COTS Selection Best Practices in Literature and in Industry

Rikard Land[1], Laurens Blankers[2,3], Michel Chaudron[3], Ivica Crnković[1],

[1]Mälardalen University, School of Innovation, Design and Engineering, Västerås, Sweden
[2] Logica, The Netherlands
[3] Eindhoven University of Technology, Dept. of Mathematics and Computing Science, The Netherlands
{rikard.land, ivica.crnkovic}@mdh.se, laurens.blankers@logica.com, m.r.v.chaudron@TUE.nl

Abstract. *This paper presents an extensive literature survey of the software COTS component selection methods published to date, followed by a meta-model consolidating the activities and practices of these methods. Together with data collected from practitioners and researchers in the embedded systems domain, we provide concrete recommendations which will enable organizations to identify suitable practices when designing a customized selection processes.*

1 Introduction

As software development organizations build software using components developed by others (OTS = Off-The-Shelf; COTS = Commercial ditto), there is an increasing need to select the right components in a systematic, explicit, objective, and cost-efficient way. Many processes and methods for COTS selection have been published, many funded by, and applied in large well-reputed organizations in demanding domains (e.g. safety-critical systems, financial systems). The published processes thus build on a rich and hard-earned body of experience, but it is a major task to penetrate all relevant publications, compare them, and adopt and combine the most relevant parts for a particular organization's needs. For this purpose, we have performed an extensive literature survey and a questionnaire survey and provide recommendations which can be used as a checklist when defining the strategy and procedures in the COTS component selection process. (In the rest of this paper, the term "component" is used to mean "COTS component".)

1.1 Related Work and Scope Limitation

In 1997 a workshop on COTS-based systems [1] was organized by the SEI (Software Engineering Institute) together with the industry where a number of issues were raised which seem to have found their way into the surveyed component selection methods. The few previous literature surveys that exist are more limited than ours in various ways: one brief overview was presented four years ago [2], however without any substantial comparison (and we include the more recent methods). In another study, three of the methods were compared with eight principles of agile software development [3]. The brief survey of three earlier methods in the presentation of the method CRE [4] focused only on requirements. The relation between the selection process and surrounding processes has been described briefly in e.g. [5].

This survey includes literature which presents itself as a complete method or process for component selection. The elements of these methods (e.g. comparison methods) could each be the starting point of a major literature survey.

Section 2 presents the research methods used in this study, followed by an overview of existing published component selection methods in section 3. Section 4 presents a meta-model which consolidates the best practices of these methods. Section 5 provides recommendations for the design of customized COTS selection methods, and section 6 concludes the paper.

2 Research Method

The first part of the research has been an exploratory, systematic literature review of published COTS selection methods. In order to identify similarities and differences, we listed preliminary dimensions of comparison, and defined, populated, rejected items, and thus grown this list iteratively until we arrived at the meta-model described in section 4. See more details in our report [6]. As a second part of this study, we conducted a qualitative survey with industrial practitioners and researchers in the embedded systems domain. We constructed a questionnaire with open and qualitative questions, which was then distributed to a targeted group of experts (typically software architects/designers) in different companies and projects with the goal of collecting as many and varying opinions as possible. This approach can be expected to give a good indication of the current state of practice. We received responses from eight industry practitioners (in eight different companies), with roles central to COTS selection (e.g. software architects), plus responses from five researchers in the field. Quotes from these responses are sometimes included (chosen with the purpose of illustrating issues mentioned by several of the respondents). As an additional validation measure, we used some interview data from earlier studies on related topics.

3 Brief Survey of Component Selection Methods

Table 1 provides a historical overview of the existing, published COTS selection methods and summarizes the main novelties introduced by each (if any). For space reasons we cannot here present the methods in more detail (for this, please refer to our report [6] or directly to the references for the methods); nor do the table intend to show how methods have adopted elements from earlier methods. When a method has not been given an explicit name by its authors, we have indicated this in italics and provide an acronym based on the title of the publication or main activities of the method. The main changes discernible over time, as new methods have been proposed are: first, the list of suggested attributes to evaluate has been extended; second, the issues of architectural compatibility have become a fundamental part through the evaluation of several complementary components simultaneously as single candidates. Another difference is that some methods assume component requirements have been defined beforehand, others that they are developed the selection process.

4 Meta-model of Existing Component Selection Methods

The meta-model is in this section introduced briefly in a top-down manner; each element is then discussed in more detail in section 5.

The published methods can be described in terms of four processes (at the top of Fig. 1): there is a *preparation process*, an *evaluation process*, a *selection process* and (only in some of the methods) *supporting process(es)*. In the *preparation process*, potential component *candidates* are identified, *evaluation criteria* are defined (which are related to system *requirements* and defined with *evaluation attributes* to use as metrics), as well as a *comparison method* which determines how to do the required multi-criteria selection. The evaluation criteria are of up to four types: *Funtionality*, *Non-functional Attributes*, *Architectural Compatibility*, and *Business Considerations*. A candidate could be either a single component or a set of complementary components that are evaluated together as a candidate solution. In the *evaluation process*, actual data is collected (*data collection*) that answers the evaluation criteria and are used to perform a *comparison* of the candidates. Two types of *evaluations* can be discerned: high-level evaluation (based only on easily collected information) and *prototyping evaluation* (where the candidate component itself is available). In the *selection process*, a decision is made based on this comparison. Both the data collected and the comparison is associated with a level of *confidence*, which may range from confidence in the statistical sense (for quantifiable metrics) to the “gut feeling” when collecting qualitative data (e.g. concerning vendor claims and when evaluating the future prospects for the vendor). Other activities found in the literature can be classified as *supporting process(es)* with activities such as team formation, documentation, planning and following up the selection process, and reflecting on the selection process as such and documenting experiences for future improvement.

5 How to Design a Customized Component Selection Method

This section discusses the elements of the meta-model. For each such element, we first describe the different approaches suggested by the surveyed component selection methods, and the results from our questionnaire and secondly suggest a number of recommendations. We have for convenience labelled these with letters, but this should not be taken as a suggestion for which order to consider them in. Some recommendations are inferred mainly from the methods survey, and should be generally applicable. By also considering the questionnaire responses, we provide recommendations directed mainly to the selection of COTS for embedded systems.

Table 1. Historical overview of published component selection methods

Year	Method	Main Novelty
1995	OTSO [7] (Off-The-Shelf Option)	Progressive filtering; evaluation criteria includes functionality, non-functional properties, strategic considerations and architecture compatibility; AHP suggested for comparison
1997	PRISM [8] (Portable, Reusable, Integrated, Software Modules)	Stand-alone test phase followed by integration evaluation and field test
1998	PORE [9,10] (Procurement-Oriented Requirements Engineering)	Closely intertwined selection of components and definition of system requirements
1999	STACE [11] (Socio-Technical Approach to COTS Evaluation)	Stresses importance of non-technical factors to evaluate
2000	COTS Score [12]	-
2001	RCPEP [13] (Requirements-driven COTS Product Evaluation Process)	Stresses evaluation objectivity
2002	CAP [14]	Large number of quality metrics (>100)
	i-MATE [15]	Reusable requirements for middleware selection
	PECA [16]	Flexible structure of activities
	RDR [17] (Requirements and Design Reviews	Explicitly describes the relation between acquired components and system parts being built in-house
	CRE [4] (COTS-Based Requirements Engineering)	Requirements engineering process drives the selection; NFR framework is used to discuss non-functional attributes
	CSCC [18] (Combined Selection of COTS Components)	Considers the total cost for a system rather than specifying in advance the individual costs for different components
2003	CEP [19] (Comparative Evaluation Process)	-
2004	CARE [20] (COTS-Aware Requirements Engineering)	Intertwines system requirements engineering with component evaluation; later named CARE/SA [21] when giving software architecture a stronger focus
2005	*CCCS* [22] (Compatible COTS Component Selection)	Considers sets of complementary component as candidates, focusing on how well components will fit together; also emphasizes prototyping as a means to collect reliable information.
	CPF [23] (Commitment, Pre-filtering, Final filtering)	Strong focus on continuous improvement of the selection process itself
2006	***CSSP*** [24] (COTS Software Selection Process)	-

We expect that during the design of a customized component selection method in an organization, some additional issues need to be considered. It must therefore be ensured that all relevant stakeholders are involved in this process.

5.1 Structure of the Activities

Typically, a progressive filtering of components [1] is described in a process where candidates are evaluated, first using some easily measured but clearly discriminating criteria, and as components are discarded the level of evaluation detail and confidence

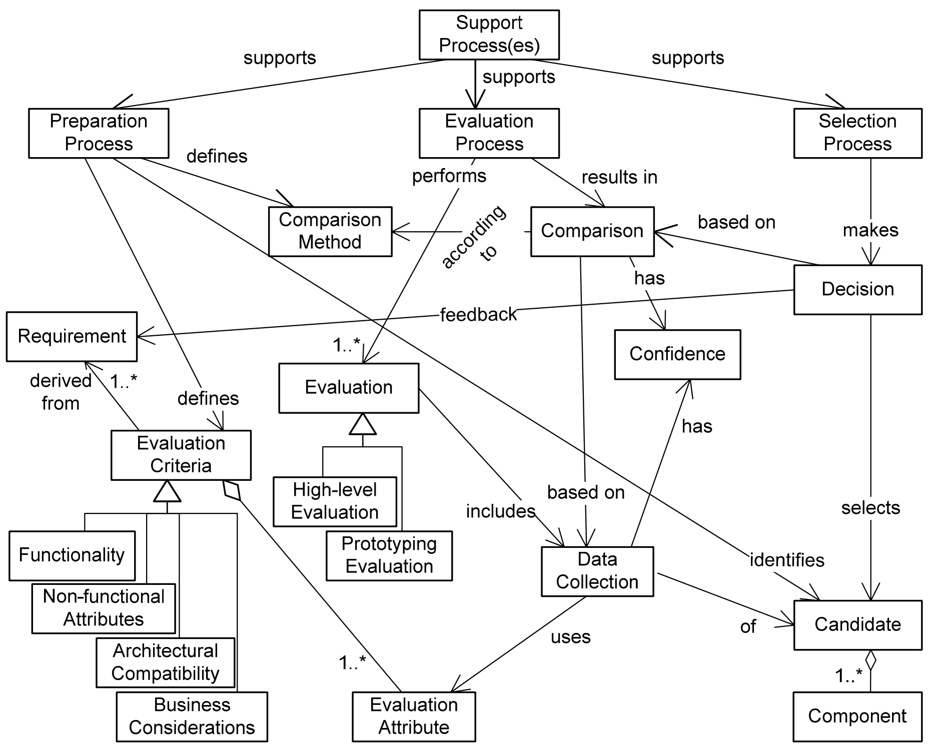

Fig. 1. Meta-model of Software Component Selection Methods

in the results is increased. The concept of an increasingly detailed evaluation of a decreasing number of components is universal, but the overall structure of the activities suggested by the published methods differs:

- **Sequential with branches (and possibly a predefined number of loops).** Methods: (PRISM), *CCCS*, (*CSSP*).
- **Iterative, i.e. continues until some exit criteria is met**: PRISM, STACE, (PECA), CRE.
- **Situation-driven/opportunistic/flexible.** I.e., given the information gathered so far, what is the most reasonable to do next? Methods: PORE, PECA.
- **Concurrent and interrelated processes.** Methods: OTSO, CAP, (CRE), CPF.

Supporting processes are mentioned in only some of the methods. Combining the suggested activities, this would include setting up a team (*CSSP*), planning and management of the evaluation and acquisition process (*CSCC*, *RDR*), and reflecting and documenting the process itself for future improvements including the actual component data collected and evaluation attributes used and how costly and useful they were during the data collection (CAP, CEP, *CCCS*). Component selection methods developed together with organizations developing embedded, safety-critical systems tend to favour waterfall-like, plan-driven component selection method (RCPEP, *RDR*, *CSSP*). However, we believe the main reason for this is that these organizations are used to plan-driven processes in general. Generally, our complementary data instead suggest that component assessment and selection should also for safety-critical and mission-critical systems be iterative, opportunistic and flexible rather than plan-driven. The COTS selection process can (with advantage) be seen as part of the requirements and design phases (which may however be part of a formalized plan-driven process). Two illustrative questionnaire responses:

- "It is inevitable that new criteria emerge"
- "…it is like when people make the decision. We always make up our mind with the information gathered so far and choose the most optimistic option."

This analysis results in the following three recommendations when selecting an overall layout:

Recommendation A: If your organization prefers some specific structure of activities (i.e. sequential, etc.), study the references to the presented methods in the bulleted list above in more detail.

Recommendation B: Strongly consider a requirements-driven and iterative or flexible selection process – even if you are developing highly critical software and are used to plan-driven processes.

Recommendation C: Since the supporting processes are mostly out of scope of the published methods, use combinations of available supporting processes (by following the references above), and when possible combine them with other sources of good practices (e.g. process guidelines already existing internally in the organization, or general process standards).

5.2 System Requirements

The methods differ in how they consider the relation between requirements engineering and component selection. A few selection methods describe themselves as driven by the requirements engineering process (CRE, CARE). In other methods the requirements are developed simultaneously with the component selection process (PORE, i-MATE). However, the majority of the methods assume system requirements exist (OTSO, PRISM, STACE, COTS Score, RCPEP, CAP, *CSCC*, PECA, *RDR*, CEP, *CCCS*, CPF, *CSSP*), but it is typically mentioned that the requirements can be renegotiated based on the component evaluation (most explicitly in PRISM, STACE, CAP, *CCCS*). PECA stresses that system requirements have to be translated into component evaluation requirements, which are not identical for all components under evaluation. CEP points out that evaluation criteria should be broad so as not to limit the search by too many constraints. The questionnaire respondents prefer loosely defined system requirements before components are searched for; example quotes:

- "detailed but on a high level of abstraction"
- "It depends on the product to be developed. It could be that the system requirements could be changed due to available software component."
- "Too detailed specification might make it unlikely that suitable components can be found"
- "Very often, clients only have a very vague idea about the functions the system should provide. Due to time constraint, the system requirements could be formulated and specified in parallel with the system development."

This discussion again leads to recommendation B in section 5.1.

5.3 Evaluation Criteria

The methods in combination suggest four types of evaluation criteria:

- **Functionality.** (Essentially all methods.)
- **Non-functional attributes.** (Essentially all methods.)
- **Architectural compatibility.** (OTSO, PRISM, CAP, PECA, *RDR*, CRE, CARE, *CCCS*, CPF, *CSSP*; we can also note that architectural compatibility is addressed when sets of components are evaluated together, see section 5.6.)
- **Business considerations.** This includes evaluation of the vendors (e.g. their reputation and financial stability; RCPEP, PECA, *RDR*, CARE, *CCCS*, *CSSP*), estimated cost and risk in both the short and long term (considering e.g. available support for the component, frequencies of updates, maintaining backward compatibility and going out of business; RCPEP, CAP, *RDR*, CRE, CARE, *CCCS*), and organization infrastructure (e.g. skills; *RDR*, CRE).

The methods generally do not specify some particular order of importance among these factors, and the general opinion among the questionnaire respondents is that "it depends" (on the domain, on the organization, on the particular system and component criticality, etc.). If anything, cost seems to be the least important, because "the proper solution will save money in the long run" (quote); however any per-deployment cost (e.g. in terms of licensing or hardware resources) is an important factor for products with large volumes. Several of the questionnaire respondents emphasize architectural compatibility. Thus, everything is important, and the non-fulfilment of a single individual evaluation criterion could exclude a component.

As said, some selection methods interleave system requirements with search for and evaluation of components (PORE, i-MATE, CRE, CARE). One reason is that it is not trivial to decompose system requirements into component requirements; one questionnaire respondent explained it as: "In addition to the component requirements, there will be additional requirements concerning several components and their interconnections (e.g., end-to-end deadlines)." Most component selection methods however assume that component requirements exist, to which the component features can be related. Concerning the level of detail to which component requirements should be specified, the questionnaire responses are inconclusive:

- "The first challenge is decomposition of the system requirements, and the initial system architecture. This architecture can then be used to loosely define the component functionalities and then their behaviour."
- "Until the system requirements are understood, it is hard to make a selection between similar choices."
- "a range of functionality is initially chosen"
- "Too much details may exclude components that could be appropriate."
- "component market within [domain] is non-existing to some extend. Component requirements could be specified in more detail, if the component market is larger."
- "[challenging to] define the responsibilities and interfaces (both syntax and semantics) … at a stage that not everything about the architecture is known yet"

This analysis results in the following two recommendations:

Recommendation D: The following four types of attributes should all be considered: functionality, non-functional attributes, architectural compatibility, business

considerations. Elaborate what they mean more specifically for your particular system and organization, and their relative importance.
Recommendation E: Consider what level of detail component evaluation criteria need to be specified in advance, since this depends on the system and organization.

5.4 Evaluation attribute, Data collection, and Confidence

It is possible to distinguish between two types of evaluations, based on how the actual data collection can be carried out (visible in most methods as well as in the questionnaire responses). We label these two phases *high-level evaluation* and *prototyping evaluation*. Some methods describe these as explicit phases (e.g. PRISM); in others they are implicit, as a consequence of iteration and refinement.

In the first type of evaluation, *high-level evaluation*, typically many components are briefly evaluated based on information about components and vendors, gathered e.g. from in-house sources, literature reviews and interviews with other customers, from the vendors in the form of marketing material, by request to the vendor, vendor appraisals, or by publicly available information about the financial stability of the vendor. Illustrative questionnaire quote: “Get opinions from different people, other departments, or even other companies (use network to gather experiences)”. In the high-level evaluation, all four types of evaluation criteria should be considered (functionality, non-functional attributes, architectural compatibility, and business considerations). There are several things to bear in mind when planning this high-level evaluation: to increase confidence in the results several sources of information should be used (triangulation), focus should be on information that can discriminate between components (PECA), and criteria should be selected for which data are easy to find.

A limited number of candidates are then selected for the second type of evaluation, *prototyping evaluation*, where the actual components are used for prototypes, systematic tests and/or experiments. This is done to assess certain properties in the context of the envisioned system with a high degree of confidence, and also to learn and understand the component. Prototyping is explicit and important in some methods (PRISM, PORE, RCPEP, i-MATE, PECA, *CCCS*) and stressed by several questionnaire respondents (example: “involving the component prior to deployment and by using extensive simulation, monitoring, or testing of the composition”).

The main distinction between the two evaluation phases is whether the component needs to be available during the evaluation or not. In prototyping, the acquisition of a component may come with a (high) cost and can introduce (substantial) effort into the evaluation process, and the evaluation itself requires learning the component and systematically setting up, executing, and documenting many tests thoroughly. This consequently limits the number of components that can practically be evaluated.

This analysis results in the following five recommendations:
Recommendation F: Use as discriminating evaluation attributes as possible, to ensure an efficient filtering process.
Recommendation G: Use evaluation attributes for which data is as easy to collect as possible, to ensure an efficient filtering process.

Recommendation H: Based on the (expected) number of existing component, the criticality of the components when used in the envisioned system, estimate how much time is acceptable and needed for high-level evaluation.
Recommendation I: Consider how many components are expected to be subject to prototyping evaluation and how detailed the evaluation needs to be (which is a consequence of the required confidence), and estimate the time and cost accordingly.
Recommendation J: If there is a conflict between project budget and the evaluation estimates, the issue need to be satisfactory resolved as early as possible; in general it can be expected that choosing an insufficient component will negatively affect the system development greatly.

5.5 Comparison Method, Comparison, and Decision

The comparison and ranking of components is naturally based on many criteria. The evaluation of the criteria could either be made subjectively, or could be based on or supported by a systematic comparison method. The comparison method most commonly proposed by the published component selection methods is AHP, Analytical Hierarchy Process (OTSO, PORE, STACE, COTS Score, CRE). Other methods are WSM (Weighted Scoring Method) and Weighted Average (RCPEP, CRE, CEP), and using COCOTS [25] for effort estimation (CRE, *CSSP*). Our questionnaire suggests that these are with a few exceptions not known to practitioners (i.e. there is an adoption cost associated with these methods). Others have argued that all of these evaluation methods ("decision-making techniques") have their drawbacks when applied to component selection [26]: the techniques may require disproportionate effort, requiring stakeholders to provide preferences and weights for many criteria and specify how to aggregate the criteria into a one-dimensional scale (i.e. ranking) in the absence of concrete products, which is difficult and inefficient. Instead, gap analysis is suggested [26], meaning that for each component, the gap between requirements and provided capabilities is analyzed, followed by an estimation of the costs of bridging the gap. Since a formal comparison runs the risk of not catching the intent of the comparison, some methods also suggest or mention discussions, reasoning, and argumentation techniques (CARE, PECA, PORE). The questionnaire responses suggest that although formalized techniques bring a necessary structure into decision-making, they are using subjective and incomplete input. They must therefore be complemented with informal discussions to ensure "the 'real' issues" are ultimately considered (quote from a questionnaire response). The complexity of the decision is illustrated by another quote: "The final selection is based on a combination of the technical evaluation, the related business case for the tool and vendor, and trying to optimize cost. The final selection is always a trade off."

The option of building a component in-house may in some cases be a realistic solution (especially if no suitable component is found), and a few methods discuss this (CAP, (CRE), *RDR*, *CCCS*). The build alternative can be treated as one alternative among others during gap analysis, with an associated effort, cost, risk etc.

This analysis results in the following recommendation:
Recommendation K: Combine a formalized comparison method with (structured) discussions. Consider gap analysis for the formalized comparison method.

5.6 Components and Candidates

It has been proposed that combinations of the available components should be evaluated together as a single candidate (*CSCC*, *CCCS*). There are two reasons for this: first, to minimize architectural mismatch, and secondly because the (hypothetical) choice of an initial component will help decomposing system requirements into component requirements (a "crystallizing seed"). This "puzzle assembly" was identified in the previously mentioned SEI workshop in 1997 [1] but has only become explicitly exploited in two of the recent selection methods. *CSCC* implements this approach by comparing the estimated total system cost using various component alternatives (rather than focusing on the cost of individual components), and *CCCS* by explicitly considering a "candidate" to be a set of components which are architecturally compatible. The questionnaire responses indicate this is the advantageous approach: "Integration can be difficult otherwise"; "Systems have to work together as a whole, and decisions cannot be made in isolation". Apart from the technical aspect of integration, business considerations are also addressed by this approach; for example, if already several risky components are used, a project might want to avoid including more (risky) components in a project. A related approach is to maintain a list of potential components for each "slot" in the architecture. If further downstream (also after development and deployment) a component is found to be insufficient (e.g. too low quality, or support is discontinued) the list will help identify a replacement component.

"Keystone identification" [1] means the selection of a central component, technology, or strategy that will have a great impact the selection of other components (e.g. "we will build on .NET", "we will use middleware from a certain vendor and then choose other products known to integrate well"). None of the surveyed methods implement this strategy explicitly. However, the questionnaire responses indicate that this commonly happens in practice:

- "Yes it is common. E.g. LINUX vs Windows."
- "For example, a central database may be the most important part of a system for functionality and performance. That choice needs to be optimized, and other choices must be made with respect to that decision."
- "If another platform is chosen (e.g. VxWorks in stead of Windows CE) this has a lot of impact on the available components."

This analysis results in the following two recommendations:

Recommendation L: Evaluate combinations of components together, in order to address architectural mismatch inherently in the process.

Recommendation M: Identify any keystone technologies, platforms, and strategies early in the process, since that will exclude many other components.

5 Summary and Conclusion

In this paper we have surveyed published software component selection methods, and provided a meta-model which provides a common terminology and comparison framework for selection methods. By bringing the collected best practices into the light, and with the additional data provided by a questionnaire distributed to software architects and researchers in the embedded systems domain, we have provided 13 recommendations which will help organizations to more rapidly design customized COTS selection processes. In brief summary, our recommendations are:

- Use four types of evaluation criteria: functionality, non-functional attributes, architectural compatibility, business considerations.
- Consider an iterative process intertwined with requirements engineering.
- Address architectural compatibility by evaluating combinations of components, and consider the cost of the total system rather than individual components.
- Consider the criticality of components, and what level of confidence is needed in the evaluation and selection decision, and allocate sufficient resources.

This work will be followed up by industrial case studies of component-based systems life-cycles and processes.

6.1 Acknowledgements

This work was partially supported by the Swedish Foundation for Strategic Research (SSF) via the strategic research centre PROGRESS. The authors would like to thank the questionnaire respondents, the people who have reviewed earlier versions of this paper, and those who in other ways been helpful in this study.

References

1. P. Oberndorf, L. Brownsword, E. Morris, C. Sledge, “Workshop on COTS-Based Systems”, Special report CMU/SEI-97-SR-019, SEI, 1997.
2. G. Ruhe, “Intelligent Support for Selection of COTS Products”, *NODe 2002* Revised Papers, LNCS vol. 2593, Springer 2003.
3. F. Navarrete, P. Botella, X. Franch, “How Agile COTS Selection Methods are (and can be)?” *Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 160-167, IEEE, 2005.
4. C. Alves, J. Castro, “CRE: a systematic method for COTS components Selection”, *Proceedings of the XV Brazilian Symposium on Software Engineering (SBES)*, Rio de Janeiro, 2001.
5. I. Crnkovic, M. Chaudron, S. Larsson, “Component-based Development Process and Component Lifecycle”, *Journal of Computing and Information Technology*, 13(4), pp. 321-327, 2005.
6. R. Land, L. Blankers, *Classifying and Consolidating Software Component Selection Methods*, MRTC report ISSN 1404-3041 ISRN MDH-MRTC-218/2007-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, November, 2007.
7. J. Kontio, *OTSO: A Systematic Process for Reusable Software Component Selection*, Univ. Maryland report CS-TR-3478, UMIACS-TR-95-63, 1995.
8. R.W. Lichota, R.L. Vesprini, B. Swanson, “PRISM Product Examination Process for component based development”, In *Proceedings Fifth International Symposium on Assessment of Software Tools and Technologies*, IEEE, 1997.

9. N. A. Maiden, C. Ncube, “Acquiring COTS Software Selection Requirements”, *IEEE Software*, 15(2), pp. 46-56, March 1998.
10. C. Ncube, N. A. Maiden, “PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm”, *Second International Workshop on Component-Based Software Engineering*, Los Angeles, 1999.
11. D. Kunda, L. Brooks, “Applying Social-Technical Approach For Cots Selection”, Proceedings of the *4th UKAIS Conference*, York, McGraw Hill, 1999.
12. A.T. Morris, “COTS Score: an acceptance methodology for COTS software”, *Proceedings of the 19th Digital Avionics Systems Conferences (DASC)*, Vol. 1, pp. 4B2/1-4B2/8, 2000.
13. P. K. Lawlis, K. E. Mark, D. A. Thomas, T. Courtheyn, “A Formal Process for Evaluating COTS Software Products”, *IEEE Computer,* 34(5), 2001.
14. M. Ochs, D. Pfahl, G. Chrobok-Diening, and B. Nothhelfer-Kolb, “A COTS Acquisition Process: Definition and Application Experience”, ISERN report 00-02, Fraunhofer Institute for Experimental Software Engineering (IESE), 2002.
15. A. Liu and I. Gorton, “Accelerating COTS Middleware Acquisition: The i-Mate Process”, *IEEE Software,* 20(2), pp. 72-79, March 2003.
16. S. Comella-Dorda, J. Dean, E. Morris, P. Oberndorf, “A Process for COTS Software Product Evaluation”, *ICCBSS*, LNCS vol. 2255, pp. 86-96, Springer, 2002.
17. M. Morizio, C. B. Seaman, V. R. Basili, A. T. Parra, S. E. Kraft, and S. E. Condon, “COTS-based software development: Processes and open issues”, *Journal of Systems and Software,* 61(3), pp. 189-199, Elsevier, 2002.
18. X. Burgués, C. Estay, X. Franch, J. A. Pastor, C. Quer, “Combined Selection of COTS Components”, *ICCBSS*, LNCS vol. 2255, pp. 54-64, Springer, 2002.
19. B. C. Phillips, S. M. Polen, “Add Decision Analysis to Your COTS Selection Process”Software Technology Support Center Crosstalk, April 2002.
20. L. Chung and K. Cooper, “Defining Goals in a COTS-Aware Requirements Engineering Approach”, *Systems Engineering,* 7(1), pp. 61-83, Wiley, 2004.
21. L. Chung and K. Cooper, “COTS-Aware Requirements Engineering and Software Architecting”, Proceedings of the *4th International Workshop on System/Software Architectures (IWSSA)*, 2004.
22. J. Bhuta, B. Boehm, “A Method for Compatible COTS Component Selection”, *ICCBSS*, LNCS vol. 3412, Springer, 2005.
23. A. Cechich, M. Piattini, “Filtering COTS Components Through an Improvement-Based Process”, *ICCBSS*, LNCS vol. 3412, Springer, 2005.
24. H. Lin, A. Lai, R. Ullrich, M. Kuca, J. Shaffer-Gant, S. Pacheco, K. Dalton, K. McClelland, W. Watkins, S. Khajenoori, “COTS Software Selection Process”, SANDIA REPORT SAND2006-0478, Sandia National Laboratories, May 2006.
25. Chris Abts, “Extending the COCOMO II Software Cost Model to Estimate Effort and Schedule for Software Systems Using Commercial-Off-The-Shelf (COTS) Software Components: the COCOTS Model”, Ph.D. Dissertation, University of Southern California, October 2001.
26. Cornelius Ncube and John C. Dean, “The Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Components”, *ICCBSS*, LNCS vol. 2255, pp. 176-187, Springer, 2002.