is decomposed into *throughput* and *response time*. In this case, the contribution among the goals is of type AND (i.e. only if all subgoals are met the overall goal is achieved).

Interestingly, it is necessary to address interactions between different kinds of non-functional requirements even though the non-functional requirements were initially stated as separate requirements. Note that *access control* contributes negatively (show as "-") for *user friendly*. As *user friendly* is a critical goal, it is not adequate to operationalize *access control*. Similarly, *validate access against eligibility* rules makes a negative contribution towards the goal *time response*. Therefore, in the context, *user identification* is the best alternative to guarantee the security of the information. In this way, COTS components that support user identification are clearly preferred.

It is important to note that architectural and environmental aspects are fundamental when modelling non-functional requirements. For example, in order to asses the performance of a COTS product it is necessary to considering the architecture (Garlan 1995) and nature of input events in the environment.

	Product1	Product2	Product3	Product4
User Friendly	++	+	-	-
Good Performance	++	+	+	--
Secure Information *(user identification)*	+	++	-	

Legend: [++] strong positive satisficing [+] weak positive satisficing
[ ] lack of significant contribution [--] strong negative satisficing [-] weak negative satisficing

Table 1 – Evaluation of COTS products based on NFR Framework

The advantages of the NFR Framework are many folded: developers are able to explicitly and systematically express non-functional requirements, which in turn drives the COTS evaluation process. Indeed, after the refinement of non-functional requirements, the process of evaluation among COTS products become easier because the number of products that meet these refined requirements decreases. As shown before, the goal *user identification* is the most promising operationalization of the goal *secure information*. So, when evaluating COTS products against *security of information*, we should concentrate on products that support *user identification*.

Table 1 illustrates the compliance among the non-functional requirements of the domain application and four COTS candidates. The product 4 is not in accordance with any of the non-functional requirements and will be removed from the candidate list. The product 3 does not satisfice the requirement user friendly, which is a critical attribute then it will be also discarded. Note that products 1 and 2 will continue in the

list. Therefore, it is both necessary and cost-effective to select only the most promising candidates for detailed evaluation.

5.3 Product Description Phase

This phase provides a detailed description of all products that continue in the list of COTS candidates after the rejection of products during the NFR framework analysis. Besides the requirements specification used to evaluate COTS products, it is also necessary to consider non-technical issues for prioritizing candidates. Product specifications, documentation, briefings and demonstrations are used in order to better understand each product. We present the Description Checklist that guides the evaluation process. The developer should verify all these issues:

- Components' costs and benefits;
- Components' capabilities;
- Easiness of installation;
- Future standards;
- Vendor assessment;
- Quality and cost of support;
- Vendor reputation and maturity;
- Vendor infrastructure;
- Access to internal component information;
- Version choice and control;
- Risk analysis.

Throughout the analysis of the Description Checklist, a comparative evaluation is performed of the product description against the customer requirements. Therefore, the iteration between this phase and the requirements acquisition is performed again. It is important to note that during the selection process both functional requirements and non-functional requirements need to be equally considered. Although, as described above, non-functional requirements usually are more critical than functional ones.

As the COTS alternatives have been evaluated, the evaluation data needs to be used for making a decision. The AHP (Analytic Hierarchy Process) (Saaty 1990) is one technique to aid in the decision making process. This technique is based on the idea of decomposing a multiple criteria decision making problem into a criteria hierarchy. At each level in the hierarchy, the relative importance of factors is assessed by pair-wise comparisons. We apply this technique in the following fashion: the product's characteristics, such as functional requirements, quality characteristics and non-technical aspects are considered as criteria during the decision making process. Each criterion is assigned a weight or a score. In the case of weighting non-functional

requirements, we can use the estimation made during the application of the NFR framework where each requirement was described as: irrelevant, important, critical, etc. Finally, COTS alternatives are compared in pairs with respect to the criteria.

During the description phase, developers should also verify the quality of the components. We suggest a methodology described by Voas (1998) for determining the quality of COTS products. This approach does not require that a COTS software vendor disclose information concerning internal development process. However, it is necessary to obtain the product to perform it. To limit costs, customer may obtain a loaner copy, a demonstration copy or a copy borrowed from another organization (after arranging any required transfers license with the vendor).

These techniques are very useful during the evaluation process because they provide information about the component's impact on the system. For instance, customers obtain more resources to assess COTS products. The level of complexity regarding COTS products evaluation increases significantly when multiple products will have to be evaluated in combination. Carney and Wallnau (1998) provide additional discussions about evaluation of COTS-Intensive systems.

5.4 Product Acceptance Phase

The main objective of this process is to screening the products that were considered suitable to integrate the system after detailed analysis of non-functional requirements and other issues. Another activity is to negotiate the legal contract with vendors and resolve legal issues pertaining to the purchasing of the product and licensing. A license between the vendor and the customer should minimally specify:

- The rights the vendor authorizes the customer to exercise in the software;
- Who owns the component and modifications to the component;
- The risks and liability each party assumes under the license;
- Support, maintenance, and warranties for the component;

If the selected components do not provide all required functionality for typical systems in the domain, extra new components need to be developed to meet the necessities, but they may be specific to a particular system and not generally reusable.

6. Conclusions and Future Work

Component-Based Software Engineering is an important development approach for software systems. The promise of shorter development schedule, lower resource cost, and higher product quality has led to the increasing adoption of COTS. However, many new risks and problems have emerged, such as difficulty in meeting non-functional requirements, use of low-quality components, and insufficient technical support.

Indeed, the components community aims to make CBSE an effective and efficient practice. A first step to reach these objectives is develop a consistent and well-defined process model. The CRE Model presented here is our attempt to address this need. As far as we know, there are not any COTS selection models that treat exclusively functional requirements or non-functional requirements. Most papers only emphasizes the importance of acquiring user requirements. The CRE model identifies the important efforts required in evaluating and selecting COTS components. In addition, it provides a practical model for acquiring and dealing with user non-functional requirements.

As observed above, the processes of non-functional requirements prioritization and decision making are very complex. Therefore, further work on the CRE Model includes the analysis of interdependencies between non-functional requirements and the improvement of rationale behind COTS choice.

Furthermore, we intend to apply the CRE Model in a real situation, for that we are currently undertaking case studies. Another future work is to extend this model through the phases of adaptation and integration of COTS products. Finally, we believe that requirements engineering provides significant implications for CBSE practice and research.

References

Aoyama, M.: New Age of Software Development: How Component-Based Software Engineering Changes the Way of Software Development. International Workshop on Component-Based Software Engineering, April (1998)

Bergner, K., Rausch, A., Sihling, M.: Componentware--The Big Picture. International Workshop on Componet-Based Software Engineering, April (1998)

Brown, A.W., Wallnau, K.C.: Engineering of Component-Based Systems, Component-Based Software Engineering. Selected Papers from the Software Engineering Institute, IEEE Computer Society Press (1996)

Chung, L, Nixon, B., Yu, E.: Dealing with Change: An Approach using Non-Functional Requirements. Requirements Engineering, Vol. 1, No. 4, 1996, pp. 238-260 (printed 1997).

Chung, L., Nixon, B., Yu, E., Mylopoulos, J.: Non-Functional Requirements in Software Engineering. Kluwer Academic Publisher (2000)

Chung, L., Nixon, B.: Dealing with Non-Functional Requirements: Three Experimental Studies of a Process-Oriented Approach. 17th International Conference on Software Engineering, April (1995)

Chung, L., Nixon, B.A., Yu, E.: Using Non-Functional Requirements to Systematically Select Among Alternatives in Architectural Design. Proceedings of 17th ICSE Workshop on Architectures for Software Systems, Seattle (1995)

D'Souza, D.F., Wills, A.C.: Objects, Components and Frameworks with UML: The Catalysis Approach. Addison Wesley (1998)

Fox, G., Marcom, S., Lantner, K.: A Software Development Process for COTS-based Information System Infrastructure. Proceedings of the Fifth International Symposium on Assessment of Software Tools - SAST'97. June (1997)

Garlan, D. R., Ockerbloom, J.: Architectural Mismatch: Why its Hard to Build Systems Out of Existing Parts. Proc. Of the International Conference on Software Engineering, April (1995)

Garlan, D., Allen, R., Ockerbloom, J.: Architectural Mismatch: or Why It's Hard to Build Systems Out of Existing Parts. Proceedings of the 17th International Conference on Software Engineering, Seattle (1995)

Lichota, R.W., Vesprini, R.L., Sawanson, B.: PRISM Product Examination Process for Component Based Development. Proceedings of the Fifth International Symposium on Assessment of Software Tools - SAST'97. June (1997)

MacGraw, G., Viega, J.: Why COTS Software Increases Security Risks. Reliable Software Technologies. Sterling VA USA. June (1998)

Maiden, A.N., Ncube, C.: Acquiring COTS Software Selection Requirements. IEEE Software, Vol 15 n°1-3, April (1998)

Ncube, C., Maiden, N.A.M.: PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm. International Workshop on Component-Based Software Engineering, May (1999)

Nwosu, C.K., Seacord, R.C.: Workshop on Component-Based Software Engineering Processes. held in conjunction with Technology of Object-Oriented Languages and Systems Tools Conference, August (1999)

Robert, C.S., Scott, A.H., Kurt, C.W.: AGORA: A Search Engine for Software Components. Software Engineering Institute, Carnagie Mellon University, USA (1998)

Saaty, T. L.: The Analytic Hierarchy Process, New York: McGraw-Hill (1990)

Seacord, R., Nwosu, K.: Life Cycle Activity Areas for Component-Based Software Engineering Processes. Workshop on Component-based Software, August (1999)

Sommerville, I.: Software Engineering. 5 th Edition ed: Addison-Wesley (1996)

Szyperski, C.: Component Software. Addison-Wesley (1998)

Tran, V., Liu, D.: A Procurement-centric Model for Engineering Component-based Software Systems. Proceedings of the Fifth International Symposium on Assessment of Software Tools - SAST'97. June (1997)

Voas, J. M.: Certifying Off-the-Shelfs Software Componentes. IEEE Computer Society. June (1998)

Wallnau, K.C., Clements, P., Zaremski, A.: Correcting, Identifying, and Avoiding Interface Mismatch: Theory and Practice. Draft Paper, Software Engineering Institute, Carnegie Mellon University (1997)

Wallnau, K.C., Carney, D., Pollak, B.: How COTS Software Affects the Design of COTS-Intensive Systems. Software Engineering Institute, Carnegie Mellon University, USA. June (1998)