Software components selection process: comparative study

Mebarka Yahlali (yahlali.info@gmail.com)
Université de Saida Dr.Moulay Tahar

Research Article

Keywords: Software Components Selection, CBSE, Quality Model, MCDM, OTSO, PORE, CRE, SCEP, SCAEP

Posted Date: September 27th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2082229/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Abstract

Component-Based Software Engineering (CBSE) paradigm aims to develop software by assembling and deploying reusable units, called software components. This approach tries to improve the flexibility, reusability, and maintainability of applications, and helps develop complex and distributed applications deployed on a wide range of platforms, by plugging commercial off-the-shelf (COTS) components, rather than building them from scratch.

In this context, the selection step is very important. It consists of searching and selecting appropriate software components from a set of candidate components in order to satisfy the developer-specific requirements.

In the selection process, both functional and non-functional requirements are generally considered. Several component selection processes have been proposed, with the aim of finding and selecting an appropriate software component from a set of candidate components to satisfy the developer's specific requirements.

1. Introduction

Component-Based Software Engineering represents a new development paradigm: ‘assembling software systems from components’. The main objective is to build applications by assembling pre-existing software components. CBSE can significantly reduce development costs and improve the maintainability, reliability, and overall quality of software systems.

Therefore, CBSE distinguishes two life cycles: one for component development (Design for reuse) and another for the development of a component-based system (Design by reuse). In the first life_cycle, reusability is the main challenge, the components are created to be reused in different applications. When building a new component, developers can use other components: Design by reuse.

However, making reusability effective poses problems such as finding components. In front of a library containing a large number of components, it is necessary to have an automatic research mechanism that allows to find the most relevant components.

In order to reduce the adaptation effort, the selection of components must be relevant, i.e. the selected components must be very close to the components we are looking for.

Consequently, a bad selection can have a major impact on project costs and expected results, which can lead to considerable losses. So, we find that the success or failure of a project depends largely on a correct evaluation of the software components during the selection step. This paper presents a comparative study of four selection processes: OTSO (Off-The-Shelf-Option), PORE (Procurement-Oriented Requirements Engineering), CRE (COTS-based Requirements Engineering) and SCEP (Software Component Evaluation Process). The organization of the paper is as follows: section two gives a general
overview of the principle of software component research, the third section presents the techniques of component research, in the fourth section, we have summarized the software components selection process and finally we present a comparative study of this process.

2. Software Component Research

According to [1]. The problem of research components is formulated as follows:

The research model is composed into five areas:

- **Problem space**: represents a set of needs of a particular application, expressed informally.

- **Problem space as understood by reuser understanding**: after interpretation of each problem by the user (Problem understanding), a modified problem space occurs. For example, all services that can meet the needs expressed in the previous space.

- **Query space**: For each service identified by the user, it is now necessary to make a request.

- **Components space**: the component space or the library where the research will be performed.

- **Codes space**: Before the research, the components must be translated in order to obtain a set of descriptions called indexes space. Each description must be of the same format as the query so that the comparison can be made.

2. Software Quality Model

Quality model is a set of characteristics and the relationships between them which provide the basis for specifying and evaluating quality requirements [2].

There are more than 300 developed standards and maintained by more than 50 different organisations [3]. The first quality model considered as a standard was developed and published by the International Standardization Organization in 1991 as ISO 9126 [4]. Ten years later new edition of this standard reviewing quality model and introducing three perspectives: quality in use, external quality and internal quality. In the same time, new international initiative Software product Quality Requirements and Evaluation (Square) was set up aiming to develop set of norms ISO/IEC 25000 [5,6]. This new approach is perceived as new generation of software quality models (cf. Table 1).

Table1.Square quality model [7]
3. Software Component Research Techniques

3.1. External classification techniques

**Objective:** index the components from an abstract representation:

1. **Keywords research:** This is the simplest external classification technique. The principle is to associate a set of keywords to the component. Components research by keywords showed its limits for different reasons: when the library size is important (big library), the system can return too many components with the same keywords. The second limitation concerns the components field: the same keyword can be used in two different domains with two different meanings [8].

2. **Facet research:** A facet represents a particular piece of information that identify and characterize a component. It is defined by its name and its vocabulary (set of key words used to describe it). To describe a component, one or more keywords must be chosen from the vocabulary of each facet [8]. The facet classification approach has been used in several component search systems such as [9,4]. This technique has reduced the difficulties raised previously and gives good results on the conditions:

   - the facets are well chosen,
   - the components indexed correctly by giving the right terms
   - and properly defining the space of the terms.

Faceted classification requires manual indexing which is often time consuming.

3. **Natural language:** In [8] Maarek, Berry and Kaiser proposed an approach based on textual information retrieval techniques on natural language component descriptions. The querying of the
component database is done through queries in natural language. Each textual description of a component is analyzed for the extraction of a set of indexing terms. These terms constitute the descriptor which will be used for the correspondence with the user requests. Indexing is based on a lexical, syntactic and semantic analysis of component descriptions. The automatic interpretation mechanism used for component analysis uses linguistic techniques to search for descriptions. This method can present good results, but it remains difficult to implement [10].

3.2. Structural classification techniques

External classification techniques are based on the components descriptions (documentation). While structural research techniques are concerned with the structural aspect of the components.

This category is divided into two sub-categories: signature matching techniques and specification matching.

1. **Signature Matching Techniques**: The signature of a component is the union of the signatures of all the interfaces that it defines. Similarly, the signature of an interface is the union of the signatures of the operations that it declares. These techniques, consider a component as a service provider through its operations. Therefore, the requests are made on the signatures of these operations. For example in [3] Zaremski and Wing propose a matching process of signature based on the following type definition: A type is either a TypeVar variable or an TypeOp type operator. This operator can be either a predefined operator (BuiltInOp) or a user-defined operator (UserOp). Starting from this definition, the equality of type (= T) is defined as follows: two types t and t’ are equal if they are lexically identical variables, where if they are operators of type Same parameters and the same result.

2. **Specification matching**: Component specification informs not on its syntax, but on its behavior. These specifications can be expressed in logical form or by contracts (invariants, pre- and post-conditions) [8]. Component search by specification matching attempt to retrieve the components of the database whose specifications match the specification of the query [10]. Among the works that have been proposed in this context:

- Matching specification Zaremski and Wing [4]: Each operation is associated with a set of pre- and post-conditions that represent its behavior.

The matching specification is defined by the following formula: Match \( (S, Q) = (Q_{\text{Pre}} R_1 S_{\text{Pre}}) R_2 (S_{\text{Post}} R_3 Q_{\text{Post}}) \)

\( \text{Or} \quad S = (S_{\text{Pre}}, S_{\text{Post}}) \) is the specification of a component,

\( Q = (Q_{\text{Pre}}, Q_{\text{Post}}) \) is the specification of a query,

\( R_1, R_2 \) and \( R_3 \) are logical connectors.
In [20] Rollins and Wing present a basis of components using a classification-by-specification approach. The components are specified in \( \lambda \)-Prolog. This technique exploits the inference mechanisms of the \( \lambda \)-Prolog language for specification matching. Only those components whose specifications are relevant to the query specification are selected.

Finally, Hemer and Lindsay propose in [20] a basis of modules. A module is a set of units. A unit can be a procedure, class, or data structure. If a developer needs a module that implements all operations on the linked lists then it must specify all the primitives it needs as units, Then it looks in the database, the module that contains these units. The authors propose three strategies for searching for modules:

1. **ALL-match**: Checks that all units in the query match separate units of the component.
2. **SOME-match**: relaxes the ALL-match criterion by checking that at least one non-empty subset of units of the query corresponds to a non-empty subset of units of the component.
3. **ONE-match criterion**: verifies that at least one of the units of the request corresponds to one of the units of the component.

### 3.3 Behavioral Research Techniques

These approaches are concerned with the dynamic aspect of the components by analyzing their behavior during execution [10].

1. **Execution traces analysis approaches**: Among the works that have been interested in this approach are those of Podgursky and Pierce [11] [12]. After a statistical observation, the authors deduced that a component belonging to a component base can be identified based solely on its behavior with random input parameters. Based on this observation, they constructed a base of components that has the following properties:

   - A component is represented by its executable code and description of its input parameters. An input parameter is defined by its type and a probabilistic distribution of the values it can take relative to its definition domain. The probabilistic distribution makes it possible to estimate the relevance of the choice of a value with respect to its occurrence during the use of the component.
   - The query consists of two parts: the space of the desired input values and a condition that determines whether the response of a component is satisfactory with respect to the user's needs.
   - The operation of comparing the request and a component of the database is done in three steps: randomly selecting a number of input values based on the parameters of the request and the description of the component in The basis, application of Input values on all the components of the database, selection of the only components that check the parameters of the request.

2. **Approaches by behavioral specification** In [13] the authors present a repository of object-oriented components using a behavioral selection technique of components. A component is represented in the database no longer by its executable code, but by semantic networks describing its behavior. The
query is compared to the components to select those that maximize the similarity function that is equal to the ratio of the cardinality of the common behaviors on the total cardinality of the query behaviors (Jaccard's coefficient). The advantage of this technique is that not only can it find simple components (a class), but also complex components (a class graph).

4. Software Component Selection

4.1. Multi-Criteria Decision Making (MCDM)

Multicriteria decision support methods are decision support tools developed around the 1960s[8]. MCDM is a discipline of operations research, which deals with decision problems in the presence of a certain number of criteria [23]. It is used to make a comparative judgment between heterogeneous projects or measures.

There are three main families of methods: methods based on multi-attribute utility theory, over-classification methods and interactive methods.

4.1.1. WSM (Weighted Scoring Method)

- Each evaluation criterion is associated with a weight which represents its importance.
- Each criterion-candidate pair is associated with a score which represents the candidate's ability to fulfill the criterion.
- The candidate with the highest total score is considered the best.
- The most common WSM formula uses addition as an aggregation function:

$$score_c = \sum_{j=1}^{n} (weight_j \ast score_{cj})$$

The major problems of this technique are

- the difficulty of defining a set of relevant evaluation criteria and then assigning them the right weights.
- the evaluation of local scores and the assignment of weights remain manual, which makes this technique tedious when faced with a large number of criteria and candidates.

4.1.2. AHP : Analytic Hierarchy Process
Compared to WSM, AHP helps describe need in an organized way. Firstly, it is necessary to define the main objective to be reached in order to make a decision on the selection of a candidate. The objective is decomposed into a hierarchical tree of criteria and sub-criteria, the leaves are the candidates to be evaluated. AHP allows to define and prioritize the various evaluation criteria with precision.

4.1.3. MAGIQ: Multi-Attribute Global Inference of Quality

MAGIQ was originally developed to validate method results (AHP). Although MAGIQ has not been subjected to extensive research, the technique has proven to be very useful in practice. The MAGIQ technique uses the ROC concept (Rank Order Centroids). Ranking centroids are a way to convert ranks (1st, 2nd, 3rd) into scores or weights which are numeric values. If $n$ is the number of attributes, the weight of the attribute $K$ is:

$$W(A_k) = \left(\sum_{i=k}^{n} \frac{1}{i}\right) \frac{1}{n}$$

4.2. Software component selection processes

Several efforts have been made during the last decade to model the selection process.

There is no commonly accepted method for components selection [28]. Nevertheless, the existing selection processes have in common a number of phases which are presented in [18] known as the General Components Selection Process (GCS) which is described as follows:

1. Step 1: The Functional and non-functional Requirements Specification Process: Define the evaluation criteria according to the system requirements.

2. Step 2: Software Component research Process:

Find components from repositories.

3. Step 3: the filtering process:

Filter search results based on requirements. This allows the definition of a list of promising candidate components that need to be evaluated in more detail.

4. Step 4: Evaluation Process:

Evaluate the candidate components of the shortlist to select the best candidate.
4.2.1. OTSO (Off-The-Shelf-Option)

This process is developed by J. Kontio. Figure 2 shows the main phases of the OTSO process [14] as well as the parameters and data used.

1. Phase N° 1: Definition of evaluation criteria. The first step consists in defining the evaluation criteria according to four parameters related to the application in which the candidate will be integrated: The first parameter (Requirement specification) relates to the specification of the needs of the application, this includes in particular the functional requirements. The second parameter (Design specification) concerns the specification of the overall architecture of the application. The third (Organizational characteristics) and the fourth parameter (Project plan) are the organizational and project constraints related to its development.

2. Phase N° 2: Search. The search in the selection process consists of identifying the potential.

3. Phase N° 3: Screening consists in filtering the most relevant candidates by means of basic criteria and certain information about these candidates (COTS Sources).

4. Phase N° 4: Evaluation, consists in evaluating in detail each candidate by means of more precise criteria.

5. Phase N° 5: Analysis of results. The final phase consists in analyzing the results of these evaluations according to the previously defined criteria.

A major disadvantage of the OTSO process is that it does not propose anything to specify the needs. Moreover, it is mainly the functional needs of the application that are taken into account, to the detriment of non-functional needs.

4.2.2. PORE (Procurement-Oriented Requirements Engineering)

Maiden and Ncube have developed the PORE method. Basically, PORE defines three types of needs: main functional needs, secondary functional needs, and non-functional needs. In order to satisfy these three types of needs, PORE uses an iterative process, decomposed into generic processes:

1. Identify Product: identifies potential candidates through market research or expertise.

2. Acquire Information: This process involves obtaining information about the customer requirements and the product requirements. The acquisition of information is done through use cases and knowledge engineering techniques.

3. Acquired Information Analysis: consists in analyzing the information acquired to produce data that can be used in the following process.

4. Decision Making: The purpose of this process is to determine whether the candidates meet the needs or not, using multi-criteria decision-making (MCDM) techniques.

5. Product selection: involves rejecting candidates that have been assessed as unsatisfactory by the previous process.
Once the selection is made, two choices are possible:

1. Either the operation is recommenced by returning to the first or second process to detail the needs and reduce the list of candidates,
2. Either we stop the selection when we consider that the query is satisfied. Each process can be repeated several times in case the information is insufficient.

The PORE process has some disadvantages, because it is sometimes difficult to know how to stop the iteration [16]. And in [17], the authors find it unclear how to use the needs in the evaluation process, as well as how to eliminate candidates

4.2.3. CRE (COTS-based Requirements Engineering)

CRE [17] is an approach similar to PORE, it has four iterative phases:

1. Phase N ° 1: Identification: Define objectives based on: user needs, application architecture, project objectives and limitations, component availability, and organization infrastructure.
2. Phase N ° 2: Description: In this phase, the evaluation criteria must be elaborated in detail, emphasizing the role of NFR (Non Functional Requirements). The CRE method uses the NFR Framework for representation and analysis of non-functional requirements. This Framework is a process-oriented approach where NFRs are explicitly represented as objectives to be achieved. These objectives will then be broken down into sub-objectives using a tree structure and logical operators connecting the parent node with its children.
3. Phase N ° 3: Evaluation: The decision to select a particular product COTS is based on the estimated cost compared to the analysis of the benefits of each COTS alternative. The CRE process suggests the use of cost models such as COnstructive COTS (COCOTS).

4.2.4. SCEP (Software Component Evaluation Process) :

In [25,26] two evaluation processes are proposed, the first concerns the component quality evaluation and the second the assembly evaluation. The evaluation process is based on two functions :

- **function of representation**: $\beta$

$E_i \beta e_{ij=1,...,n}$ means that the element $E_i$ is presented by the set $(e_1, e_2, e_3,...,e_n)$.

Two cases can be distinguished:

- $\forall c_i \in C \ [1], \ \forall a_j \in A \ [2] / c_i \beta a_j \Rightarrow$ each criterion $c_j$ is represented by a set of attributes.
- $\forall f_j \in F \ [3], \ \forall c_i \in C / f_j \beta c_i \Rightarrow$ each factor $f_j$ is represented by a set of criteria

**Evaluation function**: $\mathcal{E}$

If $E_i \beta e_{ij=1,...,n} \Rightarrow V(E_i) = \mathcal{E}(V(e_j))$
If the element $E_i$ is presented by the set of elements $e_{j/j=1,...,n}$ the quality value of $E_i$: $V(E_i)$ can be calculated based on the quality value of the elements $e_j$: $V(e_j)$.

**Process 1: SCEP (Software Component Evaluation Process)**

SCEP contains three main steps:

1. **Quality characteristics classification**: in this step, the user must classify the quality characteristics (factors) according to his needs.
2. **Weightings calculation**: in this step, the rows are converted into weighting using ROC concept.
3. **Evaluation**: the evaluation processes is based on the relation $\beta$ and the function

**Process 2: SCAEP (Software Component Assembly Evaluation Process)**

This process is based on component quality evaluation process. It comprises four steps:

1. Consists in fixing the desired quality value.
2. Construction of the application by assembling the suitable software components: to determine all possible compositions (all candidate systems $S_n/n = 1,...,g$, $g$ is the number of possible compositions).
3. Evaluate the provided quality (PQ) by each system then compare it with the desired one.

**5. Comparative study**

This part represents a comparative study of four software component selection processes. The following table (Table 2) shows if the selection process does not cover or provides partial or total coverage of the different phases of General Components Selection (GCS) process:

<table>
<thead>
<tr>
<th>GCS steps</th>
<th>Functional Requirements Specification</th>
<th>Non-Functional Requirements Specification</th>
<th>Filter search results</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORE</td>
<td>Total</td>
<td>/</td>
<td>Total</td>
<td>Total</td>
</tr>
<tr>
<td>OTSO</td>
<td>Total</td>
<td>/</td>
<td>Total</td>
<td>Total</td>
</tr>
<tr>
<td>CRE</td>
<td>Total</td>
<td>/</td>
<td>Total</td>
<td>Total</td>
</tr>
<tr>
<td>SCEP</td>
<td>Total</td>
<td>Total/ partial</td>
<td>Total</td>
<td>Total</td>
</tr>
</tbody>
</table>

**Table 2. Coverage of GCS main phases.**

The comparison shows that OTSO, PORE, CRE processes do not sufficiently deal with non-functional needs.
The SCEP Process allows a specification of functional and non-functional needs using SCM (Software Components MetaData) [26] which provides the information necessary for the selection and evaluation of components. SCM is designed into two parts:

- The first concerns the functional aspect: the developer must give a description of all the services provided and required, specifying key words.
- The second part covers the non-functional aspect.

Another characteristic common to all these methods is the parallelism between the specification of requirements and the selection and evaluation of products. Moreover, these processes are very often iterative. As requirements are collected, some candidates are rejected, new candidates appear and the analysis of these new candidates in turn reveals new requirements. This characteristic of parallelism and iteration is very important and even essential to any method of selecting software components.

Components are evaluated by a set of criteria that represent system requirements, goals, and constraints. Kontio et al [29] suggest defining the evaluation criteria hierarchically, where a set of abstract goals are progressively refined, based on factors such as software application requirements, application architecture, and existing product capabilities.

Three strategies can be followed to evaluate software components [30]:

1. **Progressive filtering**: this method requires running GCS process iteratively until a small number of components identified from which one or more can be selected for integration into the system. The process starts with a large set of software components, and then gradually defines discriminating criteria by successive iterations, the "less suitable" products are eliminated.

2. **Puzzle assembly**: implies that a software components based system requires the assembly of various components like pieces of a puzzle.

   This strategy considering the requirements of each product while simultaneously remembering the requirements of the other products in the puzzle.

3. **Keystone identification**: starts by identifying a key requirement, then searches for products that meet that fundamental requirement. This allows the possibility of eliminating a large set of candidates that do not meet the key requirement.

5.1. **Comparing software components selection process**

This section presents a survey and comparison of four software component selection processes: Table 3 compares these approaches in terms of the following criteria:

- **GCS**: Conformance to the GCS method (cf. Table 1 )
- **EVAL**: Evaluation strategy used.
1. **PF**: Progressive filtering,
2. **KS**: Keystone,
3. **PZ**: Puzzle assembly.

- **type of selection**:
  1. **SNG**: single selection.
  2. **MLT**: multiple selection.

Selecting a single software component focuses more on functional requirements than the non-functional ones and does not address the interoperability problem.

Usually, there is no single component satisfying all the defined requirements. Hence, it is necessary to have adequate methodologies for multiple software selection.

- **ADAPT**: Adaptability of the process to different contexts.
- **TOOL**: Disponibilité d’un support d’outils pour faciliter l’application de la démarche

<table>
<thead>
<tr>
<th>criterion</th>
<th>GCS</th>
<th>EVAL</th>
<th>SNG</th>
<th>MLT</th>
<th>ADAPT</th>
<th>TOOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PORE</td>
<td>~</td>
<td>PF</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>OTSO</td>
<td>~</td>
<td>PF</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CRE</td>
<td>~</td>
<td>PF</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SCEP/ SCAEP</td>
<td>✓</td>
<td>PF</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>~</td>
</tr>
</tbody>
</table>

✓ : fully satisfies the criterion
X : does not satisfy the criterion
~ : partially the criterion

Tableau 3: Comparing software components selection process

5. Conclusion

CBSE tries to improve the flexibility, re-usability and maintainability of applications, and helps develop complex applications deployed on a wide range of platforms, by plugging commercial off-the-shelf components, rather than building them from scratch. The selection step consists of searching and selecting appropriate software components from a set of candidate components in order to satisfy the developer-specific requirements.
In this paper we have presented the different mechanisms for searching and selecting software components. Existing selection processes share a number of phases that are presented in [18] as the General Selection Process (GCS).

We noted that the majority of selection methods did not adequately address non-functional needs, and the concept of assembly is absent: The selection processes do not deal with the case where the service will be provided by a component assembly.

The Process presented [25,26] respond to these needs and it facilitates the comparison and minimise the time to compare equivalent systems.

**Declarations**

**CONFLICT OF INTEREST**

Author declares that she has no conflict of interest.

**AUTHOR'S CONTRIBUTION**

The writing and preparation of the paper was carried out by YAHLALI.M

**FUNDING DECLARATION**

No funding

**References**


10. Oualid KHAYATI. Formal models and generic tools for the management and research of components, PhD thesis; National Polytechnic Institute of Grenoble. 2005


Figures

![Figure 1](image1)

Figure 1

Research components model [1]
Figure 2

Quality Model structure [2]

Figure 3

Selection Process OTSO [14]
Figure 4

function of representation $\beta$

Figure 5

Evaluation function: £