Classifying and Consolidating Software Component Selection Methods

Rikard Land
Mälardalen University
Department of Computer Science and Electronics
PO Box 883, SE-721 23 Västerås, Sweden
rikard.land@mdh.se

Laurens Blankers
LogicaCMG
Public Sector
Postbus 3190
2280 GD Rijswijk
Netherlands
laurens.blankers@logicacmg.com

Abstract

Virtually all software systems built today include pre-existing components developed by others (OTS, Off-the-Shelf or COTS, Commercial ditto). This trend has been accompanied by research into, among others, methods for evaluating and selecting components to use in a system. This paper presents a literature survey of the software component selection methods published to date. Based on this survey, a meta-model is presented, which allows for easy comparison of the methods. For each part of the meta-model, we present the best practices collected from all the existing models, thus presenting the collected experience of many research efforts in a checklist-like way. The model and practices presented are useful when choosing a method for a particular project.

1. Introduction

As software development organizations build software using components developed by others, there is an increasing need for selecting the right components in each specific project in a systematic, repeatable, objective, cost-efficient way. We have therefore seen a rich flora of various processes and methods for component assessment and evaluation, many of which are funded by – and applied in – large well-reputed organizations building large complex systems, such as Hughes, Lockheed Martin, NASA, Siemens, the United States Department of Energy, and UK Ministry of Defence. Many methods are created and evaluated in cooperation with well-renowned research institutes such as the Software Engineering Institute (SEI) in the US and Fraunhofer Institute for Experimental Software Engineering (IESE) in Germany. It is therefore assumed that these published processes and methods build on a rich and hard-earned body of experience.

Many of these methods display striking similarities and embed many best practices based on the hard-earned experiences from above mentioned organizations. We feel that the field is mature enough to consolidate the collected experiences, which we do in the present paper by presenting:

1) a meta-model by which it is possible to describe the existing processes in a uniform manner, compare them, and choosing the one(s) best suited for the project at hand.
2) the collected best practices embodied in the publications.

It is possible to discern an evolution from earlier to later publications, and this paper summarizes the collected best practices rather than grade or rank the methods.

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 assessment methods. There exist few previous surveys or summaries of component selection methods. The ones that do exist are more limited than the survey presented here in various ways: one brief overview of component assessment methods was conducted 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]. There is also a brief survey of three earlier methods in the presentation of the method CRE [4], but the survey is

short and focused 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. Each element of these methods, such as evaluation, ranking, metrics etc. (as they will be presented later in our meta-model), could in itself be the starting point of a major literature survey.

1.2. Paper Outline

The outline of the paper is as follows: Section 2 describes the research method used. A historical overview of the methods is given in section 3, followed by the presentation of a meta-model which enables comparison of the methods in a structured way in section 4. The activities of the existing methods are categorized in the context of the meta-model in section 5, and section 6 collects the best practices of the methods organized per element of the meta-model. In section 7 we discuss various other observations, and finally section 8 concludes the paper.

2. Research Method

The term "systematic literature review" has previously been used to denote a study to find answers to a more specific research question [6]; the survey this paper is based on intends to be systematic, but is an explorative survey of existing publications where the goal was to identify similarities and differences along a number of dimensions, unknown at the outset of the study. We have used certain keywords to search major publication databases (IEEE, ACM, Citeseer, Springer, Google scholar, and a university system which searches several databases simultaneously). We have also included publications already known to us, as well as followed (recursively) all promising references. Throughout our search, we have listed preliminary dimensions of comparison, and defined, populated, rejected items, and thus grown this list iteratively. This was later refined in a more detailed meta-model which enables discussing similarities and differences in a uniform way. For pedagogical reasons, this paper presents these two stages in the opposite order, i.e. the meta-model is presented first, after which the existing methods are presented in terms of the meta-model in a large table inspired by surveys in other fields [7].

Additionally we have studied four different companies by interviewing one architect in each company. In two of the companies, which are building dependable embedded systems, the interviewees represent concrete cases of component selection, while the other two interviewees work as independent consultants with experiences from a number of component selection projects. We use this data to make our own independent validation of the best practices suggested in the literature; more details on this part of the study can be found in [8,9].

3. Historical Overview

This section introduces the surveyed methods in chronological order of their first publication, and we limit the descriptions to the main novelties introduced. In case the methods have not been given an explicit name by their authors, we have indicated this in italics and provided an acronym based on the title of the publication or main activities of the method.

1995: The OTSO [10] method (Off-The-Shelf Option) was the first method to be published, and used in Hughes corporation when developing a system for NASA. In many ways OTSO set the scene for component selection methods to come, by introducing the basic idea of progressive filtering, dividing evaluation criteria into not only functional and non-functional properties of the component, but also strategic considerations and architecture compatibility, and by suggesting a particular method for comparison of candidates (AHP, Analytical Hierarchy Process).

1998: As the name indicates, PORE [11,12] (Procurement-Oriented Requirements Engineering, developed and used in a project with the UK Ministry of Defence), introduced a somewhat different viewpoint, namely that selection of components and the definition of system requirements should be closely intertwined. In later publications the method was used for the banking domain and named BANKSEC [13] by its creators.

1999: STACE [14] (Socio-Technical Approach to COTS Evaluation) contributed by stressing the importance of non-technical factors to evaluate.

2000: COTS Score [15] intends to be a decision support tool by formalizing evaluation criteria.

2001: RCPEP (Requirements-driven COTS Product Evaluation Process) [16] stresses evaluation objectivity.

2002: In 2002, a number of methods were published. CAP [17] (created at Fraunhofer Institute for Experimental Software Engineering (IESE) together with Siemens) introduced some hundred quality metrics to evaluate, although they seem to never have been published in detail. The i-MATE [18] method (studied in 5 cases) focuses on middleware selection, and the main contribution is the description of reusable requirements for that domain. PECA [19] (Plan, Establish, Collect, Analyze, from the SEI) can be noted for its flexible structure of activities, and *RDR* [20] (Requirements and Design Reviews, developed and

used at NASA Goddard) describes well the relation between acquired components and system parts being built in-house. In CRE [4] (COTS-Based Requirements Engineering) the requirements engineering process drives the selection, and the fairly established NFR framework is used to discuss non-functional attributes. *CSCC* [21] (Combined Selection of COTS Components), intends to consider the total cost for a system rather than specifying in advance the individual costs for different components.

2003: In this year, CEP [22] (Comparative Evaluation Process) was published.

2004: CARE [23] (COTS-Aware Requirements Engineering) also intertwines system requirements engineering with component evaluation and selection; later named CARE/SA [24] when giving software architecture a stronger focus.

2005: *CCCS* [25] (Compatible COTS Component Selection) consider sets of complementary component as candidates, focusing on how well components will fit together; it also emphasizes prototyping as a means to collect reliable information. In CPF [26] (Commitment, Pre-filtering, Final filtering) a strong focus is continuous improvement of the selection process itself.

2006: *CSSP* [27] (COTS Software Selection Process) was published, developed by Lockheed Martin for the US Department of Energy.

The main changes discernible over time, as new methods have been proposed are:

1) that the suggested attributes to evaluate has grown and matured,
2) that the issues of architectural compatibility have become a fundamental part through the evaluation of several complementary components simultaneously as single candidates, and
3) that different viewpoints have been introduced concerning whether component requirements are assumed to exist or if they are developed and negotiated during the selection process.

4. Meta-model

The published methods can be described in terms of four processes (at the top of Figure 1): there is a *preparation process*, an *evaluation process*, a *selection process* and *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. A candidate could be either a single component or a set of complementary components that together form a candidate.
- In the *evaluation process*, data is collected (*data collection*) which are used to perform a *comparison* of the components. The types of data collected can be very diverse and include not only direct measurements and tests of the component itself, but also qualitative statements such as other customers' opinion about the vendor; this is further discussed in section 6.3.
- In the *selection process*, a decision is made based on this comparison. Both *data collection* and *comparison* have a *confidence*, which may range from confidence in the statistical sense (for quantifiable metrics) to the "gut feeling" when collecting very 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 the forming of teams, documentation, planning and following up the selection process, and reflecting on the selection process as such and documenting experiences for future improvement.

Note that in the meta-model, we do not assume or recommend any particular order of the activities, since the existing methods differ significantly in this matter. (This is the reason we chose the term "process" instead of "activity" or "phase".) There is nevertheless an implicit order of some activities in the processes determined by the data flow from definitions (e.g. evaluation attribute) to executions (e.g. data collection). It can be noted that the meta-model focuses on activities and their attributes, and does include artifacts (such as evaluation results). Given this limitation, we claim that the meta-model can be instantiated to describe all existing COTS-selection methods to date; this is not to say that there may not appear other, fundamentally different methods in the future. In fact, the meta-model itself may stimulate totally new ideas and the invention of new methods.

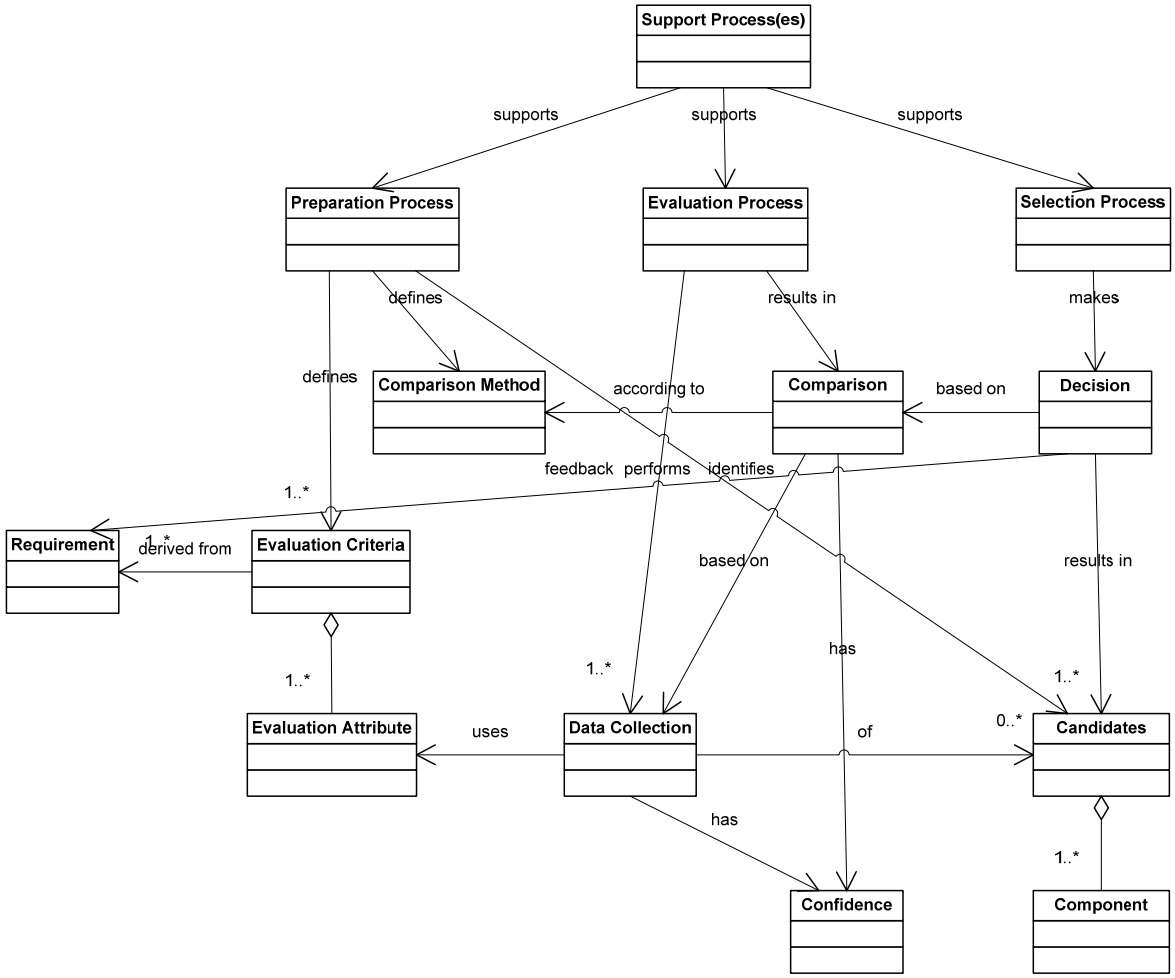

Figure 1: Meta-model of Software Component Selection Methods

5. Activities in Existing Methods

The surveyed methods differ significantly in the description given by the respective authors, not least in how the activities are ordered, e.g. sequentially, iteratively, or not pre-defined but rather flexible or opportunistic. Nevertheless, there are a number of similarities between most methods. Typically, the overall method is described as a filtering process which starts out with many candidates with some easily measured but clearly discriminating criteria, and continues by gradually discarding candidates while increasing the level of evaluation detail and confidence in the results at the expense of more time and effort invested into the selection. Some methods describe this as a fixed sequence of phases, others by a less pre-defined structure in terms of iterations or an even more flexible or opportunistic structure of activities; however the concept of increasing detail for a decreasing number of components is universal. The majority of the methods assume there exist component requirements (including non-technical objectives and constraints, such as cost) to which the component features can be related, while some build on a more iterative model where system requirements are developed and elaborated while surveying available components.

Table 1 maps the activities of the methods into the processes of the metamodel, thus highlighting the similarities while also indicating how and where they differ. (Some of the methods listed earlier in section 3 are omitted from the table below because of their clearly limited scope or limited contribution – e.g. COTS Score is an evaluation technique not suggesting any particular set of activities.)

Table 1: The activities of the phases, mapped to the processes of the meta-model

Method[*] ***Main flow***	**Preparation**	**Evaluation**	**Selection**	**Supporting Process(es)**
OTSO [10] *Five concurrent processes*	1. Define the evaluation criteria	2. Search 3. Screening 4. Evaluation 5. Analysis of results	6. Deployment	7. Assessment
PORE [11,12]/BANKSEC [13] *Situation-driven*	1. Requirements acquisition (in close iterations with evaluation)	2. Supplier selection 3. Software package selection 4. Contract production 5. Package acceptance	-	6. Management of system procurement
STACE [14] *Iterative*	1. Select underlying technology 2. Define social-technical criteria 3. Search and screen 4. Revise requirements	5. Evaluate…	5. … and select	-
RCPEP [16] *Sequential*	1. Trade study	2. Evaluation based on vendor and user input 3. Narrowing the field 4. Hands-on evaluation	5. Final analysis	-
CAP [17] *Three concurrent processes*	Initialisation (CAP-IC) Execution (CAP-EC): *Exploration, Screening,*	Execution (CAP-EC): *Ranking*	Execution (CAP-EC): *Decision*	Reuse (CAP-RC)
CSCC [21] *Four phases, a "global level" and "local level"*	2. Configuration of scenarios to be evaluated	(Evaluation of local scenarios) 3. Evaluation of global scenarios	4. Final selection	1. Initial planning
i-MATE [18] *Sequential*	1. Elaborate customer requirements 2. Augment with generic requirements 3. Rank overall requirements 4. Identify candidate products	5. Product evaluations	6. Product selection	-
PECA [19] *Flexible (iterative)*	1. Planning 2. Establish criteria (In opportunistic iterations with 3 & 4)	3. Collect Data 4a. Analyze data (In opportunistic iterations with 1 & 2)	4b. Analyze data: *Making recommendations*	-
RDR [20] *Sequential*	1. Requirements Analysis 2. System Requirements Review	3a. Package Identification/ Evaluation/ Selection	3b. Non-COTS development 4. Identify Glueware and Integration Requirements 5. System Design Review 6. Write Glueware and Interfaces 7. Integration and Test 8. … (later life cycle phases)	Project Management
CRE [4] *Four iterative (and somewhat parallel) phases*	A. Identification B. Description	C. Evaluation	D. Acceptance	-
CEP [22] *Sequential*	1. Scope Evaluation Effort 2. Search and Screen Candidate Components 3. Define Evaluation Criteria	4. Evaluate Component Alternatives	5. Analyze Evaluation Results	6. Preserve Evaluation Data

[*] Acronyms in italics are for methods not named by their authors but inferred by us from the methods' publication titles.

Method* ***Main flow***	**Preparation**	**Evaluation**	**Selection**	**Supporting Process(es)**
CARE [23] *Sequential*	A11. Define System Agents	A12. Define System Goals (with COTS) including *define COTS goals* A13, A14, A15. Define System & Software Requirements, Define architecture (with COTS)	-	-
CCCS [25] (Compatible COTS Component Selection) *Sequential with branches and feedback loops*	0. Entry conditions: OC&P 1. Identify 2. Classify	3. Evaluate 4. Buy information 5. Filter out 6. Evaluate combinations 7. Prototype	9. Decision making 10. Re-negotiate OC&P 11. Develop custom component	8. Preserve options
CPF [26] *Three interrelated processes*	A1-1: Derive goals A1-2: Compute preferences	A2-1: Functional Suitable Measurement A2-2: Functional Suitability Analysis A3-1: Architectual Adapatibility Measument A3-2: Architectual Adapatibility Analysis	-	-
CSSP [27] *Sequential (possible to step back)*	3. Identify COTS criteria	4. Apply level I filter 5. Apply level II filter 6. Analyze data…	6. … and document results	1. Form an evaluation team 2. Apply Team non-sw Process

As Table 1 shows, the methods focus on different processes in the metamodel. Of particular interest is the supporting process(es), which is only mentioned in some of the methods. Combining the suggested activities, this would include setting up a team, planning and management of the evaluation and acquisition process, and reflecting and documenting the process itself for future improvements (including e.g. evaluation attributes used and how costly and useful they were during the data collection). Also, after the system has been deployed more insight into the selection process may be collected which was not obvious during the process itself.

6. Best Practices

This section discusses several best practices found in multiple methods. The section is organized based on the meta-model.

6.1. System Requirements and Evaluation Criteria

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, 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 explicit in 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.

Essentially all methods agree that the evaluation criteria should include functional and non-functional attributes. Many also name architectural compatibility as an important factor (OTSO, CAP, PECA, *RDR*, CRE, CARE, *CCCS*, CPF, *CSSP*; note that this is partly addressed by the method to evaluate component sets as candidates, see section 6.4). Business considerations are also listed, such as evaluation of the vendors (RCPEP, PECA, *RDR*, CARE, *CCCS*, *CSSP*), estimated cost and risk in both the short and long term (RCPEP, CAP, *RDR*, CRE, CARE, *CCCS*,), and organization infrastructure (e.g. skills; *RDR*, CRE).

It is difficult to say something about the relative importance of these types of requirements, but it appears that not fulfilling any one of them satisfactory could act as an inhibitor for selecting a particular component. Clearly, functionality and quality are important, but one study reports that “architecture is

more important than requirements for product selection" [28].

Another study claims that "familiarity [with the COTS product] is often the only attribute considered" [28].

There are some specific guidelines to be found in the published methods. CRE [4] builds on a fairly established framework of non-functional attributes (NFR framework [29]), CAP uses a list of a hundred metrics (which has not been published in detail however), and i-MATE provide generic, reusable requirements for a particular domain (middleware).

6.2. Comparison Method, Comparison, and Decision

The comparison and ranking of components is naturally based on many criteria. Evaluating the criteria could either be made subjectively, or could be based on or supported by a systematic comparison method. The most commonly proposed comparison method is AHP, Analytical Hierarchy Process (OTSO, PORE, STACE, COTS Score, CRE). Other methods mentioned include WSM (Weighted Scoring Method) and Weighted Average (RCPEP, CRE, CEP), and using COCOTS [30,31] for effort estimation (CRE, *CSSP*). PECA also suggests a sensitivity analysis to be performed, i.e. a statistical analysis of how sensitive the resulting component ranking is with regard to individual criteria and evaluations. i-MATE [18] mentions a spreadsheet tool, but does not inform how it is constructed. COTS Score also mention Multi-attribute Utility Theory, Multiple Criteria Decision-Making, Pareto Optimality.

A formal comparison runs the risk of not catching the intent of the comparison. In CARE, the comparison is rather described as a discussion around the expected impacts of using a particular component (including both technical and non-technical concerns), PECA talks about "sound and careful reasoning", and PORE proposes argumentation techniques. Perhaps it is more fruitful to compare components by estimating the cost and risks of using them (both in the long and short term), which is probably what ultimately matters from a business perspective. We refer to [32] for a very thorough discussion, where it is argued that all of these evaluation methods ("decision-making techniques") have their drawbacks when applied to component selection. 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. It is proposed that gap analysis is more appropriate, meaning that for each component, the gap between requirements and provided capabilities are analyzed, followed by estimating the costs of bridging the gap.

6.3. Evaluation attribute, Data collection, and Confidence

Concerning what attributes to be evaluated and how to perform the actual data collection one can discern two phases in most methods. Sometimes these are described as explicit phases, sometimes they are the consequence of iteration and refinement. We label these two phases *high-level evaluation* and *prototyping evaluation.*

- 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. Not only the technical characteristics are evaluated but also business considerations such as the available support for the component, the vendor's reputation and financial stability. 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). One should focus efforts on gathering information that can discriminate between components. Criteria should be selected for which data are easy to find; in one of our cases some ninety components were evaluated for five minutes each by searching the vendors' web pages (leaving some twenty-five components for further, still high-level evaluation).
- 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. Experiments and prototypes are created 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 (PORE, RCPEP, i-MATE, PECA, *CCCS*).

The main distinction between the two evaluation phases is whether the component needs to be available during the evaluation or not, which have a big impact on the cost and time (and skills) required for the

evaluation and consequently limits the number of components that can practically be evaluated. In prototyping, the acquisition of a component may come with a (high) cost and can introduce (substantial) effort in the evaluation process, and the evaluation itself requires learning the component and systematically setting up, executing, and documenting many tests thoroughly.

6.4. Components and Candidates

Most methods consider evaluation and comparison of single components of the same kind (e.g. selecting a database or selecting a graphics package). However, when an envisioned system will be built using several COTS components (say, a database and a graphics package and a web server and a communication server…) it makes sense to treat combinations of the available components together as a single candidate (e.g. the Apache web server and the MySQL database go well together would be treated together as one candidate).

At the previously mentioned SEI workshop [1], a division into three types of COTS evaluation were characterized: *progressive filtering*, *puzzle assembly*, and *keystone identification*. As seen in the surveyed methods, these are not mutually excluding each other, but it is possible to trace elements of all three in the surveyed methods:

Progressive filtering is, as have been described, part of all of the surveyed COTS assessment processes.

Puzzle assembly means to simultaneously select several components needed in the system that fit together (to minimize architectural mismatch [33]). This idea has been implemented explicitly in *CSCC* (by comparing the estimated total system cost using various component alternatives) and *CCCS* (by explicitly considering a "candidate" to be a set of components which are architecturally compatible). Also PORE acknowledges that "the scope of the product under evaluation is difficult to define" [11].

Keystone identification 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 backbone from a certain vendor and then choose related products"). None of the surveyed method implements this strategy explicitly; i-MATE is probably closest, as it focuses solely on selection of middleware (which can be seen as a keystone for a particular class of systems). Treating component sets together as described above could be a starting point for keystone identification ("let us choose Apache and MySQL and then find other smaller components for e.g. graphics that are designed to work in this environment").

It may seem surprising that of these three approaches identified ten years ago by a major research institute together with industry, only one has had a great impact on the methods published after that (at least when simply counting the number of methods implementing each strategy).

7. Discussion

7.1. Option to Build

Many of the surveyed methods are designed exclusively to select a pre-existing component. However, in case no suitable component is found there should be the option to build a component (at least in principle, depending on the type of component the associated costs and risks may be far too large). This is included explicitly only in a few of the methods (CAP, *RDR*, *CCCS*, the build option is also mentioned by e.g. CRE). The "gap analysis" mentioned in section 6.2 could be a simple way to treat the build alternative in the same way as existing components: one would estimate effort, cost (both in the long and short term), risk etc. of each candidate as usual, and also include the candidate "new component" (where missing features are built rather than adapted). How to estimate the cost of the "build" alternative is another issue not discussed further here.

7.2. Customizability of Existing Processes

Many methods state that they are customizable, but there are typically no guidelines on how to customize them. Often, the methods are not customizable in any other sense than that they require that many details are filled when applied to a particular project. There is nothing to be found e.g. how to remove or add activities, or different ways of carrying out activities depending on the size of the project, the number of components expected or found, etc. PORE and PECA seem to be designed to meet this need, by allowing for opportunistic rearrangement of activities depending on what is most reasonable at any point in time during a project. i-MATE is the only method that explicitly intends to be less general, by providing substantial guidelines on a particular domain (middleware selection).

7.3. Domains

For safety-critical systems, there seems to be consensus that the suitable development model is a specification-first, plan-driven, waterfall-like process, and that iterative, evolutionary and agile processes are more feasible for e-business, web development, and

entertainment systems [34]. There seems to be a tendency that organizations and domains used to waterfall-like development also suggest a plan-driven component selection method (RCPEP, *RDR*, *CSSP*), however we see no inherent reason for this. On the contrary, our complementary interviews show that component assessment and selection can also for safety-critical and mission-critical systems be iterative and flexible. The reason for this is that component selection process can (with advantage) be seen as part of the requirements and design phases, which may be part of a very formalized and plan-driven process.

8. Summary and Conclusion

In this paper we have surveyed the published software component selection methods. We have provided a meta-model that captures the different parts of these methods, hence providing a common terminology and comparison framework for selection methods. We then continued by describing the collected best practices, thus leveraging the hard-earned efforts of many researchers and practitioners in the field over the years.

This paper and the meta-model would be the starting point for someone designing a selection process within a particular organization or development project. Based on the comparisons made in this paper, it is possible to select a single method which seems suitable for the context at hand, e.g. intertwined with requirements engineering (such as PORE, CRE, CARE). Either a method can be chosen where the layout of activities is described as sequential (RCPEP, *CSCC*, i-MATE, *RDR*, CEP, CARE, *CSSP*), iterative (STACE, CRE, *CCCS*), or a more opportunistic or parallel ordering of activities or interrelated process (OTSO, PORE, CAP, PECA, CPF). Even if the rest of the development process is designed to be waterfall-like (which is typical and usually advisable for critical systems), it should not be taken for granted that the selection process also needs to be sequential.

It is also perfectly possible to use one of the selection methods as a basis, and modify it to also take into account good practices from the other methods. In the future, we expect to carry out case studies which will provide more detailed advice how to carry out this customization. The meta-model can also be a starting point for the creation of novel methods, which may even differ so much from our classification so as to challenge the meta-model itself. We are currently studying the extent to which the principles of agile development support component-based development processes.

8.1. Acknowledgements

This work is partly funded by the Swedish Foundation for Strategic Research. The authors would like to thank the interviewees for openly sharing their experiences, unfortunately, we can not name all interviewees out of consideration for the companies they work for. We would like to thank Eoin Woods and Tim Trew.

9. References

[1] Patricia Oberndorf, Lisa Brownsword, Ed Morris, and Carol Sledge, “Workshop on COTS-Based Systems”, Special report CMU/SEI-97-SR-019, Software Engineering Institute, 1997

[2] Günther Ruhe, “Intelligent Support for Selection of COTS Products”, *Web, Web-Services, and Database Systems: NODe 2002, Web- and Database-Related Workshops*, Erfurt, Germany, 2002. Revised Papers, Lecture Notes In Computer Science 2593, Springer 2003

[3] Fredy Navarrete, Pere Botella, and Xavier 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] Carina Alves and Jaelson Castro, “CRE: a systematic method for COTS components Selection”, Proceedings of the *XV Brazilian Symposium on Software Engineering (SBES)* Rio de Janeiro, 2001

[5] Ivica Crnkovic, Michel Chaudron, and Stig Larsson, “Component-based Development Process and Component Lifecycle”, *Journal of Computing and Information Technology*, Volume 13, Issue 4, pp. 321-327, 2005

[6] Barbara Kitchenham, “Procedures for Performing Systematic Reviews”, TR/SE0401, Keele University, 2004

[7] Nenad Medvidovic and Richard N. Taylor, “A Classification and Comparison Framework for Software Architecture Description Languages”, *IEEE Transactions on Software Engineering*, Volume 26, Issue 1, January 2000

[8] Laurens Blankers, “Techniques and Processes for Assessing Compatibility of Third-Party Software Components”, M.Sc. Thesis, Eindhoven University of Technology and Mälardalen University, 2006

[9] Rikard Land, Stig Larsson, and Ivica Crnkovic, “Interviews on Software Integration”, MRTC report ISSN 1404-3041 ISRN MDH-MRTC-177/2005-1-SE, Mälardalen Real-Time Research Centre, Mälardalen University, 2005

[10] Jyrki Kontio, “OTSO: A Systematic Process for Reusable Software Component Selection”, University of Maryland report CS-TR-3478, UMIACS-TR-95-63, December 1995

[11] Neil A. Maiden and Cornelius Ncube, “Acquiring COTS Software Selection Requirements”, *IEEE Software*, Volume 15, Issue 2, pp. 46-56, March 1998

[12] Cornelius Ncube and Neil 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, CA, USA, 1999.

[13] Neil A. Maiden, H. Kim, and Cornelius Ncube, "Rethinking Process Guidance for Selecting Software Components", Proceedings of the *First International Conference on COTS-Based Software Systems*, Lecture Notes In Computer Science, Vol. 2255, pp. 151-164, Springer-Verlag, 2002

[14] Douglas Kunda and Laurence Brooks, "Applying Social-Technical Approach For Cots Selection", Proceedings of the *4th UKAIS Conference*, University of York, McGraw Hill, 1999

[15] 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

[16] Patricia K. Lawlis, Kathryn E. Mark, Deborah A. Thomas, and Terry Courtheyn, "A Formal Process for Evaluating COTS Software Products", *IEEE Computer*, Volume 34, Issue 5, 2001

[17] 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

[18] Anna Liu and Ian Gorton, "Accelerating COTS Middleware Acquisition: The i-Mate Process", *IEEE Software*, Volume 20, Issue 2, pp. 72-79, March 2003

[19] Santiago Comella-Dorda, John Dean, Edwin Morris, and Patricia Oberndorf, "A Process for COTS Software Product Evaluation", Proceedings of the *1st International Conference on COTS-Based Software System*, Orlando, Florida, Vol. 2255, pp. 86-96, Springer, 2002

[20] 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, pp. 189-199, Elsevier, 2002

[21] X. Burgués, C. Estay, X. Franch, J. A. Pastor, C. Quer, "Combined Selection of COTS Components", Proceedings of the *First International Conference on COTS-Based Software Systems*, Lecture Notes In Computer Science, Vol. 2255, pp. 54-64, Springer, 2002.

[22] Barbara Cavanaugh Phillips and Susan M. Polen, "Add Decision Analysis to Your COTS Selection Process"Software Technology Support Center Crosstalk, April 2002

[23] L. Chung and K. Cooper, "Defining Goals in a COTS-Aware Requirements Engineering Approach", *Systems Engineering* Volume 7, Issue 1, pp. 61-83, Wiley, 2004

[24] Lawrence Chung and Kendra Cooper, "COTS-Aware Requirements Engineering and Software Architecting", Proceedings of the *4th International Workshop on System/Software Architectures (IWSSA)*, 2004

[25] Jesal Bhuta and Barry Boehm, "A Method for Compatible COTS Component Selection", Proceedings of the *4th International Conference on COTS-Based Software Systems*, Spain, LNCS, Vol. 3412, Springer, 2005

[26] Alejandra Cechich and Mario Piattini, "Filtering COTS Components Through an Improvement-Based Process", Proceedings of the *4th International Conference on COTS-Based Software Systems*, Spain, LNCS, Vol. 3412, Springer, 2005

[27] Han Lin, Anh Lai, Rebecca Ullrich, Michal Kuca, Jessica Shaffer-Gant, Sandra Pacheco, Karen Dalton, Kelly McClelland, William Watkins, and Soheil Khajenoori, "COTS Software Selection Process", SANDIA REPORT SAND2006-0478, Sandia National Laboratories, May 2006

[28] Marco Torchiano and Maurizio Morisio, "Overlooked aspects of COTS-Based Development", *IEEE Software*, Volume 21, Issue 2, IEEE, March/April 2004

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

[30] 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

[31] Chris Abts, Barry W. Boehm, and Elizabeth Bailey Clark, "COCOTS: A COTS Software Integration Lifecycle Cost Model Model Overview and Preliminary Data Collection Findings", 2000

[32] Cornelius Ncube and John C. Dean, "The Limitations of Current Decision-Making Techniques in the Procurement of COTS Software Components", Proceedings of the *First International Conference on COTS-Based Software Systems*, LNCS, Vol. 2255, pp. 176-187, Springer-Verlag, 2002

[33] David Garlan, Robert Allen, and John Ockerbloom, "Architectural Mismatch: Why Reuse is so Hard", *IEEE Software*, Volume 12, Issue 6, 1995

[34] Barry Boehm and Richard Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, ISBN 0321186125, Addison-Wesley Professional, 2003