European Journal of Applied Sciences – Vol. 9, No. 5
Publication Date: October 25, 2021
DOI:10.14738/aivp.95.10772.
Ekanem, B. A., & Agbele, K. K. (2021). A Review of Software Components Identification Methods and Quality Assessment Criteria. European Journal of Applied Sciences, 9(5). 194-209.

A Review of Software Components Identification Methods and Quality Assessment Criteria

Bassey Asuquo Ekanem
Computer Science Department
Delta State Polytechnic, Ozoro, Nigeria

Kehinde K. Agbele
Mathematics and Computer Science Department
Elizade University, Ondo, Nigeria

ABSTRACT

Identification of suitable reusable software components from repositories and other sources is a key process in software reuse. Many methods for components identifying exist. However, considering the increased adoption of reuse in software projects and the need to identify appropriate components from the numerous sources based on multiple quality criteria, it becomes imperative to undertake a review of existing methods to determine their strengths and weaknesses as well as gaps in components identification. In view of the above, this research was designed to undertake a systematic review of components identification methods in software reuse. Research findings indicate that existing methods and automated tools were inadequate in handling components identification from multiple sources especially where multiple quality criteria are considered. Also, findings indicate that multi-objective optimization algorithms like non-dominated sorting genetic algorithm III (NSGA-III) are capable of addressing such multidimensional problems; therefore, greater efforts in designing and developing automated tools that are based on this technique are highly recommended. The need for adequate methods for cataloguing and categorizing components that are stored in repositories and other sources is also identified as capable of enhancing the process of components identification in software reuse.

Keywords: Software Reusability, Components Identification, Automated Tools, Quality Assessment Control

INTRODUCTION

The rapid changes in technologies used in the ecosystem to enhance the quality of life in a fast changing world has correspondingly affected the way software systems are developed and maintained. As new technologies emerge, the need to build suitable software systems to drive them as quickly as possible also unfolds. Being that modern software systems are more large-scale, complex and difficult to build from scratch, there is increased adoption of software reuse in modern software projects as a strategy to cope with the challenge [1], [2], [3], [4]. Software reuse offers great benefits in modern software projects because it leads to reduced development time and cost, quick time to market, enhanced quality of the developed software, leveraging on development knowledge and expertise of other professionals amongst others [2], [5], [6], [7].

In software reuse, quality and suitable reusable components must be identified and selected from available sources before they are integrated into the project to build software systems. Components identification is the process of recognizing that the required components exist by matching the characteristics of the required component with the classified characteristics of components stored in repositories. However, the tasks of identifying quality and suitable components for reuse is complex and difficult due to increased number of components and sources ranging from components repositories in the internet to legacy sources [2], [8], [9]. Most times, integrators are confused on what components to use considering the uncountable number of components from various sources in the Internet repositories [10], [11]. Also, the choice of identification method could impact negatively on the process if a wrong method is chosen [12].

Therefore, this research is a systematic review of software reuse with reference to components identification methods and quality criteria used in the process. The research is aimed at revealing the current state-of-the-art in this area of software engineering to guide practitioners on choice of suitable methods and to initiate thought provoking researches that could further enhance software reuse and reuse products.

RESEARCH METHODOLOGY

The methodology adopted for this research work is the Systematic Literature Review (SLR) methodology introduced by Kitchenham and Charters. The choice of this methodology was based on its ability to present a fair evaluation of a research topic using trustworthy, rigorous, and auditable process [13]. The methodology has three phases as outlined below:

Plan Phase

In this phase the following were undertaken: identification of the need for the review, commissioning the review, specifying the research questions, developing and evaluating the review protocol.

Conduct Phase

This stage has five activities explained below:

a) **Search Strategy:** This has to do with generating the search strategy and performing trial searches using different combinations of search terms derived from the research questions.
b) **Inclusion and Exclusion Criteria**: This has to do with specifying inclusion and exclusion criteria to be used in identifying relevant resources based on the research questions.
c) **Data collection:** At this stage, relevant data were collected based on specified inclusion and exclusion criteria. This was done with a search conducted on the Internet using Google search engine. The retrieved data were documented in a data collection form showing title, authors, publication years and other information.
d) **Information synthesis**: the extracted data were grouped into quantitative and qualitative results. The quantitative results were presented in tables and charts while qualitative data were summarized using sentential formats.

Report Phase
Finally, findings from the review were documented in a report that presents the current state-of-the-art of software reuse with respect to components identification and quality criteria for components identification.

The Need for the Review
Difficulty in identifying reusable software components for reuse remains the crucial problem of software reuse since there are multiple sources of components especially in the internet which contents are also increasing geometrically on daily basis [9]. Also, existence of various methods for components identification further complicates the matter since wrong choice of methods could impact the process negatively [2], [7], [14], [15]. Furthermore, inadequate information on the current state-of-the-art in components identification methods and quality criteria to be considered when identifying appropriate components for reuse in a key challenge for the process. Most times, software reuse professionals spent much time and resources trying to identify appropriate components due to inadequate information on the most suitable method hence wrong choice of method [1], [4], [16]. These prompted the need for this research to enhance the body of knowledge needed to effectively support the process.

A systematic review of the process is capable of revealing the state-of-the-art of the process and gaps that need to be addressed.

The Research Questions
A set of seven research questions were designed for the review as shown in table 1.

Table 1: Research Question for the Review

Question Number	Research Question
Q1	What are the benefits of reusable software components in software reuse?
Q2	What are the different sources of reusable software components?
Q3	What are the methods for classifying reusable components that are stored repositories?
Q4	What are the challenges in identifying suitable reusable components from available components sources?
Q5	What are the methods for identifying relevant reusable components from available components sources?
Q6	What are the quality criteria that influence components identification from sources?
Q7	What are the gaps in components identifying from available sources?

Inclusion and Exclusion Criteria
Inclusion and exclusion criteria were used in this research to determine what materials are relevant to the study? It serves to guide the reviewer on what materials should be included or excluded from the study. Inclusion and exclusion criteria for this review were specified as given below:

a) Inclusion Criteria

I1: The study investigates software reuse and methods for identifying reusable software component. A study that only investigates software reuse without methods for identifying reusable software component is not included in the review.
I2: The study mentions the sources of reusable software components.
I3: The study mentions the challenges in identifying relevant reusable components from available components sources
I4: The study mentions the criteria that influence components identification from sources.

It is not necessary for a study to satisfy all inclusion criteria to be included in the review. The rule for the inclusion criteria is expressed with Boolean operators thus: I1 AND (I2 OR I3 OR I4). This means that the criterion I1 is mandatory for any article to be included in addition to only one of I2 or I3 or I4 being satisfied.

b) Exclusion Criteria

At least one of the exclusion criteria is enough for a study to be excluded from the review. The rule for the exclusion criteria is expressed with Boolean operators thus: E1 OR E2 OR E3 OR E4:
E1: A study that does not meet any of the above mentioned inclusion criteria is to be excluded. For example, a study that investigates software reuse without considering methods for identifying reusable software components is to be excluded.
E2: A publication that is an editorial or a short paper without the basic sections of reporting scholarly research are excluded.
E3: Duplicate studies. A particular materials may appear at two or more locations as duplicate copies. Where this happens, duplicate copies of the article are excluded.
E4: Research papers published earlier than 2005. This is necessary to exclude resources that are too old as well as reduce the volume of resources to a reasonable and relevant materials for effective reviewed.

The Review Protocol

In this section, review protocols were specified. Data for the review were obtained from the internet using google search engine with the following search string:

"Software Reuse, benefits of software reuse, methods for identifying reusable software components, sources of reusable components, challenges of identifying appropriate software components, quality criteria of reusable components, after: 2004-12-31"

The retrieved data were validated by checking the displayed search results one after the other to ensure their contents conform to expected results. Search results not in conformity were excluded from the list. Also, the selection criteria used are articles from popular sources namely ACM Digital library, IEEE Xplore, Springer, Science Direct, Wiley Online Library, Web of Science, Scopus and Google Scholar.

Research Data Extraction

The google search engine was used to search and retrieve the data from the Internet by typing the search string given earlier. A total of 13,400 resources were retrieved but reduced to 105 by the search engine when similar and already displayed materials were omitted from the

search results. Furthermore, the extracted results were reduced to 66 following the application of inclusion and exclusion criteria which excluded articles that did not meet inclusion criteria.

RESULTS AND DISCUSSION

A careful review of the extracted data undertaken by reading through the contents revealed the following relevant facts presented in line with the research questions.

Benefits of Reusable Software Components in Software Reuse

This section provides answers to **Q1: What are the benefits of Reusable Software Components in Software Reuse?** All of the 66 reviewed articles maintain that reusable software components are of great benefits in software reuse, hence its increased adoption in software reuse. In [17], [18], [19] it is emphasized that reusable software components are classical solutions to developing large and complex software systems and are capable of reducing software development time and costs as well as increase the quality of the developed product.

Also, [2], [5] report that, increased inclination of most software developers towards software reuse is because the use of reusable components reduces project risks and costs especially where new technologies are required to develop software from scratch as well as promotes applications interoperability. When reusable components are correctly applied in software reuse, it could increase developers' productivity, shorten time-to-market, improve software quality and reduce software maintenance costs [4], [20], [21].

Sources of Reusable Software Components

The answers to **Q2: What are the different sources of reusable software components?** are provided thus. The reviewed articles indicate that there exist various sources of reusable software components as outlined below:

i. **In-house Components Development Project** – following the need for specific components in a software project, in-house development team could be constituted to develop new reusable components suitable for the project [1], [22], [23].
ii. **Legacy Systems** – old software system with quality reusable components can be examined with the intent of identifying and extracting the quality components for reuse in new projects [1], [24], [25]. The legacy system could be an open source software (OSS) or a legacy asset of the organization.
iii. **Components from new Development Projects** – in this case, as a new software is being developed, components are planned for and created from the software that is being developed for use in reuse projects [1], [26].
iv. **Components Repositories** – this is a well-structured storage of reusable components with adequate characterisation of components to facilitate easy storage, identification and retrieval of suitable and quality components for reuse [1], [26], [27].The repository can be on the Internet or in-house repository of the organization. In [22] DHIS2 Shared Component Platform, that facilitates component reuse within the DHIS2 platform ecosystem is designed, developed, evaluated and presented for use.
v. **COTS Sources** – this refer to vendors who offer reusable software components for sale in the form of COTS [1], [28]. In this case the internal structure and contents of the component is unavailable therefore making it difficult to assessed the quality of

the component save for quality rating specified by the developer or the vendor on the product.

Methods for Classifying Components Stored in Repositories

The way and manner reusable components are organized in repositories could affect the process of component identification. **Q3: What are the methods for classifying reusable components that are stored repositories?** Was designed to review the methods for components classification. In [29], the following four methods for components classification are presented:

i. Free Text Classification

In this method, keywords that are relevant to each components are carefully selected and used to automatically index the components stored in the repositories. The indexing process has to do with creating a list of search items for all components stored in available repositories to be used in components search as initiated by users. During a search, the user inputs the keywords that best describe the component and a list of ranked components that match the search criteria are displayed in return. This is the search technique adopted by search engines like Google, MSN Search, Yahoo and others. The disadvantage of this technique is that irrelevant components may form part of the search results since a component may be associated with many keywords.

ii. Enumerated Classification

This is a one dimensional classification scheme where components are classified based on a set of mutually exclusive classes with a provision for sub categorization at different levels enumerated within the main class. Being that it is one-dimensional, it cannot accommodate multiple characteristics of components, hence not flexible enough to yield the desired results from a search.

iii. Attribute Value Technique

This technique uses a set of attributes to classify a components. For instance, a component may be associated with attributes like name, classification code, author, function, size, last updated date and so forth. The setback of this technique is that, where a search is initiated with attributes that are not part of the components classification, the search result will be incorrect.

iv. Faceted Classification

In this case, facets are used to describe the components. Facets in this refer to group of descriptors like component's functionalities, implementation requirements and so forth. Faceted classification, goes with guided search because users are usually presented with options to select those that are relevant to the search being conceived. The method is very effective but highly labour intensive to classify components with relevant facets [29].

Challenges in Identifying Suitable Reusable Components from Sources

The following findings were made with respect to **Q4: What are the Challenges in identifying suitable reusable components from available components sources?** 100% of the reviewed articles maintain that even though software reuse is highly beneficial, it is not without challenges. Some of these challenges are outlined thus:

i. **Inadequate methods for cataloguing and classifying components that are stored in various repositories:** findings from the review indicate that existing methods are inadequate as they do not provide for multi-dimensional classification of components to accommodate both functional and non-functional characteristics of components, resulting in increased costs and efforts to identify components [1], [22], [30], [31], [32], [29].

ii. **Inadequate tools and methods for components Identification using multi-objective optimization Technique**: in [4], [16] it is stated that existing tools and methods are inadequate in using multiple quality criteria to identify components as they do not harness multi-objective optimization problem solving technique. Also, lack of adequate metrics for measuring most of the quality criteria encourages the use of manual methods which are characterized by low results accuracy [5], [4]. Furthermore, [33], [34] maintains that most components identification methods are theoretical and very difficult to comprehend, hence rarely used by component integrators especially for large system.

iii. **High cost of components identification due to searching of multiple sources:** As reported in [8], [11] component identification process usually involves searching various components repositories in the Internet to identify the desired components, a process that could be very expensive to undertake. This is also confirmed in [5], [35], [7], [36], [37]. High costs of searching for suitable glue code is another cost factor; and all of these are influenced by the complexity of the optimal selection problem which increases with increase in number of repositories, volume of components in each repository and quality attributes being considered [38], [39], [40], [41], [19].

iv. **Difficulty in meeting licensing and authorization requirements to access some components repositories.** Most component sources on the internet where quality components could be found are protected by some licensing and authorization policies that must be met before access is granted [2]. Meeting these requirements could be a bit difficult for some especially where many of such sources are involved.

Methods for Identifying Relevant Reusable Components from Components Repository

This section responds to **Q5: What are the methods for identifying relevant reusable components from available components sources?** In the review, about 84% of the reviewed articles commented on methods for identifying components from sources for reuse. These methods are outlined thus:

i. **Multi-objective Optimization Methods**

These methods seek to address components identification problem as a multi-objective optimization problem where multiple quality criteria are used. Findings indicate that this is a novel method and yet to be adequately explored [42]. In [4], a method that is based on non-dominated sorting genetic algorithm II (NSGA-II) is proposed for solving components identification problem using multi-objective optimization problem technique. The method uses three quality criteria namely cohesion, coupling and complexity to identify appropriate components. The use of non-dominated sorting genetic algorithm II (NSGA-III) is highly recommended for upcoming methods to allow for the use of multiple criteria in the optimization problem [42].

ii. Clustering Methods

In software reuse, clustering methods are highly adopted by practitioners because of its ability to reduce the search space and optimize search time [9], [43]. [44], [45]. This method uses high cohesion and low coupling strategy to identify components from available repositories. Different flavours of clustering methods exist. In [46], clustering methods that uses similarity function to identify component clusters and estimate project cost is presented. Also, in [47], clustering method that uses Gaussian function to count the features and evaluates features distribution in a component is presented. Another clustering method that is based on use cases in proposed in [48]. Unfortunately, clustering methods are not adequately automated with robust and user-friendly features hence require some manual processes to be fully implemented [9].

iii. Graph-based Method

In this methods, components identification problems is modelled as a graph, G =G(N,E) where N is the set of nodes and E is the set of edges. In [49] the method is used to address components identification problem by applying the heuristic from graph concepts where weighted graphs are used to identify mutually disjoint sub-graphs as components with high quality. The disadvantage of this approach is that weights are manually assigned to edges by a domain expert, who may not estimate the weights accurately.

iv. CRUD-based Method

CRUD is an acronym for four relationship types namely Create, Read, Update and Delete used to specify semantic relationship between elements that are presented in a matrix that can be used to solve problems. As presented in [4], CRUD method called COMO was presented decades ago for use in components identification based on use cases or class matrix created from use case diagrams or class diagrams respectively. Thereafter, a tool similar to COMO called O2BC that uses business events and domain objects as inputs to create the matrix was presented. However, this method has a number of limitations similar to Clustering due to the use of clustering method, hence not highly supported.

v. Heuristic/Metaheuristic Methods

Heuristic methods are designed for solving problems more quickly when classic methods are too slow, or used for finding approximate solution when classic methods fail to find any exact solution. In software reuse, this situation could arise depending on the complexity of the problem to solve [6]. Relying on heuristic concepts, search-based software engineering has been introduced to deal with such situations [9]. In [6], metaheuristic algorithms are presented as a tool for extracting components from existing software to build standard Java Bean reusable software components.

vi. Neural Network Method

In [39], a method named Neuro-fuzzy Inference engine that combines the features of both neural networks and Fuzzy algorithms to perform domain relevancy appraisal needed to identify quality reusable components from existing object-oriented systems is presented. Because neural network can be used to define fuzzy rules for the fuzzy inference system as well as discover relationships and pattern in the data, it can be used to pre-process data in the fuzzy system, hence the combination of the two.

vii. Formal Concept Analysis Method

Formal concept analysis (FCA) is a mathematical theory about concepts and concepts hierarchies. It explicitly formalizes the extension and intension of a concept, their mutual relationships and the fact that increasing intent implies decreasing extent and vice versa. FCA is also applied in components identification as in [50] where components stability is used as a key quality attribute together with dispersion and distance concept to replace cohesion and coupling concepts used in other methods. The disadvantage of these methods is that some key processes are manually undertaken by experts, a situation that could result in low accuracy, hence not adopted.

viii. Other Methods

There are some other methods that are not identified with any of the above, hence classified in this review as other methods and scribed thus. In [10], a method called Rate of Reusability Metric is presented that uses components reusability rate to create 'Matrix-of Reusability' that can be used in selecting best components from the repositories. Also, [3] presents a method that can be used to identify quality components from legacy software in three stages by assessing components structural properties, token-based semantic relationship and change history evolutionary relationships respectively.

A method called preference-based component identification using particle swarm optimization (PCI-PSO) is presented in [51] which use customized particle swarm optimization (PSO) technique to automatically classify classes in a software into suitable logical components based on software architect's preferences. In [11], a method that uses genetic algorithm to find the best optimal software reusable component from components repository is presented. JReuse is another method proposed in [52] that uses naming similarity of two object-oriented entities, namely classes and methods to identify quality components from repositories. Automated tool was also developed from the method and used to evaluate 38 e-commerce software from GitHub for quality components and confirmed to be effective and efficient. In [53], a method called Search-Based Logical Component Identification (SBLCI) that is based on genetic algorithm undertaken iteratively is presented.

Also, in [54], a method called Clone detection and program slicing is presented and structured into four phases namely, tagging the user query to class and method names, constructing pattern/s of code features using clone detection technique, searching components in the library based on the clone pattern/s and aid the comprehension of the retrieved code components with program slicing technique.

In summary, [22], [33] and [53] maintain that most of these methods rely on experts' opinion hence may not be as accurate as expected. Also, [8] and [15] report that most components identification methods are theoretical and very difficult to comprehend, hence rarely used by component integrators especially for large system. The use of approximation techniques like Analytical Hierarchy process (AHP) in most methods is another issue, hence the need for further researches to explore other methods that could yield accurate results.

E. Quality Criteria for reusable Components Ranking and Selecting

A quality criterion was also accessed using **Q6: What are the quality criteria that influence components identification from sources?** Following the review, about 96% of the reviewed

articles emphasized the importance of quality criteria in components identification. In [33], component quality model with 11 attributes including reusability, coupling, complexity, cohesion, interface documentation is presented. These and other attributes considered key in components identification are presented thus:

i. **Reusability:** the degree to which a software component can be reused in other software projects [6], [55], [14], [56], [57], [58]
ii. **Reliability:** this is the quality of reusable component that makes it robust, fault-tolerant and highly available [25], [12], [41], [7], [11], [19], [58].
iii. **Robustness:** the degree to which a component continues to function properly under abnormal conditions or circumstances during a given period of time [10], [7], [11], [56], [59].
iv. **Testability:** this has to do with the ability of a component to facilitate the creation and execution of successful tests [8]**,** [10], [60], [58].
v. **Availability:** this attribute defines the proportion of time that a component that the component is available to perform useful work [6]**,** [14]., [33], [8]**,** [56]**,** [61].
vi. **Capacity:** the maximum number of entities that a component can successfully handle at a single point in time. This could be in terms of the number of users, simultaneous transactions, records and so forth Pandey and [8], [55]**,** [58], [61].
vii. **Usability:** This attribute defines the ease with which a components can be used by developers to develop new software systems. This could be assessed in terms user-friendliness of the component interface and reference support [55]**,** [56], [58].
viii. **Maintainability:** This is degree to which a software component **can** be modified to correct faults, improve performance or or adapt to a changed environment. This also addresses adaptability and extensibility as well [6], [55], [14], [19], [41], [7], [11], [59].
ix. **Portability:** quality of a component that minimizes efforts required to support it across heterogeneous operating system platforms, programming languages and compilers [3], [28], [55], 2018), [14], [56].
x. **Stability:** quality of reusable component that over the years from one version to another has gone through minimal number of modifications yet still working perfectly according to specifications [6]**,** [28], [62], [63], [7], [56]
xi. **Interoperability:** this refers to the component's ability to successfully run, communicate and exchange information with other components or systems outside its domain [28], [56], [59].
xii. **Cohesion:** this is the degree to which the constituent parts in a component belong together [8], [6], [14]**.** Also, components with high cohesion tend to be preferable, because high cohesion enhances other attributes like robustness, reliability, reusability, and understandability [55], [33].
xiii. **Coupling:** this is the degree of interdependence between the constituent parts in a component. **It is** a measure of how closely connected the elements are. Components with low coupling are preferable because low coupling will impact positively on other attributes [8], [6]**,** [55], [8], [33].
xiv. **Complexity:** this is the measure of the level of interaction between the different parts of the components. This attribute is determines by coupling and cohesion. Low coupling and high cohesion are indicator for acceptable complexity [8], [6]**,** [55], [8], [33].
xv. **Documentation level:** this attribute refers to the quality of a component that provides useful reference materials about the component and its interface; how to use it, identify

and resolve issues that could arise in course of using it [37], [19], [41], [7], [11], [25], [12], [59], [56].

xvi. **cost of integration:** this describes the cost of utilizing the component in the development projects which include cost identifying, selecting or extracting the component as well as integrating it to develop the new [41], [7], [11], [25], [12], [59], [56].

Gaps in Components Identification from Available Sources

This section responds to **Q7: What are the gaps in components identifying from available sources?** The gaps from the review are outlined below:

- **i.** The need for methods with multi-dimensional approach in cataloguing and classifying software components that are stored in components repositories using both functional and non-functional characteristics of reusable components is emphasized as existing methods are inadequate in this regard [6], [31], [30], [64], [32], [65].
- **ii.** Also, the need for automated tools and methods that utilize multi-objective optimization algorithm in solving components identification problems to enable the use of multiple quality criteria in components identification problems is identified [5], [6], [13], [5], [16], [11], [66].
- **iii.** The need for adequate metrics for measuring quality attributes of reusable software components which values could be used in effective an efficient cataloguing and classifying of components to minimize efforts, time and costs of searching available sources to identify components is also identified [2], [5], [63], [60].
- **iv. The need for Unified platforms to support** software reuse with respect to components identification, selection and integration to enhance the quality of software reuse process and reuse products [2], [67], **[58].**

CONCLUSION

Increased adoption of software reuse in software projects has further emphasized the importance of quality reusable software components in modern software projects. Considering the numerous sources of reusable components especially on the Internet and the geometric increase of components in such repositories on daily basis, it becomes imperative that methods for identifying components from these repositories are reviewed with the intension of evolving methods, techniques and tools that will enable integrators identify components from repositories with minimal time and costs since existing methods are inadequate in handling multidimensional components identification problems.

The use multi-objective optimization algorithms like non-dominated sorting genetic algorithm III (NSGA-III) are presented as novel approach to addressing such multidimensional problems. Therefore, greater efforts in designing and developing automated tools that are based on this technique are highly recommended to optimize the problem solution. Furthermore, the need for adequate methods for cataloguing and categorizing components that are stored in repositories and other sources is also identified. With these, the process of components identification using multiple quality criteria will be enhanced.

References

[1] Al-Badareen, A. B., Saleh, A. M., Abuaddous, H. Y. and Enaizan, O. (2021). In the Process of Software Development: Available Resources and Applicable Scenarios; Journal of Theoretical and Applied Information Technology; 99(1)

http://www.jatit.org/volumes/Vol99No1/3Vol99No1.pdf

[2] Saeed, T. (2020). Current Issues in Software Re-usability: A Critical Review of the Methodological & Legal Issues; Journal of Software Engineering and Application 3(9)

https://www.scirp.org/journal/paperinformation.aspx?paperid=103273

[3] Rathee, A. and Chhabra, J. K. (2019). Sensitivity Analysis of Evolutionary Algorithm for Reusable Software Component Identification; Mendel Soft Computing Journal, 25(1)

https://doi.org/10.13164/mendel.2019.1.031

[4] Gholamshahi, S. and Hasheminejad, S. M. H. (2020). Software component identification and selection: A research review; Journal of Software Practice and Experience; 49(1), pp.40-69, Wiley Online Library https://onlinelibrary.wiley.com/doi/10.1002/spe.2656

[5] Imoize, A. L., Idowu, D. and Bolaji, T. (2019). A Brief Overview of Software Reuse and Metrics in Software Engineering; World Scientific News, An International Science Journal

https://www.worldscientificnews.com

[6] Rathee, A. and Chhabra, J. K. (2020) Mining Reusable Software Components from Object-Oriented Source Code using Discrete PSO and Modeling Them as Java Beans; Information Systems Frontiers, Springer, vol. 22(6), pages 1519-1537, December.

https://ideas.repec.org/a/spr/infosf/v22y2020i6d10.1007_s10796-019-09948-4.html

[7] Garga, R. (2017). A systematic review of COTS evaluation and selection approaches; Growing Science;http://www.m.growingscience.com/ac/Vol3/ac_2017_3.pdf

[8] Pandey and Tulika (2019). Review for Different Approaches for identify a Software Component, Journal of Emerging Technologies and Innovative Research (JETIR, 6(5), 2019

https://www.jetir.org/papers/JETIR1905K19.pdf

[9] Gholamshahi, S. and Hasheminejad, S. M. H. (2018). A method for identifying software components based on Non-dominated Sorting Genetic Algorithm; Soft Computing Journal (SCJ), University of Kashan, 2018; 7 (2) :47-64
http://scj.kashanu.ac.ir/article-1-521-en.html

[10] Tiwari, U. K. And Kumar, S. (2019). Selection of Components Using Rate-Of-Reusability And Matrix-Of-Reusability; Journal of Engineering Science and Technology Vol. 14, No. 2 (2019); School of Engineering, Taylor's University pp. 659 - 673

https://jestec.taylors.edu.my/Vol%2014%20issue%202%20April%202019/14_2_11.pdf

[11] Islam (2017). An Effective Approach for Evaluation and Selection of Component; Journal of Computer Engineering and Intelligent Systems; Vol.8, No.1, 2017 www.iiste.org

[12] Alrubaee, A. U., Cetinkaya, D., Liebchen, G. and Dogan, H. (2020). A Process Model for Component-Based Model-Driven Software Development, MDPI

https://www.mdpi.com/2078-2489/11/6/302/pdf

[13] Kitchenham, B. and Charters, S. (2007). Guidelines for Performing Systematic Literature Reviews in Software Engineering; Keele University and Durham University Joint Report, (2007)

https://userpages.uni-koblenz.de/~laemmel/esecourse/slides/slr.pdf

[14] Ekanem, B. A (2017). Quality Indicators for Effective Component-Based Software Application; International Journal of Computer Science Trends and Technology (IJCST) – Volume 5 Issue 6, Nov - Dec 2017

https://www.academia.edu/35339806/_IJCST-V5I6P12_Ekanem_Bassey_Asuquo

[15] Pande, J. (2012). On Some Critical Issues in Component Selection in Component based Software Development; International Journal of Computer Applications (0975 – 8887) Volume 46– No.4, May 2012

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.735.3500&rep=rep1&type=pdf

[16] Goyal, S. and Parashar, A. (2017). Selecting the COTS Components Using Ad-hoc Approach; I.J. Wireless and Microwave Technologies, Modern Education and Computer Science Press, vol. 5 pp. 22-31http://mecs-press.org/ijwmt/ijwmt-v7-n5/IJWMT-V7-N5-3.pdf

[17] Dirnfeld, R. (2021). Reuse in Self-Adaptive Software Systems A Literature Review; Master Thesis Project, Linneuniversitetet, Kalmar Vaxjo https://www.diva-portal.org/smash/get/diva2:1573734/FULLTEXT01.pdf

[18] Anasuodei, M. and Akpofure, O. N. (2021). Software Reusability: Approaches and Challenges; International Journal of Research and Innovation in Applied Science (IJRIAS) |Volume VI, Issue V, May 2021

https://www.rsisinternational.org/journals/ijrias/DigitalLibrary/volume-6-issue-5/142-146.pdf

[19] Kaur, I. and Kaur, S. (2015). Efficient ranking and Retrieval of reusable software Components; International Journal of Applied Engineering Research 10(15):35354-35361

https://www.researchgate.net/publication/283762688_Efficient_ranking_and_retrieval_of_reusable_software_components

[20] Kaur, P. S., Singh, J. and Singh, H. (2014). Component Selection Repository with Risk Identification; Conference: 2014 International Conference on Computing for Sustainable Global Development (INDIACom)

https://www.researchgate.net/publication/269273645_Component_Selection_Repository_with_Risk_Identification

[21] Bhatia, R., Dave, M. and Joshi, R. C. (2006). Retrieval of Most Relevant Reusable Component Using Genetic Algorithms. Conference: Proceedings of the International Conference on Software Engineering Research and Practice & Conference on Programming Languages and Compilers, SERP 2006, Las Vegas, Nevada, USA, June 26-29, 2006, Volume 1

https://www.researchgate.net/publication/221610702_Retrieval_of_Most_Relevant_Reusable_Component_Using_Genetic_Algorithms

[22] Bengtsson, A. (2021). Facilitating Software Reuse Using Design Science Research to Develop Design Principles for Implementing A Component Repository; M. Sc. Thesis on Informatics: Programming and System Architecture; Department of Informatics Faculty of mathematics and natural sciences University of Oslo.

https://www.mn.uio.no/ifi/english/research/groups/is/research-library/masters-theses/bengtsson.pdf access date: 26th July, 2021

[23] Kamalraj, R., Kannan, A. R. and Ranjani, P. (2012). Classification and Selection of Reusable Software Components Using Mutation Technique; International Journal of Computer Science And Technology (IJCST) Vol. 3, Issue 3 http://www.ijcst.com/vol33/3/a952.pdf

[24] Ekanem, B. A. and Woherem, E. (2016). Legacy Components Stability Assessment and Ranking Using Software Maturity Index. International Journal of Computer Applications (IJCA), Foundation of Computer Science (FCS), 134(13), New York, USA, 2016

https://www.ijcaonline.org/archives/volume134/number13/23976-2016908157 Access Date: 12th March, 2021

[25] Arsene, K. K. I., Adama, S., Prosper, K. K. and Marcellin, B. K. (2019). Proposal of Automatic Methods for the Reuse of Software Components in a Library; (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 10, No. 2, 2019

https://thesai.org/Downloads/Volume10No2/Paper_8-Proposal_of_Automatic_Methods.pdf

[26] Qureshi, M. R. J. and Sandhu, M. E. (2013). Survey-Based Analysis of the Proposed Component-Based Development Process; https://arxiv.org/ftp/arxiv/papers/1202/1202.2498.pdf

[27] AL-Badareen, A. B, Selamat, M. H., Jabar, M. A., Din, J. and Turaev, S. (2011). Reusable Software Component Life Cycle, INTERNATIONAL JOURNAL OF COMPUTERS Issue 2, Volume 5, 2011 https://www.naun.org/main/NAUN/computers/19-863.pdf

[28] Capilla, R., Gallina, B., Cetina, C., & Favaro, J. (2019). Opportunities for Software Reuse in an Uncertain World: from Past to Emerging Trends. Journal of Software: Evolution and Process, 31(8). https://doi.org/10.1002/Smr.2217

[29] Kaur, V. and Goel, S. (2011). Facets of Software Component Repository; International Journal on Computer Science and Engineering 3(6)

https://www.researchgate.net/publication/228726189_Facets_of_Software_Component_Repository

[30] Manjhi, D. and Chaturvedi, A. (2019). Software Component Reusability Classification in Functional Paradigm, 2019 IEEE International Conference, 2019

https://ieeexplore.ieee.org/abstract/document/8869123 Access Date: 19th April, 2021

[31] Kalian, A. (2014). Characterization of Reusable Software Components For Better Reuse, International Journal of Research in Engineering and Technology 03(05):584-588, 2014

https://www.researchgate.net/publication/273301351_Characterization_Of_Reusable_Software_Components_for_Better_Reuse Access Date: 12th March, 2021

[32] Hudaib, A., Huneiti, A. and Othman, I. (2016). Software Reusability Classification and Predication Using Self-Organizing Map (SOM), Communications and Network, 8(3), 2016

https://www.scirp.org/journal/paperinformation.aspx?paperid=69828 Access Date: 17th April, 2021

[33] Ali, M. A. and Yap, N. K. (2019). Software Component Quality Model; International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, 9(1), October 2019 https://www.ijeat.org/wp-content/uploads/papers/v9i1/A2659109119.pdf

[34] Sharma, H. (2016). Software Reuse. https://www.slideshare.net/indusharma3517/software-resuse Access Date: 28th May, 2021

[35] Selvarani, R. and Mangayarkarasi, P. (2017). Modeling of Reusability Estimation in Software Design with External Constraints, Software Project Management for Distributed Computing, Springer, Cham, pp. 3-23

[36] Kessel, M. and Atkinson, C. (2016). Ranking software components for reuse based on non-functional Properties, *Information Systems Frontiers* **volume 18**, pages825–853 (2016)

https://link.springer.com/article/10.1007/s10796-016-9685-3

[37] Mittal, S. and Bhatia, P. K. (2013). Framework for Evaluating and Ranking the Reusability of COTS Components based upon Analytical Hierarchy Process; International Journal of Innovations in Engineering and Technology (IJIET)

http://ijiet.com/wp-content/uploads/2013/09/47.pdf

[38] Garga, R., Sharma, R. K. and Sharma, K. (2016). Ranking and selection of commercial off-the-Shelf using Fuzzy distance based approach; Decision Science Letters 5 (2016) 201–210

http://www.growingscience.com/dsl/Vol5/dsl_2015_51.pdf

[39] Kim, H. (2008). Automatic Identification of Potential Reusable Mobile Components; IAENG International Journal of Computer Science, 35(4)

http://www.iaeng.org/IJCS/issues_v35/issue_4/IJCS_35_4_04.pdf

[40] Gall, H. (2010). ReUse: Challenges and Business Success; Software Evolution & Architecture Lab (Seal). https://www.slideshare.net/hcgall/software-reuse-challenges-and-business-success Access Date: 28th May, 2021

[41] Jha, S. K. and Mishra, R. K. (2016). Multi Criteria Based Retrieval Techniques for Reusable Software Components From Component Repository; International Journal of Engineering Applied Sciences and Technology, 2016 Vol. 1, Issue 6, pp. 88-91

https://www.ijeast.com/papers/88-91,Tesma106,IJEAST.pdf

[42] Yannibelli, V., Pacini, E., Monge, D., Mateos, C. and Rodriguez, G. (2020). A Comparative Analysis of NSGA-II and NSGA-III for Autoscaling Parameter Sweep Experiments in the Cloud; Hindawi Scientific Programming Volume 2020, https://www.hindawi.com/journals/sp/2020/4653204/

[43] Basha, N. M. J. and Mohan, C. (2014). A Strategy To Identify Components Using Clustering Approach For Component Reusability; Computer Science & Information Technology
https://arxiv.org/ftp/arxiv/papers/1406/1406.4123.pdf

[44] Srinivas, C., Radhakrishna, V. and Rao, C. V. G. (2013). Clustering Software Components for Component Reuse and Program Restructuring; ICCC '13: Proceedings of the Second International Conference on Innovative Computing and Cloud Computing, December 2013; ACM Digital Library, pp.261–266
https://dl.acm.org/doi/abs/10.1145/2556871.2556933

[45] Shtern, M. and Tzerpos, V. (2012). Clustering Methodologies for Software Engineering; Advances in Software Engineering, Hindawi

https://www.hindawi.com/journals/ase/2012/792024/

[46] Srinivas, C., Radhakrishna, V. and Rao, C. V. G. (2014A). Clustering and Classification of Software Component for Efficient Component Retrieval and Building Component Reuse Libraries; 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014; Procedia Computer Science 31 (2014) 1044 – 1050

https://pdf.sciencedirectassets.com/280203/1-s2.0-S1877050914X00050

[47] Srinivas, C., Radhakrishna, V. and Rao, C. V. G. (2014B). Software Components Clustering and Classification Using Novel Similarity Measure, Procedia Technology, 8th International Conference Interdisciplinary in Engineering, INTER-ENG 2014, 9-10 October 2014, Tirgu-Mures, Romania; 19 (2015) 866-873, Elsevier.

https://core.ac.uk/download/pdf/82019856.pdf

[48] Shahmohammadi, G., Jalili, S. and Hasheminejad, S. M. H. (2010). Identification of System Software Components Using Clustering Approach, Journal of Object Technology 9(6):77-98

https://www.researchgate.net/publication/220299201_Identification_of_System_Software_Components_Using_Clustering_Approach

[49] Albani, A., Overhage, S., Birkmeier, D. (2008). Towards a Systematic Method for Identifying Business components. In: Chaudron MRV, Szyperski, CA, Reussner, RH, eds *Component-based Software Engineering: 11th International Symposium, CBSE 2008, Karlsruhe, Germany, October 14-17, 2008, Proceedings,* Berlin, Germany:Springer-Verlag Berlin Heidelberg; 2008:262-277, Leture Notes in Computer Science; vol 5282

[50] Hamza, H. S. (2009). A Framework for Identifying Reusable Software Components using Formal Concept Analysis. In: Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations: 2009; Las Vegas, NV

[51] Hasheminejad, S. M. H. and Gholamshahi, S. (2107). PCI-PSO: Preference-Based Component Identification Using Particle Swarm Optimization; Journal of Intelligent Syatems; 2019; 28(5): 733–748; De Gruyter Publishers

https://www.degruyter.com/document/doi/10.1515/jisys-2017-0244/html

[52] Oliveira, J., Fernandes, E., Souza, M. and Figueiredo, E. (2016). A Method Based on Naming Similarity to Identify Reuse Opportunities; XII Brazilian Symposium on Information Systems, Florianópolis, SC, May 17-20, 2016 https://sol.sbc.org.br

[53] Hasheminejad, S. M. H. and Jalili, S. (2014). An evolutionary approach to identify logical components; Journal of Systems and Software; Elsavier, Vol. 96, October 2014, Pages 24-50

https://www.sciencedirect.com/science/article/abs/pii/S0164121214001228#!

[54] Aziz, M. A and North, S. (2007). Retrieving Software Component using Clone Detection and Program Slicing; Research Publication, The Memorandum CS-07-03, Department of Computer Science

http://www.dcs.shef.ac.uk/intranet/research/public/resmes/CS0703.pdf

[55] Kessel, M. and Atkinson, C. (2018). Integrating reuse into the rapid, continuous software Engineering cycle through test-driven search; Association for Computing Machinery (ACM) Digital Library https://dl.acm.org/doi/10.1145/3194760.3194761

[56] Upadhyay, N., Despande, B. M., &Agrawal, V. P. (2011). Towards a software component quality model. In International Conference on Computer Science and Information Technology (pp. 398-412).Springer, Berlin, Heidelberg

[57] Amin**, F.** Mahmood, A. K. **and** Oxley, A. (2011). A Review of Software Component Reusability Assessment Approaches, Research Journal of Information Technology, 3: 1-11, 2011

https://scialert.net/fulltext/?doi=rjit.2011.1.11 Access Date: 17th April, 2021

[58] Priyalakshmi, G. and Latha, R. (2018). Evaluation of Software Reusability Based on Coupling and Cohesion, International Journal of Software Engineering and Knowledge Engineering,

Vol. 28, No. 10, pp. 1455-1485, 2018

https://www.worldscientific.com/doi/abs/10.1142/S0218194018500420 Access Date: 12th March, 2021

[59] Alvaro, A., Almeida, E. S., & Meira, S. L. (2005). Quality Attributes for a Component Quality Model. 10th WCOP/19th ECCOP, Glasgow, Scotland, 31-37.

[60] Ekanem, B. A. (2016). Dealing with Components Reusability Issues as Cutting-edge Applications Turn legacy, SAI Computing Conference Proceedings 2016, July 13-15, 2016, London, UK, 2016 https://ieeexplore.ieee.org/ie17/7547233/7555953/07556129.pdf

Access Date: 12th March, 2021

[61] Kaur, R., Arora, S., Jha, P. C. and Madan, S. (2015). Fuzzy Multi-criteria Approach For Component Selection of Fault Tolerant Software System under Consensus Recovery Block Scheme; International Conference on Advanced Computing Technologies and Applications (ICACTA2015); Procedia Computer Science 45 (2015) pp. 842 – 851; Elsevier

[62] Ekanem, B. A. 2015. Assessment of Components Stability for Modernization Using Software Maturity Index, International Journal of Scientific Research and Engineering Studies (IJSRES) 2(12) www.ijsres.com Retrieved on: March 16, 2020

[63] Ajayi, O. O., Chiemeke, S. C. and Ukaoha, K. C. (2019). Assessing the stability of selected Software Components for Reusability; International Journal of Intelligent Computing and Information Science. https://ijicis.journals.ekb.eg/article_96103_d688575be2b0486cc0124407a0945c91.pdf

[64] Niranjam, P. and GuruRao, C, V. (2011). A Model Software Reuse Repository with an Intelligent Classification and Retrieval Technique, Computer Science and Engineering; 1(1): 15-21; 2011

http://article.sapub.org/10.5923.j.computer.20110101.03.html Access Date: 17th April, 2021

[65] Capretz, L. F., Ahmed, F., Al-Maati, S. and AlAghbari, Z. (2008). COTS-Based Software Product Line Development; International Journal of Web Information Systems, Volume 4, Number 2, pp. 165-180, Emerald Group Publishing, 2008. https://core.ac.uk/download/pdf/61639163.pdf

[66] Kessel, M. and Atkinson, C. (2015). Ranking Software Components for Pragmatic Reuse, 2015 IEEE/ACM 6th International Workshop on Emerging Trends in Software Metrics, 2015

[67] Flood, C. A. (2017). Unified Software Engineering Reuse: A Methodology for Effective Software Reuse; Master's Theses and Graduation Research, San Jose State University

https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=8343&context=etd_theses