Domain-Based COTS-Product Selection Method*

Hareton K.N. Leung[1] and Karl R.P.H. Leung[2]

[1] Laboratory for Software Development & Management
Dept. of Computing. Hong Kong Polytechnic University, Hong Kong
cshleung@comp.polyu.edu.hk
[2] Compuware Software Testing Laboratory
Dept. of Information & Communications Technology
Hong Kong Institute of Vocational Education (Tsing Yi), Hong Kong
kleung@computer.org

Abstract. Use of commercial-off-the-shelf (COTS) products is becoming an acceptable software development method. Current methods of selecting COTS products involve using the intuition of software developers or a direct assessment of the products. The former approach is subjective, whereas the latter approach is expensive as the efficiency of the direct assessment approach is inversely proportional to the product of the number of modules in the system to be developed and the total number of modules in the candidate COTS products. With the increase in the number of available COTS components, the time spent on choosing the appropriate COTS products could easily offset the advantages of using them. A domain model is a generic model of the domain of an application system. It captures all of the features and characteristics of the domain. In this chapter, we present a new indirect selection approach, called the Domain-Based COTS-product Selection Method, which makes use of domain models. We also report a successful case study in which we applied our selection method to the development of an on-line margin-trading application.

1 Introduction

The use of commercial-off-the-shelf (COTS) products as units of large systems is becoming popular. Shrinking budgets, the rapid advancement of COTS development and the increasing demands of large systems are all driving the adoption of the COTS development approach. A COTS component is defined as an independent unit that provides a set of related functions and which is suitable for reuse [8]. COTS components are different from software components in terms of their completeness [4]. Those systems that adopt COTS development as much

* This work was partially supported by the Hong Kong Polytechnic University research grant AP205, and also by two grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Nos: CityU 1118/99E and HKU 7021/00E).

A. Cechich et al. (Eds.): Component-Based Software Quality, LNCS 2693, pp. 40–63, 2003.

as possible are called *COTS-Based systems (CBS)*. Compared with traditional software development, CBS development promises faster delivery with lower resource costs. The shift from custom development to CBS is not limited to new application development projects. Many maintenance projects also involve COTS products. Rather than building the whole system from scratch, a new system can be assembled and constructed by using existing, market proven and vendor supported COTS products. For example, COTS components for inventory control and accounts receivables can be purchased and integrated into an accounting system. The use of COTS products has become an economic necessity [3, 15].

Developing a CBS involves selecting the appropriate COTS products, building extensions to satisfy specific requirements, and then *gluing* the COTS products and other units together. The success of CBS development depends heavily on the ability to select the *appropriate* COTS components. An inappropriate COTS product selection strategy can lead to adverse effects. It could result in a shortlist of COTS products that can hardly fulfill the required functionality, and it might also introduce overheads in system integration and maintenance phases of a project. An effective and efficient COTS product selection process is essential to the delivery of the full potential of CBS development. If the effort required in selecting the appropriate COTS product is too high, then it may offset the time saving in using the COTS development approach.

We classify the current COTS product selection methods into three categories, namely, the *intuition* approach, the *direct assessment* approach and the *indirect assessment* approach. In the intuition approach, software developers select COTS products according to their experience and intuition. This approach is subjective, and some COTS products that are qualified candidates for an application may be omitted inadvertently.

Most of the recently proposed COTS component selection methods belong to the *direct assessment (DA)* approach, which selects COTS components directly from their source. These methods consider ALL of the descriptions of the COTS products and then try to make decisions on their suitability. For example, a recent study proposes the use of a matching process between the system requirements and the constraints imposed by the COTS products [13]. These methods all require a searching phase followed by an evaluating phase that examines both the functional and non-functional aspects of the COTS products. These approaches are more objective than the intuition-based approaches. However, the efficiency of direct assessment approach is inversely proportional to the product of the number of modules[1] in the system to be developed and the total number of modules in the candidate COTS products. The cost of selecting an appropriate COTS product can be expensive and hence, may offset the advantages of using COTS. Furthermore, we noticed that the vendor's information has not been well utilized. By making better use of the vendor's information, we can reduce the time required for selecting COTS products.

We have developed an indirect method that does not directly compare the modules of the system to be developed with the COTS product during the selec-

[1] The term "module" in this chapter refers to a unit of a system.

tion process. The *Domain-Based COTS-product Selection method* [11], instead of assessing all of the available COTS products, makes use of the specific domain model of the intended system to decide the suitability of the COTS product. The intention of this approach is to reduce the amount of work required for the COTS selection.

Domain modelling has been recognized as an effective tool in developing quality software. A *domain model* is a generic model of the domain in question. In other words, a domain model is the meta-model of the domain. It models a domain by capturing all the intrinsic of the domain and it also captures all the relations between these intrinsic. Hence, a domain model captures all of the properties, characteristics, functions and features of the domain. Then an application system of the domain is a refinement of the domain model. Domain models can be expressed in any appropriate software specification languages, like OMT, depending on the characteristics of the domain.

There are two basic strategies for selecting a COTS product, depending on whether an application development needs the best available COTS product:

Best-fit strategy: the selection process is aimed at identifying the best COTS product among all of the candidates.

First-fit strategy: the selection process is aimed at identifying the first COTS product that satisfies all of the requirements. If no COTS product satisfies all of the requirements, then the best product from the available COTS products is selected.

In general, the best-fit strategy will require an analysis of all of the COTS candidates, whereas the first-fit strategy may require less effort since it will stop once the first COTS candidate that meets the requirements has been identified. However, the latter strategy may not identify the best COTS product.

In Section 2 of this chapter, we provide a review of the COTS product selection methods. Section 3 presents the relationships among the domain model, COTS products and CBS. Based on the insights from these relationships, we have developed a new COTS product selection method, which is called the Domain-Based COTS-product Selection (DBCS) method. An overview of the DBCS method is given in Section 4, and detailed procedures in Section 5. In Section 6, we first analyze the efficiency of DBCS in Section 6.1. We have successfully applied the DBCS method to the development of an on-line margin-trading system for a local bank. This case study is presented in Section 6.2. In Section 7, we present our conclusions.

2 COTS Selection Methods

We first give an overview of some direct assessment methods that have been proposed for COTS product selection. They are the *Off-The-Shelf-Option* (OTSO) [7], *COTS-based Integrated System Development* (CISD) [16], *Procurement-Oriented Requirements Engineering* (PORE) [12], *COTS-based Requirements Engineering* (CRE) [1], and the *Infrastructure Incremental Development Approach* (IIDA) [6].

2.1 OTSO

Kontio proposed the *Off-The-Shelf-Option* (OTSO) selection method [8, 7]. The OTSO method assumes that the requirements of the proposed system already exist. However, in practice, the requirements cannot be defined precisely because the use of certain COTS products may require some changes to the requirements. The main principle followed by the OTSO method is that of providing explicit definitions of the tasks in the selection process, including entry and exit criteria. It also uses the Analytic Hierarchy Process as a decision-making method to analyze and summarize the evaluation results.

The inputs for OTSO include the requirements specification, design specification, project plans and organizational characteristics. The outputs are the selected COTS product, the results of the evaluation, and cost models.

The OTSO selection method comprises three phases: *searching, screening and evaluation*. Based on knowledge of the requirements specification, design specification, project plan and organizational characteristics, a set of selection criteria is set up for selecting the COTS products. The searching phase attempts to identify all potential COTS candidates that cover most of the required functionality. This phase emphasizes *breadth* rather than *depth*. The search criteria are based on the functionality and the constraints of the system.

The objective of the screening phase is to decide which COTS candidates should be selected for detailed evaluation. The screening criteria are similar to those of the searching phase. The less-qualified COTS candidates are eliminated during this stage.

In the evaluation phase, COTS candidates undergo a detailed evaluation. The evaluation criteria are defined by decomposing the requirements for the COTS products into a hierarchical set of criteria. Each branch of this hierarchy ends in an *evaluation attribute*, which is a well-defined measurement that will be determined during the evaluation. Although the set of criteria is specific to each case of COTS products selection, most of the criteria can be categorized into four groups:

- functional requirements of the COTS;
- required quality-related characteristics, such as reliability, maintainability and portability;
- business concerns, such as cost, reliability of the vendor, and future development prospects;
- issues relevant to the software architecture, such as constraints presented by an operating system, the division of functionality in the system, or the specific communication mechanisms that are used between modules.

The OTSO method uses the Analytic Hierarchy Process (AHP) to consolidate the evaluation data for decision-making purposes [14]. The AHP is based on the idea of decomposing a complex, multi-criteria decision-making problem into a hierarchy of the selection criteria. The AHP helps decision makers to structure the important components of a problem into a hierarchical structure. The AHP

reduces the complex multi-criteria trade-off decisions to a series of simple pairwise comparisons and synthesizes the results. The AHP not only helps a decision-maker to arrive at the best decision, but also provides a clear rationale for making a particular choice.

A limitation of OTSO is that some of the selection criteria may not have been clearly defined.

2.2 CISD

Tran and Liu have proposed the *COTS-based Integrated System Development* (CISD) model for COTS development [16]. The CISD model is not solely for COTS product selection. It is a model that can be used to generalize the key engineering phases of selecting, evaluating and integrating COTS products for a CBS.

The inputs for the selection process are the system requirements and information about the COTS products, and the outputs include a prioritized list of the COTS products and the architecture of the system.

The CISD model consists of three distinct phases: *identification, evaluation* and *integration/enhancement*. The COTS product selection process lies within the identification and evaluation phases.

The identification phase includes all of the technical activities that are required to generate a prioritized collection of products for subsequent evaluation. It includes two sub-phases: *product classification* and *product prioritization*. In the product classification sub-phase, information on potential COTS products is collected based on the requirements of each (application) service domain. This phase is similar to the searching phase of the OTSO selection process.

In the prioritization sub-phase, candidate COTS products are screened and prioritized. Two important criteria for prioritization are interoperability and the ability to fulfill multiple requirements of the service domains. The first criterion ensures that the selected COTS candidates can be integrated readily, leading to a reduction in the overall time and effort required during system integration. The second criterion gives a higher rating to products that support multiple domains because these products can reduce the number of interconnected interfaces and architectural mismatches.

The *evaluation phase* encompasses the process of creating prototype software and temporary integration and testing of the candidate COTS products. A detailed evaluation of the COTS products is performed. Three important attributes of the COTS products are examined:

- Functionality
- Architecture and interoperability
- Performance

The *integration/enhancement* phase encompasses all of the development efforts that are required to interconnect the different selected COTS products into a single integrated system. The key advantage of the CISD model comes from its integration of the strengths of the Waterfall and Spiral models.

2.3 PORE

The *Procurement-Oriented Requirements Engineering* (PORE) method guides a software development team in acquiring customer requirements and selecting COTS products that satisfy those requirements [12]. It uses a *progressive filtering* strategy, whereby COTS products are initially selected from a set of potential candidates, and then progressively eliminated when they do not satisfy the evaluation criteria [9].

The PORE method supports an iterative process of acquiring requirements and selecting COTS products. During each of the iterations, the software development team acquires information about the customer requirements that help discriminate between the COTS product candidates. The team also undertakes multi-criteria decision-making to identify candidates that are not compliant with the requirements. Therefore, the team rejects some of the COTS candidates and then explores the remaining COTS candidates by discovering possibly new customer requirements that might discriminate more thoroughly between the remaining candidates.

The inputs of the PORE method are the attributes of the COTS products, supplier requirements, information about product branding, open standards, product certification, the development process, the supplier's CMM level [5], the product and the supplier's past record, reliability, security, and dependability. The output is the shortlist of COTS products.

PORE offers techniques such as scenarios to discover, acquire and structure the customer requirements and formulate test cases that are used to check the compliance of the COTS products with the customer requirements.

2.4 CRE

The *COTS-based Requirements Engineering* (CRE) method was developed to facilitate a systematic, repeatable and requirements-driven COTS product selection process. A key issue that is supported by this method is that of the definition and analysis of the non-functional requirements during the COTS product evaluation and selection [1].

The CRE method is goal oriented in that each phase is aimed at achieving a predefined set of goals. Each phase has a template that includes some guidelines and techniques for acquiring and modeling requirements and evaluating products.

The inputs of the CRE method include defined goals, evaluation criteria, information about the COTS candidates and test guides. The output is the selected COTS products.

This method has four iterative phases: *identification, description, evaluation* and *acceptance*. The identification phase is based on a careful analysis of influencing factors, which come from the classification proposed by Kontio [8]. There are five groups of factors that influence the selection of COTS products: *user requirements, application architecture, project objectives & restrictions, product availability* and *organizational infrastructure*.

During the description phase, the evaluation criteria are elaborated in detail with an emphasis on the non-functional requirements. This is followed by a refinement of the description of the requirements.

In the evaluation phase, the decision to select a particular COTS product is based on the estimated cost versus an analysis of the benefits. The cost model that is used is called COCOTS (Constructive COTS) [2]. In particular, the *best* COTS product is identified by continuously rejecting non-compliant candidates. COTS products that do not meet the requirements are rejected and removed from the list of candidates.

The acceptance phase is concerned with the negotiation of a legal contract with vendors of the COTS products. During this phase, the evaluation team has to resolve legal issues pertaining to the purchase of the products and their licensing.

The selection criteria of the CRE method include:

- Functional and non-functional requirements. The selected COTS products have to provide all of the required capabilities, which are necessary to meet essential customer requirements. Among these requirements, the non-functional requirements play a critical role during the assessment process.
- Time restrictions. The time available for searching and screening all of the potential COTS candidates.
- Cost rating. The cost of acquiring the COTS products. This includes expenses such as acquiring a license, the cost of support, expenses associated with adapting the product, and on-going maintenance costs.
- Vendor guarantees. This addresses the issue of the technical support that is provided by a vendor. Consideration is given to the vendor's reputation and the maturity of their organization, and the number and kinds of applications that already use the COTS product.

A disadvantage of the CRE method is that the decision-making process can be very complex, given that there are a large number of potential COTS products and many evaluative criteria.

2.5 IIDA

Fox has proposed the *Infrastructure Incremental Development Approach* (IIDA) for the development of technical infrastructure using COTS products [6]. This approach is a combination of the classical waterfall and spiral development models in order to accommodate the needs of CBS development. The process of selecting COTS products in the IIDA relies on two phases: *analysis prototype* and *design prototype*.

In the analysis-prototype phase, COTS candidates are selected from each COTS *product family*. A COTS product family is defined as a group of COTS products that perform similar functions and/or provide related services. Based on the general capabilities and basic functionalities of the COTS candidates, the qualified COTS products that fulfill the infrastructure requirement are identified

in each of the COTS product families. However, the IIDA does not specify how the COTS product family is constructed or provide possible sources for that information.

The purpose of the design prototype phase is to select and evaluate the best COTS products from the earlier phase. Basic evaluation criteria include functionality and performance.

2.6 Summary of Direct Assessment Methods

Table 1 summarizes the inputs, selection procedures, selection criteria and outputs of the COTS product selection methods.

Although the five direct assessment methods (mentioned above) have some differences in their finer details, the typical steps of these methods are as follows:

1. Inspect all of the modules of each of the available COTS products to check whether they have modules that satisfy some or all of the functional requirements of the CBS being developed.
2. Check whether a COTS product also satisfies the non-functional requirements of the CBS. Non-functional requirements may include properties such as the interoperability of the modules of the COTS product with other systems.
3. Select the most appropriate COTS product that satisfies both the functional and non-functional requirements of the CBS.

The efficiency of these exhaustive direct assessment methods is inversely proportional to the product of

- the number of modules to be developed using COTS products, and
- the total number of modules in all of the available COTS products.

As more and more COTS products become available in the market, the total number of modules in all of the available COTS products will become large. Therefore, the efficiency of the direct assessment methods will decrease sharply.

3 Relationships among the Domain Model, COTS Products and CBS

The domain-based COTS-product selection method is founded on the relationships among the domain model, COTS products and CBS. In this section, we present an analysis of these relationships.

3.1 Relationships between COTS Products and a CBS

Many of the COTS products in the market are generic. The vendors of these products claim that they are applicable to systems in different application domains. A CBS development requires that we examine COTS products from multiple vendors. The selection of COTS components for an individual CBS is then a

Table 1. Comparing the direct assessment methods

	OTSO	CISD	PORE	CRE	IIDA
Input	– Requirement Specifications – Design Specifications – Project plans	– System Requirements – COTS Products	– COTS Product attributes – Supplier Requirements – Product development process	– Defined goals – Evaluation criteria – COTS Products – Test guides	– COTS from each COTS product family
Selection Procedure	– Searching – Screening – Evaluation	– Product Identification: * Classification * Priorization – Evaluation	– Acquires customer requirements – Multi-criteria decision making – Rejects non-compliant COTS – Explores other candidates for new customer reqs.	– Identification – Description – Evaluation – Acceptance	– Analysis prototype – Design prototype
Selection criteria	– Functional requirements – Quality characteristics – Business concerns – Relevant software architecture	– Functionality – Architecture/ interoperability – Performance – Fulfil multiple requirements from the service domain	– Development process – Supplier CMM level – Product/supplier past record – Reliability – Security – Dependability	– Functional and non-functional requirements – Time restrictions – Cost rating – Vendor guaranties	– Functional requirements – Performance
Output	– Selected COTS – Evaluation results – Cost models	– Prioritized COTS products – System Architecture	– Selected COTS	– Selected COTS	– Selected COTS

Fig. 1. Many-to-many Relation between COTS and CBS

Fig. 2. Relation between a Domain Model and a CBS

many-to-many matching process (Fig. 1). The efficiency is inversely proportional to the product of the number of modules to be developed using the COTS products and the total number of modules being considered in the COTS products. If the CBS is a complex system and there are many COTS products available on the market, the system developers will have to examine many COTS products, and hence they will exert a great effort in making the selection. Consequently, this many-to-many relation may offset the advantages of using COTS development.

With the increase in the number of available COTS products and the many-to-many relation between COTS and CBS, it is difficult, without a good selection method, for system developers to examine all of the COTS products. Also, searching through a large number of available COTS products is error-prone. Ideally, the selection method should be automated to increase its efficiency and reduce errors.

3.2 Relationship between a Domain Model and a CBS

A domain model is a generic model of a domain. There may be an infinite number of systems for a domain. Hence, the relation between a domain model and an individual CBS is one-to-many. However, the relation between a domain model and an individual CBS is a simple one-to-one relation Fig. 2 [2]. This is the case because each module of the CBS should have corresponding modules in the domain model. Otherwise, the domain model is not an appropriate generic model of the domain. It is a simple relation because it is easy to identify the modules in the domain model that correspond to the CBS.

3.3 Relationship between COTS Products and a Domain Model

In general, a domain model can be supported by more than one COTS product, or the modules of a domain model can be fitted by a number of modules,

[2] The parallelogram represents a specific entity, which denotes a domain model and a CBS in this example. Rectangles represent a class of entities, such as multiple CBS and COTS products in Fig. 1.

Fig. 3. Relation between Class of COTS and a Domain Model

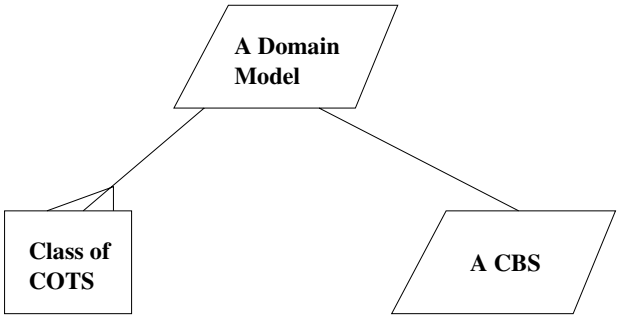

Fig. 4. Relation between COTS, a Domain Model and a CBS

each from different COTS products. Consequently, the relation between COTS products and a domain model is many-to-one (Fig. 3).

It should be noted that the vendors of COTS products have all the information about the COTS products, including the functional and non-functional applicability of the product. Furthermore, it is a common marketing strategy that products are focused to some specific application domains instead of any domain in general. Consequently, the COTS vendors should be familiar with their target domains. Hence, it would be efficient and appropriate for the vendors to map their COTS products to the domains in which their COTS products apply. They could specify how the COTS products can be used in the applicable parts of a domain. This would not only help the software developers to select the right COTS products, but also help the vendors to market their products.

4 Domain-Based COTS-Product Selection Method

From the above analysis of the relationships between the class of COTS products, a domain model and a CBS, we notice that the relation between the class of COTS products and a domain model is a many-to-one relation and the relation between a domain model and a CBS is a one-to-one relation (Fig. 4). We can take advantages of the properties of these relations and design a new approach to select COTS products. We use the domain model as an agent between the COTS products and a CBS. The modules from the COTS products that are claimed by the vendor to be appropriate for specific modules of a domain model are first mapped to these modules by the vendors. Then, when selecting the COTS modules for a CBS, instead of selecting the COTS components directly,

the corresponding modules in the domain model of a CBS are consulted first. Through the mappings between a domain model and the COTS products, the corresponding COTS modules that are (claimed to be) appropriate for the CBS module are identified. A vetting procedure is then used to select the best COTS products for the application. As the mappings between the COTS products and the domain models have been provided in terms of information on how the products can be applied to a domain, selecting an applicable COTS product can be automated based on the mappings. However, the vetting procedure cannot be fully automated because some of the functional and non-functional criteria will require human judgment.

To summarize, this method consists of two phases: set-up and selection.

1. **Set-up phase**
   When vendors rollout their COTS products, besides making the COTS products available on the market, they also need to map their COTS modules to those modules of the domains that they find are applicable. These mappings are available to the application system developers and can be accessed electronically.
2. **Selection phase**
   (a) The corresponding modules in a domain model are identified for each of the modules of the CBS in question.
   (b) The COTS modules that claim to be applicable are identified by the mappings from the domain model to the COTS modules. It should be noticed that these modules are functionally applicable to the domain models.
   (c) The non-functional properties of the identified COTS modules are assessed.
   (d) With reference to all of the assessment results, the most appropriate COTS modules are selected.

We illustrate this method with the example shown in Fig. 5. In the set-up phase, the vendor of COTS1 has identified that C11 is applicable to n1 of the domain model; the vendor of COTS2 has identified that C21 and C22 are applicable to n1 and n2, respectively; the vendor of COTS3 has identified that C31 is applicable to n2; and the vendor of COTS4 has identified that C41 and C42 are applicable to n2 and n3, respectively.

Then, when the specification of the CBS is developed, it is easy for the developers to identify that S1 corresponds to n2 and S2 corresponds to n3 of the domain model. With reference to n2 and n3, COTS modules C22, C31, C41 and C42 are identified. The next step is to validate the non-functional requirements of these four modules. One module has to be chosen from C22, C31 and C41. Since C42 is the only applicable module for S2, it will be selected if it satisfies the non-functional requirements.

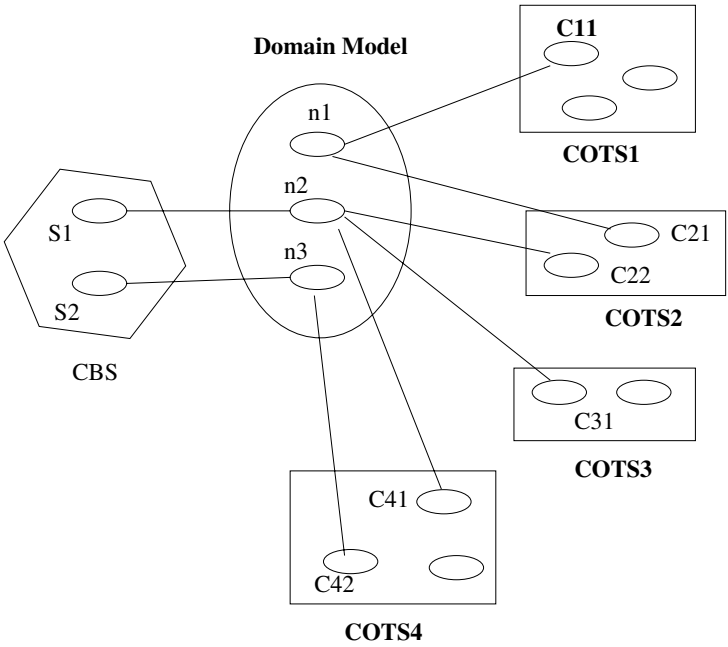

Fig. 5. An example of using the DBCS method

5 Steps of the Domain-Based COTS-Product Selection Method

In this section, we present two procedures for the domain-based COTS-product selection method. These procedures correspond to the best-fit strategy and first-fit strategy.

Before we continue our discussion, we first define some auxiliary operators. Let $elem(l)$ be a function that returns the set of unique elements from a list, l. For example, $elem([1,1,2]) = [1,2]$. Let $card(S)$ be a function that returns the cardinality of a set, S. For example, $card(1,2,3) = 3$. Let $xtp(i,t)$ be a function that extracts the i^{th} element from a set of tuples, which is denoted by t, and let it return the elements from the tuples in a set. If the i^{th} element in a tuple does not exist, xtp returns an empty set. For example, $xtp(2,(a,b),(c,d,e)) = b,d$ and $xtp(3,(a,b),(c,d)) = \{\}$.

5.1 Definitions

Let there be n modules in the CBS ($n \geq 0$) and x COTS products available ($x \geq 0$). Let c_i denote an individual COTS product. Let cm_i be the set of modules of c_i. Then, (c_i, cm_i) form a tuple that captures the COTS product identifier and its set of modules. Let $L_c = [(c_1, cm_1), (c_2, cm_2), \cdots, (c_x, cm_x)]$ be the list that represents a set of x COTS products.

Let there be y COTS products among the x available COTS products that have some modules satisfying some of the functional requirements of the n CBS modules.

Let each of these y COTS products be denoted by d_i. Let dc_i be a module of d_i, and let n_i be a module of the CBS that is functionally satisfied by dc_i. Let dm_i be a set of tuples (dc_i, n_i). Then, a list, $L_d = [(d_1, dm_1), (d_2, dm_2), \cdots, (d_y, dm_y)]$, can be constructed to keep the COTS candidates together with their set of module pairs that satisfy the functional requirements of the CBS.

Since L_d is selected from L_c, $card(elem(L_c)) = x$, and $card(elem(L_d)) = y$, we have

1. $x \geq y$,
2. $xtp(1, elem(L_c)) \supseteq (xtp(1, elem(L_d))$,
3. $\forall c_i, \exists cm_i, dm_i : (ci, cmi) \in elem(L_c) \wedge (c_i, dm_i) \in elem(L_d) \Rightarrow cm_i \supseteq xtp(1, dm_i)$.

Let the CBS have v non-functional criteria. Among the y COTS products, let there be z COTS products that satisfy the non-functional requirements of the CBS.

Let each of these z COTS products be denoted by e_i. Let ec_i be a module of e_i, which satisfies the non-functional requirements, and let n_i be a module of the CBS, which is functionally satisfied by ec_i. Let em_i be the set of tuples (ec_i, n_i).

For functional requirements, if we express the statements of the requirements in conjunction normal form, each of the requirement clauses is then either satisfied or unsatisfied. However non-functional requirements, unlike functional requirements, can have partial or various degrees of satisfaction. For example, the performance of COTS A may just meet the requirements and cost \$A, while COTS B gives a better performance than A and costs less than \$A. Then, we can say that the overall rating on the non-functional requirements for B is greater than A. This assignment of the overall rating depends on a wide variety of factors such as company policy, experience, costing and the specific CBS. Therefore, its evaluation requires human judgement and it cannot be automated. Let p_i be the overall assessment rating of the non-functional requirements of the COTS product e_i. Then, a list, $L_e = [(e_1, em_1, p_1), (e_2, em_2, p_2), \cdots, (e_z, em_z, p_z)]$, can be constructed to keep the COTS products together with their set of modules that satisfy both the functional and non-functional requirements of the CBS and their overall rating on the non-functional requirements.

Since L_e is constructed from L_d and $card(elem(L_e)) = z$, we have

1. $y \geq z$,
2. $xtp(1, elem(L_d)) \supseteq xtp(1, elem(L_e))$,
3. $\forall e_i, \exists em_i, p_i : (e_i, em_i, p_i) \in elem(L_e) \Rightarrow (e_i, em_i) \in elem(L_d)$.

Fig. 6 shows the relationship between the three types of COTS products (x COTS, y COTS and z COTS) and the CBS.

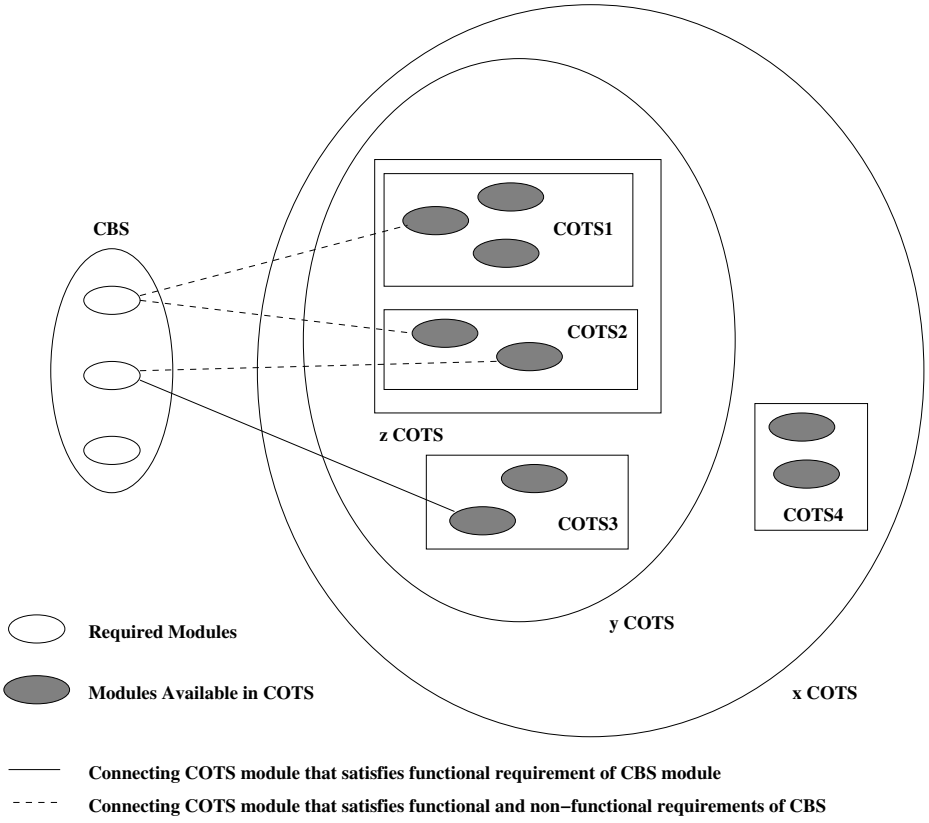

Fig. 6. CBS and related COTS

5.2 Domain-Based COTS-Product Selection Using a Best-Fit Strategy

When selecting a COTS product using the DBCS method and the best-fit strategy, the following five steps are involved.

Step 1: *The modules that correspond to the CBS are identified from a domain model.*
The developers should be familiar with the system in question and also the corresponding domain model. Then, it is easy for the developers to build the mapping between the modules of the CBS and the corresponding domain model.

Step 2: *Identifiers of the COTS modules, which are mapped to the modules in the domain model from Step 1, are found through the mappings between the domain model and the COTS modules.*
These COTS modules are claimed (by the vendor) to functionally satisfy the modules of the domain model. Since the relation between an individual

COTS module and a specific domain model is a one-to-one mapping, the time required to search for the corresponding modules in the domain model during this step is directly proportional to the number of modules of the domain model.

If no mappings from COTS modules to a domain model are found, this means that no COTS module is applicable to that domain. Hence, the selection of COTS modules is finished with the result being no applicable COTS modules.

Step 3: *Information is collected on those COTS products that have modules selected during step 2.*

This step is used to identify the COTS products that are claimed to satisfy the functional requirements. There is no need to retrieve information about the COTS products during this step. Only the identifiers of the COTS products are required. This step is performed by consulting the mappings between the domain model and the sources of the COTS products. These mappings are provided by the vendors of the COTS products in advance, and they can be stored in computer systems or obtained through the Internet. Hence, this step is similar to retrieving information from databases.

Let the n modules in the domain model be denoted by $(n_1, n_2, \cdots, n_n)$. Let l_i denote the set of identifiers of the COTS modules that satisfy the functionality required of module n_i. Then, the list $L_n = [(n_1, l_1), (n_2, l_2), \cdots, (n_n, l_n)]$ denotes the list of modules in the domain model together with the set of identifiers of the modules of the COTS products. In the example shown in Fig. 5, there are three modules in the domain model, four COTS products available, and a total of six matching pairs of modules, so that $L_n = [(n1, c11, c21), (n2, c22, c31, c41), (n3, c42)]$. The time required to construct L_n can be seconds or minutes.

Step 4: *The COTS products are vetted against the non-functional requirements of the CBS.*

This step involves constructing $L_d^{'} = [(d_1, dm_1), (d_2, dm_2), \cdots, (d_{y'}, dm_{y'})]$ from L_n. This includes two sub-steps: functional vetting and non-functional vetting.

- Functional vetting involves constructing the list of pairs given by $L_d'' = [(d_1, ldm_1), (d_2, ldm_2), \cdots, (d_y', ldm_y')]$ from L_n, where d_i is the i^{th} COTS product and ldm_i is the set of identifiers of the modules of the i^{th} COTS product that contains the functionally appropriate modules for the CBS.
- Non-functional vetting involves retrieving information about the COTS modules to form the list L_d' and checking that the non-functional requirements are met. In the example shown in Fig. 5,

$$\begin{aligned} L_d' = [\ & (COTS1, (C11, n1)), \\ & (COTS2, (C21, n1), (C22, n2)), \\ & (COTS3, (C31, n2)), \\ & (COTS4, (C41, n2), (C42, n3))] \end{aligned}$$

Step 5: *The best COTS product is selected from the candidates that satisfy both functional and non-functional requirements.*

5.3 Domain-Based COTS-Product Selection Using a First-Fit Strategy

In applying the first-fit strategy, ideally, once we find a COTS product that fits all of the functional and non-functional requirements, we can stop the selection process. However, a suitable COTS product may not exist. If no COTS product satisfies all of the functional and non-functional requirements, then the one with the most modules that satisfy both the functional and the non-functional requirements is chosen.

The steps of the first-fit strategy of the DBCS method (DBCS-FF) are as follows:

```
0.  Identify the corresponding modules of the CBS in the domain model.
1.  while (there are unselected COTS products) or
            (no appropriate COTS product has been found) {
2.      Choose a COTS product;
3.      if (not all n modules in the domain model have
                mappings from this COTS product) {
4.          Insert the COTS product into the working list
                L_w = [(d_1, dm_1), ..., (d_u, dm_u)];
5.      } else {
            Get information about the COTS products from the sources.
6.          if (it satisfies the non-functional requirements) {
                An appropriate COTS product has been found;
            }
        }
7.  } // end while
8.  if (no appropriate COTS product has been found) and
            (L_w is not empty) {
9.      Sort the list L_w in descending order according to card(dm_i).
10.     while (there are COTS products in the sorted list) and
                (no suitable COTS product has been found) {
11.         Get the head of the sorted list;
12.         Retrieve information on the COTS product from the source;
13.         Check the non-functional requirements of the COTS product.
14.         if (it satisfies the non-functional requirements) {
                An appropriate COTS product has been found;
            }
15.     } // end while
16. } // end if
```

In using the DBCS-FF, since there are mappings between the COTS products and the domain model, all of the COTS modules that are functionally appropriate for the CBS can be obtained directly through these mappings. The DBCS-FF tries to find the first COTS product that functionally satisfies most

of the modules of the CBS and also the non-functional requirements. The second half of DBCS-FF is used to handle the situation when no COTS product meets all of the functional requirements of the CBS, and the suitability of these partially matched COTS products are then accessed.

6 Discussion

Section 6.1 discusses the efficiency of DBCS while Section 6.2 substantiates the performance study by a case study of an on-line margin-trading system.

6.1 Efficiency of the DBCS Method

In this section, we make several observations on the efficiency of the DBCS method. In the set-up phase, the modules of the COTS products are first mapped to domain models by the vendors. Since the vendors have full information about the functionalities and features of the COTS products, it should be easy for them to decide how the modules are to be used in different domain modules. The mappings from the COTS modules to a domain model are reused each time a CBS from that domain is to be developed. Moreover, the mappings between the COTS modules and the domain modules help the vendors to market their products because they can demonstrate easily how their COTS products can be applied in an application domain.

Conceptually, the relation between a domain model and all of the COTS products is one-to-many. Since the mappings are developed from the COTS-product side to the domain-model side, it is still a one-to-one relation. Consequently, the complexity in developing the mapping is reduced. Furthermore, these mappings are reused in every CBS development. The mappings between the COTS products and the domain models can be managed via computing systems, and retrieving information about the COTS modules from the domain models can be done automatically. Consequently, much effort can be saved by using this approach.

In the selection phase, the first step is to identify the corresponding modules in the domain model for a CBS. This step is simple because a domain module has captured all of the features of the domain. The relation between a module of a CBS and its domain module is one-to-one. It should be easy for the developer of a CBS to identify the corresponding modules in the domain model.

Although the process of selecting the COTS products for these modules depends on the number of mappings, it is still a single and simple step because information on the COTS products can be collected through the defined mappings. Furthermore, if the mappings are well managed, after the modules of the domain models are identified, the mappings between the COTS products and the domain modules can be retrieved by automatic systems. This would be an accurate and efficient method to identify those COTS products that satisfy the functional requirements of a CBS. A weighting method based on the degree of functional fitness can then be used to help select the most suitable COTS product.

The DBCS method is an efficient method, as the one-to-many relation between a CBS and all of the COTS products is reduced to two relations, namely, a many-to-one relation between the COTS products and a domain model and a one-to-one relation between a domain model and a CBS. The latter relation is a simple relation and is handled easily. Although the former relation is complex to deal with, the vendors who possess all of the necessary information are able to easily solve this problem. We have reported the quantitative proof that the DBCS method is a more efficient method than a direct assessment method in [10].

6.2 A Case Study

We have applied the DBCS method to the development of an on-line margin-trading system and we have obtained encouraging results [11]. We have successfully developed a prototype margin-trading system for a large and leading bank in Hong Kong with the use of COTS products. The margin-trading system deals mainly with the margin trades between trading parties. It is one of the core activities in the banking and financial industry. Due to confidentiality requirements, we cannot release the functional details of this system.

In applying the DBCS method, the first requirement is the availability of a domain model for the specific application. Since an industrial-scale margin-trading domain model was not available for our research, our first step was to create such a domain model. This domain model was built by an experienced developer who had been working on a margin-trading system in a leading bank for several years. He was familiar with the business, the main activities, the detailed operations and the general architecture of a margin-trading system.

The construction of the domain model of margin-trading comprises three steps:

1. Gather domain knowledge
2. Perform domain analysis
3. Consolidate various domain models

The domain model was built using object-oriented technology and expressed in OMT (Object Modeling Technique).

In this study, we focused on the main function of the margin-trading system, namely, providing a trading environment for a dealer to perform a margin trade with various trading parties. It was decided to use COTS for two modules of the margin-trading system:

1. data security module
2. currency handling module

Fig. 7 shows the class diagram of the margin-trading system. Fig. 8 gives an example object diagram, showing a dealer starting a trading. Sample screen outputs are shown in Figures 9 and 10.

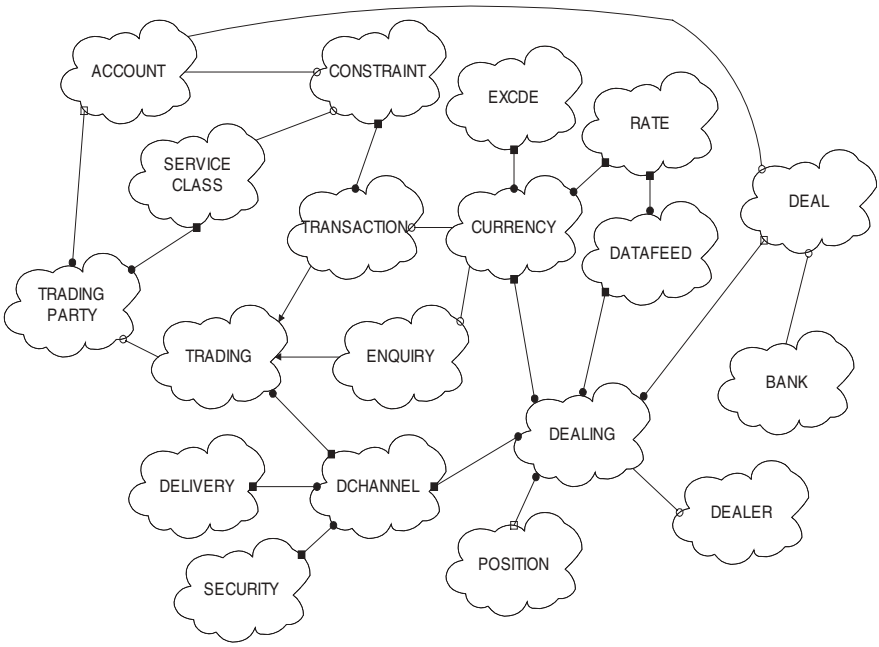

Fig. 7. Class diagram of margin-trading system

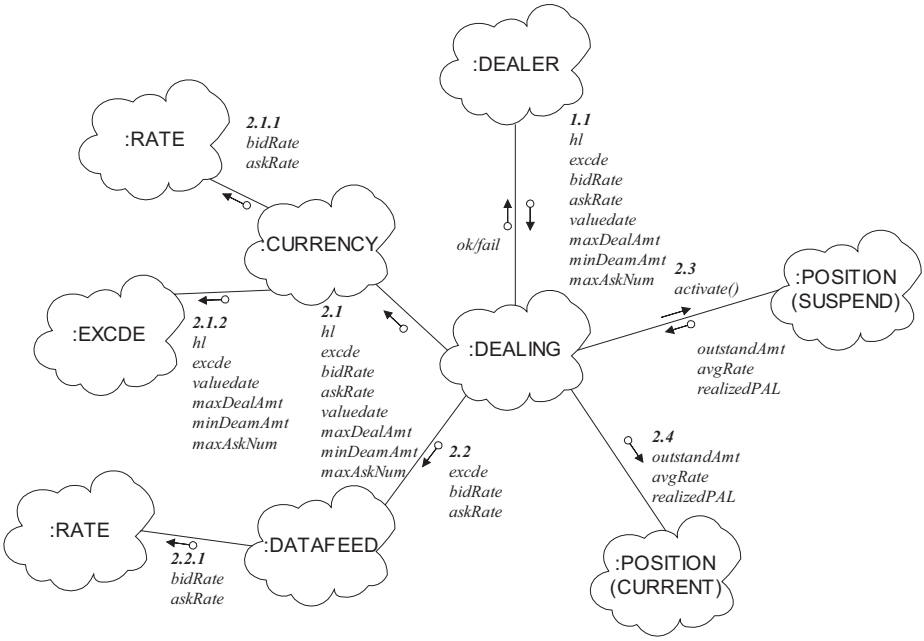

Fig. 8. Object diagram showing a dealer initializing a trading currency

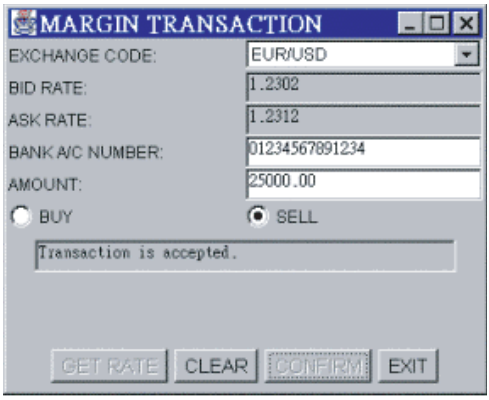

Fig. 9. Margin transaction screen at the trading site

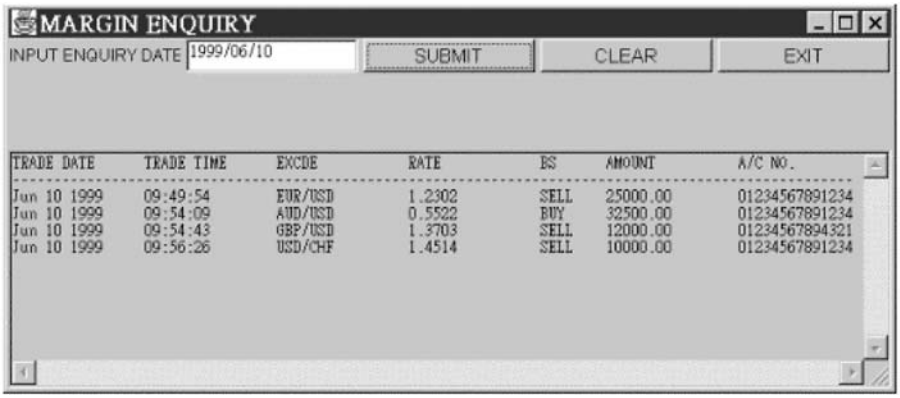

Fig. 10. Margin enquiry screen at the trading site

Before applying the DBCS to identify the best COTS available, we need to have a list of suitable COTS provided by the vendors, with the proper mapping to the domain model. As there was no such mapping available, we developed the mapping by ourselves. The non-functional criteria for the system included interoperability, performance, and ease of use.

We then mapped the modules of two COTS products to the modules of the margin-trading domain model. Afterwards, we asked the software developers to identify the COTS modules by following the steps of the DBCS method using both best-fit and first-fit strategies on the two modules.

When implementing the on-line margin-trading system in our case study, we collected data on the effort required for various activities. The data is shown in Table 2.

The values of the variables indicating the time required in various steps of our selection method are consistent with our assumptions about the expected scale of these variables.

Table 2. Values for Time Variables

Time	Scale	Effort/module
Average time for searching for a COTS product	Hours to Days	41 hours
Average time for vetting the functional appropriateness of a module of a COTS product	Hours to Days	5 hours
Average time for assessing the non-functional appropriateness of a module of a COTS product	Hours to Days	10 hours
Time required for getting information on the COTS products from the sources	Minutes	30 minutes
Average time required for identifying the modules, which are claimed to be functionally appropriate for a module of the CBS through the mappings between the COTS products and the domain model	Seconds to Minutes	30 minutes
Average time for identifying a module from a domain model	Minutes	30 minutes
Average time required for checking whether there is a mapping between a module of the COTS and the domain model	Seconds to Minutes	10 minutes

Our experience in developing the margin-trading system generated the following observations:

1. The DBCS method can reduce the complexity and improve efficiency in COTS selection. It breaks down the complicated many-to-many relationship between COTS products and system requirements into a one-to-one relation and a one-to-many relation.
2. The functional interfaces of the two COTS products used in our development did not fully match our system requirement. Software wrappers were needed to mask the mismatched interfaces. Fortunately, the effort in developing the wrappers was minimal compared to the estimated effort for developing the functionalities provided by the COTS products. We estimated that we saved two person-weeks of development time by using the two COTS products, compared to developing everything from anew.

This experiment indirectly shows that the DBCS method is better than a direct assessment method in terms of its efficiency.

Our case study also indicates that it is difficult to compare the performance of the best-fit strategy with the first-fit strategy because the performance of the latter depends on factors that cannot be fully controlled, as follows:

1. The position of the first acceptable COTS product in the sequence of COTS candidates. For example, if an acceptable COTS product happens to be the first one to be analyzed, then the first-fit strategy will succeed immediately.

2. The number of acceptable COTS products in the set of COTS candidates. If there are L acceptable COTS products in the set of N COTS candidates, then the average number of COTS products that is required to be analyzed by the first-fit strategy is L/N. The best-fit strategy will always analyze all of the N COTS products.

7 Conclusions

In this chapter, we have classified the methods for selecting COTS products into three classes: intuition, direct assessment and indirect methods. Most of the existing methods are based on selecting COTS products either by intuition or direct assessment. The former approach has the weakness of being subjective and it may omit good quality COTS candidates. The efficiency of the latter approach is inversely proportional to the product of the number of modules of the CBS in question and the total number of modules in the available COTS products. This can lead to a large selection effort that may offset the advantages of using the COTS products in the development.

We have developed a new indirect assessment method called the domain-based COTS-product selection method. We have applied the DBCS method to the development of a margin-trading system. The DBCS method reduces the complexity and improves the efficiency of COTS product selection. This is because the DBCS method takes advantage of the detailed view of the relations between the available COTS products and the CBS. Furthermore, the relations between the COTS products and the domain model are reused every time a CBS from the domain is to be developed. This helps to reduce the effort of selecting a COTS product for a CBS.

References

1. C. Alves and J. Castro. CRE: A Systematic Method for COTS Selection. In *Proc. of the XV Brazilian Symposium on Software Engineering*, Brazil, Oct. 2001.
2. C. Abts, B. Boehm and E. Bailey. COCOTS Software Integration Cost Model: an Overview. In *Proc. of the California Software Symposium*, Oct. 1998.
3. B. Boehm and C. Abts. COTS Integration: Plug and Pray. *IEEE Computer*, pages 135–138, Jan. 1999.
4. M. Broy, *et al.* What Characterizes a (Software) Component? *Software—Concepts and Tools*, 19(1):49–56, 1998.
5. Software Eng. Inst. Carnegie Mellon University. *The Capability Maturity Model: Guidelines for Improving the Software Process.* Addison-Wesley Publishing Company, Inc., Reading, Mass., 1995.
6. G. Fox, K. Lantner, and S. Marcom. A Software Development Process for COTS-based Information System Infrastructure. *Proc. of IEEE*, pages 133–142, 1997.
7. J. Kontio. A Case Study in Applying a Systematic Method for COTS Selection. In *Proc. of ICSE-18*, pages 201–209, 1996.
8. J. Kontio, S. F. Chen, and K. Limperos. A COTS Selection Method and Experiences of Its Use. In *Twentieth Annual Software Engineering Workshop*, 1995.

9. D. Kunda. Applying Social-Technical Approach for COTS Selection. In *Proc. of the 4th UKAIS Conference*, University of York, April 1999.
10. K. R. P. H. Leung and H. K. N. Leung. On the Efficiency of Domain-based COTS Selection Method. *Journal of Information and Systems Technology*, 44(12):703–715, September 2002.
11. K. R. P. H. Leung, H. K. N. Leung, and F. Suk. A COTS Selection Method Using Domain Model. Technical Report TR-20, Department of Computing, Hong Kong Polytechnic University, 1999.
12. N. Maiden and C. Ncube. COTS Software Selection: The Need to Make Tradeoffs between System Requirements, Architecture and COTS Components. In *COTS workshop*, 2000.
13. C. Rolland. Requirement Engineering for COTS-based Systems. *Information and Software Technology*, 41(14):985–990, 1999.
14. T. L. Saaty. Analytic hierarchy. In *Encyclopedia of Science & Technology*, pages 444–468. McGraw-Hill, 1997.
15. V. Tran and D. B. Lui. A Risk-Mitigating Model For The Development of Reliable and Maintainable Large-Scale Commercial-Off-The-Shelf Integrated Software Systems. In *Proc. of Annual Reliability and Maintainability Symposium*, pages 452–462, 1997.
16. V. Tran, D. B. Lui, and B. Hummel. Component-Based Systems Development: Challenges and Lessons Learned. In *Proc. of IEEE*, pages 452–462, 1997.