Reusability Metrics of Software Components: Survey

Marko Mijač, Zlatko Stapić
Faculty of Organization and Informatics
University of Zagreb
Pavlinska 2, 42000 Varaždin, Croatia
{marko.mijac, zlatko.stapic}@foi.hr

Abstract. *Reusing software assets has many advantages and has been essential feature of all software development approaches. Component based software development has been especially inspired by reuse. In order to reuse software component, the component has to be designed and built for reusability. Since reusability is influenced by a number of different factors, there are various approaches and metrics used to measure reusability. In this paper we conducted extensive literature review in order to identify reusability metrics and factors influencing reusability. Total of 39 papers introducing reusability metrics were found and analyzed. We identified 36 different factors influencing reusability, 12 black box component metrics and more than 20 white box/glass box metrics.*

Keywords. Software Components, Reusability, Metrics, Software Quality

1 Introduction

From the very beginning of software development, reusability has been considered as one of the most important characteristics of software quality. Different approaches to software development in different ways and at different levels tried to achieve reuse. Over time it became clear that most of the artifacts that arise in the process of software development can be reused to some extent. For example, Frakes et al. [1] name several active areas of reuse possibilities, e.g.: reuse libraries, domain engineering methods and tools, reuse design, design patterns, domain software architecture etc.

Although reuse of software has been an essential feature of all software development approaches, there is one approach in which reuse is the main determinant. It is a component-based software development (CBSD), which according to Sharma et al. [2] emphasizes the design and construction of software systems using reusable components.

In order to understand reusability, it is essential to differentiate reusability and reuse. Frakes et al. [1] describe *reuse* as the use of existing software and software knowledge to construct new software. Same authors define *reusability* as a property of a software asset that indicates its probability of reuse.

To be able to embed component into various systems, the component must be reusable. It is therefore essential both in the process of constructing components and the process of searching for existing components, to recognize the characteristics that make the component reusable. This way we can define metrics for quantification of these characteristics, which is going to allow us to evaluate, compare and finally choose appropriate component.

In the academic community and the industry as well, a number of metrics for measuring reusability of components, quality models and frameworks are proposed. There are also a few papers and dissertations containing a survey of existing reusability metrics (see Shanmugasundaram et al. [3], Koteska et al. [4], Kumar et al. [5], Lee [6], Sharma et al. [2] and [7], Suri et al. [8]). However, to our best knowledge, no systematic reviews have been conducted on this topic. Therefore, the goal of this paper is to make a systematic review of the literature and research in the field of software components' reusability metrics, in order to identify relevant reusability metrics.

The paper is organized as follows. The second section describes research method used to review papers on reusability metrics. The third section contains overview and meta-analysis of relevant papers. The results of the review are presented in fourth section. Finally, last section answers the research question and concludes the topic.

2 Research method

In order to conduct thorough and systematic literature review and to minimize possible bias, guidelines for systematic literature review method from Kitchenham, elaborated in [9] and [10] were used.

Prior to selection of relevant studies a review protocol has been formulated, which is composed of sections including definition of research question, search strategy, selection criteria, study quality assessment and data extraction strategy.

2.1 Research question

The research question that the review is intended to answer is following: *What* are *the proposed metrics for reusability of software components?*

2.2 Search strategy

From the research question following keywords were extracted in order to form search queries: *software components, metrics, reusability, reuse.*

Using logical operators following search queries were constructed:

SQ1: *software components AND metrics AND reusability*
SQ2: *software components AND metrics AND reuse*
Generic query: software components AND metrics AND (reusability OR reuse)

In order to obtain relevant research papers on given topic, constructed search queries were executed on several scientific databases: *SCOPUS, IEEE Xplore, ACM Digital Library, Google Scholar.*

To obtain only relevant papers search queries will be executed on title, abstract, keywords and other available meta-data. Considering Google Scholar does not offer searching by abstract and title, executing search query results in a vast number of papers not relevant for current study. To mitigate this, Google Scholar will be instructed to sort results by relevance, and first 100 papers will be taken into consideration. Also, papers published prior to 2000 will not be considered.

2.3 Selection criteria

Following table contains explicitly defined selection criteria used to decide which papers are relevant and which are not.

Table 1 Inclusion and exclusion criteria

Inclusion criteria	Exclusion criteria
English written papers	Non-English written papers
Papers published in scientific conferences and journals in Information and Computer sciences.	Papers published in non-related areas.
Papers focused on software components	Papers not related to the research topic.
Papers focused on reusability metrics	Papers focused on metrics describing level of reuse.
Papers containing metrics which give quantitative or at least linguistic measure of reusability.	Papers which do not propose concrete reusability metrics.
Papers published between 2000 and 1st. Quarter of 2014.	Papers published prior to 2000.
	Duplicated studies

The first selection of papers will be performed by executing aforementioned search queries on selected scientific databases. Then, duplicated results will be excluded. After that, the papers will be evaluated by their title and abstract, and those not related to current study will be excluded. For remaining papers full-text version will be obtained and examined. Here, the papers that don't satisfy quality criteria or don't focus on topic of our study will be excluded. For example, while initially analyzing a small subset of papers obtained from aforementioned databases, it was clear that some papers covered metrics for determining the level of reuse in software systems. Although in search query we included keyword *reuse*, to prevent missing relevant metrics, we do not explicitly deal with reuse metrics in this paper. Rather our study focuses on metrics for determining the ability of components to be reused (reusability), so papers describing *reuse* metrics were excluded.

Selection criteria has been defined by a single researcher, but evaluated by colleague senior researcher. Suggestions and requests made by second researcher were taken into consideration.

2.4 Study quality assessment

The quality of analyzed studies will be assessed by considering quality of scientific publication the study is published in. Also, the studies will be assessed according to the overall structure, methodology, level of scientific evidence and reasoning.

2.5 Data extraction strategy

Data will be extracted by thoroughly reading and examining papers that satisfied selection criteria. Along this process, wherever necessary, for each examined paper separate notes and comments will be taken. Also, in order to simplify management of bibliographic and citation data, all relevant papers will be entered in bibliographic management software – Zotero.

3 Overview of the included papers

3.1 Selection process

Initial search query run on aforementioned scientific databases resulted in total of 515 papers. Through 3 selection iterations inclusion and exclusion criteria were applied, and a final number of 39 relevant papers were obtained.

Initial pool of 515 papers contained a number of duplicate papers, so in the first iteration duplicate papers were eliminated, resulting in total of 443 unique papers. In the second iteration papers were evaluated according to title and abstract data, since in our opinion selection of papers based solely on title data is inaccurate. Even so, for some papers it was difficult to determine the relevancy in this iteration or we were simply not sure if they are relevant or not. These papers entered next iteration, to be evaluated more thoroughly. Total of 142 papers entered third iteration, and were evaluated according to introduction, conclusion and brief scan of the paper content. However, full texts of 14 papers we were not able to obtain, due to database restrictions, broken links or simply were not available online, so they were omitted in the third iteration. The reasons for exclusion of 34 papers applied in third iteration are as follows:

Table 2 Reasons for exclusion in third iteration

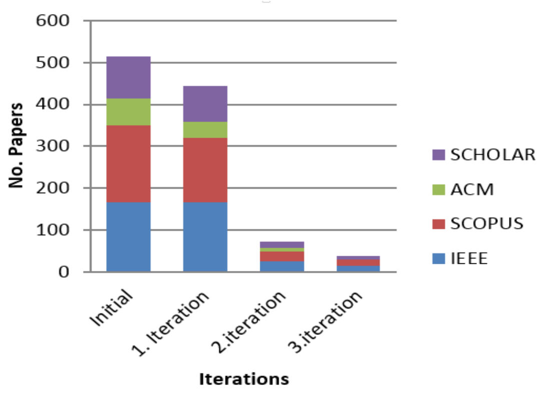
Reason for exclusion	**No. Papers**
Survey of existing metrics	8
Replicating existing research	4
No explicit reusability metrics	10
Validation of existing metrics	5
Not relevant topic	7
TOTAL	**34**

This resulted in total of 39 relevant papers. Following table and graph show detailed data per iteration and database:

Table 3 Number of papers per iteration

	Initial	**1. Iter.**	**2. Iter.**	**3. Iter**
IEEE	167	166	25	14
SCOPUS	184	154	24	15
ACM	64	39	8	2
SCHOLAR	100	84	16	8
TOTAL	**515**	**443**	**73**	**39**

Figure 1 Number of papers per iteration

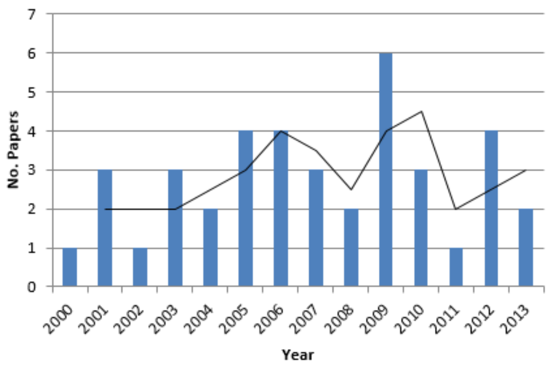


The final list of papers that met inclusion criteria and that were identified for final data extraction contains following 39 papers: [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48] and [49].

3.2 Temporal view

If we take a look at the **Figure 2** depicting number of relevant papers per year it is hard to spot if there is an increasing or decreasing trend in publishing papers that deal with reusability metrics. However, we can notice that every year at least one paper that proposes new reusability metrics or uses existing metrics to measure reusability in a novel way has been published. This can be interpreted as a confirmation that measuring reusability is an important and always present topic, and that we can safely assume that it will remain such. Reusability of components depends upon numerous different external and internal factors. The very nature of the link between reusability and these factors is often unclear and is hard to quantify. So we cannot expect sudden solution of reusability problem, but rather we can expect gradual improvements in quantifying reusability.

Figure 2 Number of papers published per year

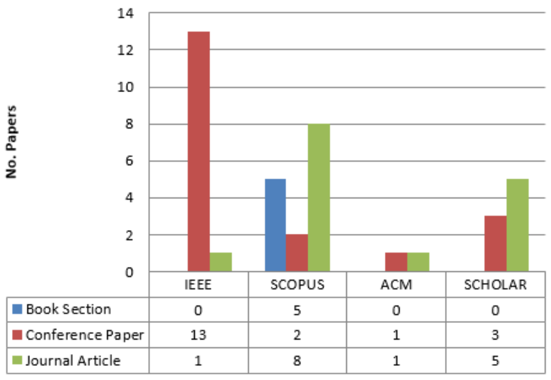


3.3 Types of papers

Among 39 relevant papers most of them are presented on various scientific conferences and published in conference proceedings (19 papers), while 15 papers sourced from journal articles. Finally, 5 papers were published as a book sections.

Figure 3 shows detailed distribution of papers per type and per database. We can see how conference papers prevail in IEEE Xplore database, while in other databases journal articles are dominant type of papers.

Figure 3 Number of papers per type



	IEEE	SCOPUS	ACM	SCHOLAR
🟦 Book Section	0	5	0	0
🟥 Conference Paper	13	2	1	3
🟩 Journal Article	1	8	1	5

3.4 Author analysis

Chosen 39 relevant papers were authored by total of 81 authors. As it can be seen in the following table, majority of identified authors authored only one paper, while only five of them authored two or more papers.

Table 4 Number of authored papers

No. of papers authored	No. of authors	Authors
1	75	-
2	3	H. Washizaki, V. Gupta, Sharma A.
4	2	S.D. Kim, H. Singh
9	1	P.S. Sandhu
TOTAL	**81**	

3.5 Citation analysis

Citation data extracted from Google Scholar shows that 31 out of 39 papers were cited at least once, while 8 papers were still not cited. However, one should take into account that low citation of some papers can be justified with relatively recent publish date. Following table shows citation rate in detail.

Table 5 Paper citation categories

Cited by	< 10	10 - 19	20 - 29	30 - 39	40 - 49	50 - 59	≥ 60
No. Papers	20	7	4	1	2	1	4

Following table shows a list of 10 most cited studies:

Table 6 Top 10 most cited papers

Rank	Cited by	Paper
1	245	Sant'Anna C. et al. [11]
2	170	Washizaki H. et al. [12]
3	94	Cho E.S. et al. [13]
4	70	Boxall M.A.S. et al. [14]
5	54	Etzkorn L.H. et al. [15]
6	48	Aggarwal K.K. et al. [16]
7	41	Kim S.D. et al. [17]
8	38	Dandashi F. [18]
9	29	Sharma A. et al. [19]
10	26	Sandhu P.S. et al. [20]

4 Results of the review

The results of the data extraction on factors influencing reusability are stated in **Table 10**, along with metrics used to evaluate these factors and with proposed overall reusability metrics. This table also classifies papers whether they consider black box or glass/white box components. Some approaches described in these papers passed some form of validation, so this also is indicated here.

4.1 Factors influencing reusability

Reusability is hard to quantify because of numerous factors influencing it. To complexity of the situation contributes the fact that it is often not clear at which extent some factor influences reusability. Also these factors do not only influence reusability, but they also influence each other.

We extracted the set of these factors which can be found in relevant papers, and the frequency of their appearance. Total of 36 extracted factors confirm that measuring and quantifying reusability is not an easy task. However we can see that internal factors such as coupling, cohesion, and complexity dominate and were used to asses reusability in most of the relevant papers.

Table 7 Number of papers mentioning certain reusability factors

Factor	No. of papers
Coupling	18
Cohesion	11
Complexity	8
Portability, Inheritance, Volume, Reuse frequency	6
Understandability, Interface complexity, Customizability, Regularity	5
Adaptability, Maintainability	4
Modularity, Commonality	3
Composability, Comprehensability, Usability, Functionality, Reliability, Documentation, Size	2
Extent of templating, Domain context, Interface soundness, Functional dependencies, Usefulness, Variability, Ease-of-use, Transplanting, Utilizability, Configurability, Compatibility, Separation of Concerns, Completeness, Confidence	1

If we take a look at individual papers, we can see that most of them used multiple factors to appropriately quantify reusability. Only 11 proposed metrics relied solely on one factor.

Table 8 No. of factors used to quantify reusability

No. of factors	1	2	3	4	5	6	7	8
No. of papers	11	3	8	5	8	1	2	1

4.2 Type of assessed components

One of the ways to classify reusability metrics is according to which component's box model metric can be applied to. As reported by Sharma et al. [2] component can be characterized as black box, glass box or white box component. According to Lee and Choi [21] these components differ by two criteria: visibility and changeability. Black box component offers only interface to external user, while the implementation details are hidden. Conversely, glass box and white box components allow external user to see internal structure of component, with the difference that the white box component allows you to modify source code.

As can be seen from the following table, analysis has shown that a majority of papers proposed metrics for white box and glass box components. This can be seen as a confirmation that internal structure of components, its classes, methods and properties have great impact on reusability. However, these metrics can be applied only in situations when the source code of components is available. For example when we are dealing with:

- Proprietary, in-house components,
- Open-source components,
- Commercial components with available source code.

Table 9 Number of papers proposing metrics for certain component type

Component type	No. Papers
White/Glass box	25
Black box	12
N/A	3

Conversely, metrics for black box components are based on externally visible properties of components and environment variables, so they can be applied in situations where the source code is not available. Such is the case with commercial components that are offered exclusively as compiled executable files or libraries.

4.3 Metrics used to quantify reusability

In quantifying reusability most authors relied on existing well-known metrics. Some papers offer direct metrics for reusability which result in explicit quantitative or linguistic value depicting level of reusability. Some, however, offer indirect metrics, which give value for some other quality characteristic or measured factor, which again greatly influences reusability. In such cases, evaluating reusability is a matter of interpretation of given values.

Just as large is the number of factors influencing reusability; equally large is the number of different metrics to measure these factors.

4.3.1 Black box metrics

Washizaki et al. [12] propose black box metric *component overall reusability* (COR) which combines metrics for understandability, adaptability and portability. Understandability is measured by EMI (Existence of meta-information) metric, adaptability by RCC (Rate of Component Customizability) metric, and portability by SCCr (Self Completeness of Component's Return value).

Another metric for assessing reusability of black box components is proposed by Rotaru et al. [22]. They propose metrics for compose-ability and adaptability of component which proportionally influence component's reusability.

Wang et al. [23] advocate reusability metric based on complexity of component's interface. They compare the number of public members (methods, properties, events) of component, with the total number of component members in order to measure reusability.

Cho et al. [13] took similar approach by comparing sum of interface methods providing common functions to sum of total interface functions.
Another approach which is based on component's interface is described by Boxal et al. [14]. They evaluate component's reusability by assessing understandability of component's interface. They measure interface size, number of distinct arguments, argument repetition, argument identifiers, and density of reference arguments.

Bhattacharya et al. [24] took slightly different approach. They do not consider characteristics of the component itself, but rather consider the context and problem domain in which component is going to be applied.

Kim et al. [25] define component quality model, and metrics for several quality characteristics. They measure reusability by measuring level of component commonality, modularity, customizability and comprehensiveness.

Sagar et al. [26] claim that reusability is affected by many different factors, and that there is no straight-forward way to estimate them. Therefore they applied soft-computing fuzzy approach to estimate reusability considering four factors: customizability, interface complexity, portability and document quality. Similar approach was taken by Jatain et al. [27], who identified five factors (customizability, configurability, interface complexity, portability, compatibility) which influence reusability, and estimated reusability using fuzzy approach. Sharma et al. [19] identified four factors influencing reusability (customizability, interface complexity, portability, understandability) and built artificial neural network in order to assess reusability.
Venkatesan et al. [28] extracted usability, portability and confidence as factors influencing reusability, and used them to estimate reusability level.

Lee and Choi [21] offer reusability metrics for both black box and white/glass box components. According to them, reusability of black box component can be estimated considering reuse frequency.

4.3.2 White box / Glass box metrics

If we are owners of component, or we can obtain component's source code some other way, we will have a large number of white/glass box metrics for measuring reusability at our disposal. Reusability metrics we were able to identify are as follows.

Table 10 Final extracted data on factor metrics

Paper	Type	Factors	Factor metrics	Reusability metric	Validated
[12]	B	Understandability, Adaptability, Portability	Existence of Meta Information (EMI), Rate of Component Customizability (RCC), Self-Completeness of Component's Return Value (SCCr)	COR - Component Overall Reusability	Yes
[22]	B	Adaptability, Composability	Component Adaptability (Ac(kc)), Composability Degree of Component (σc(μc))	-	No
[29]	G, W	Coupling, Cohesion, Inheritance	CK metrics (WMC, DIT, NOC, CBO, LCOM), Regression algorithms	-	Yes
[23]	B	Interface complexity	No. public members in component interface	R(C) - Reusability of component	No
[16]	G, W	Extent of templating	FTF - Function Template Factor, CTF - Class Template Factor	-	Yes
[30]	G, W	Coupling, Volume, Complexity, Regularity, Reuse Frequency	McCabe's Cyclomatic Complexity, Halstead Software Science Indicator	f(u) - Reusability using Segueno-Type Fuzzy Inference System	Yes
[13]	G, W	Complexity, Customizability, Reuse Level	CPC - Component Plain Complexity, CSC - Component Static Complexity, CDC - Component Dynamic Complexity, CCC - Component Cyclomatic Complexity, CV - Component Variability for Customizability	CR – Component reusability	Yes
[32]	G, W	Complexity, Regularity. Volume, Reuse Frequency, Coupling	McCabe's Cyclomatic Complexity, Regularity metric, Halstead Software Science Indicator, Reuse frequency metric, LCOM	Level of reusability using: Fuzzy-based approach, Neuro-fuzzy based approach, Hybrid fuzzy-GA based approach	Yes
[21]	B, G, W	Reuse Frequency, Understandability, Rate of modification	-	R = f(Rf), R = f(Rf, Cu), R = f(Rf, Cu, Rm)	No
[24]	B	Domain Context	-	R = f (context)	Yes
[14]	B	Understandability	APP - Arguments per Procedure, DAC - Distinct Argument Count, DAR – Distinct Argument Ration, ARS - Argument Repetition Scale, MSC - Mean String Commonality, MIL - Mean Identifier Length, RAD - Reference Argument Density	-	Yes
[31]	G, W	Coupling, Cohesion, Inheritance	CK metrics (WMC, DIT, NOC, CBO, LCOM),	Level of reusability using: Neuro-fuzzy inference system	Yes
[36]	G, W	Coupling, Cohesion	WTcoh - Weighted Transitive Cohesion measure, WTcoup - Weighted Transitive Couping measure	-	Yes
[37]	G, W	Cohesion	SBFC - Similarity-Based Functional Cohesion	-	Yes
[38]	G, W	Commonality, Modularity, Adaptability, Comprehensability, Composability, Interface soundness	CoS - Cohesiveness of Services, MoD - Minimality of Dependencies, CoV - Coverage of Variability, CoA - Completeness of Variant Set, ESA - Effectiveness of Service Adaptation, ICC - Inter-CaaS Composability, CAC - CaaS-to-Application Composability	IR – Integrated reusability (average value)	Yes
[17]	W	Functional dependencies	Functional dependencies metric	-	Yes
[39]	G, W	Complexity, Volume, Regularity, Reuse Frequency, Coupling	McCabe's Cyclomatic Complexity metric, Halstead Software Science Indicator, Regularity Metric, Reuse-Frequency Metric, LCOM	Feed Forward Neural Network (with SCG algorithm)	Yes
[40]	G, W	Coupling	CC - Coupling between Classes, CPC - Coupling of Component, CBC - Coupling between Components	-	Yes
[41]	G, W	-	-	Quality Framework	Yes
[48]	-	Usability, Usefulness, Variability	-	-	-
[42]	G, W	Cohesion, Coupling	COHOT - Cohesion Metric by Operation Type, COPOT - Coupling Metric by Operation Type	-	-
[43]	G, W	Coupling	DD - Component Dependency Metric	-	Yes
[25]	B	Commonality, Modularity, Customizability,Comprehensability	Metric for Commonality, Metric for Modularity, Metric for Customizability, Metric for Comprehensiveness	-	No
[33]	G, W	Complexity, Volume, Regularity, Reuse Frequency, Coupling	McCabe's Cyclomatic Complexity metric, Halstead Software Science Indicator, Regularity Metric, Reuse-Frequency Metric, LCOM	Neural Network - Resilient Backpropagation algorithm (RB)	Yes
[34]	G, W	Coupling, Cohesion, Inheritance	CK metrics (WMC, DIT, NOC, CBO, LCOM)	Neural Network - Levenberg & Marguardt algorithm	Yes
[44]	G, W	Cohesion	PCoh - Package Cohesion	-	Yes
[47]	-	Coupling, Cohesion, Inheritance	CK metrics (WMC, DIT, NOC, CBO, LCOM)	K-means clustering algorithm & Decision Approach	Yes
[35]	G, W	Coupling	CC1 - CC8 (Coupling Counts)	-	Yes
[45]	G, W	Functionality, Reliability, Maintainability, Ease-of-use, Transplanting	-	RV - Reusability Value based on Colony Decision	Yes
[46]	-	Functionality, Reliability, Maintainability, Utilizability, Portability	-	RMV (Reusability Measure Value)	No
[26]	B	Customizability, Interface complexity, Portability, Documentation	-	Soft Computing Fuzzy approach	Yes
[27]	B	Customizability, Configurability, Interface complexity, Portability, Compatibility	-	Fuzzy inference system	Yes
[11]	G, W	Separation of Concerns, Size, Cohesion, Coupling	-	-	Yes
[20]	G, W	Complexity, Regularity, Volume, Reuse Frequency, Coupling	-	Neuro-fuzzy inference system	Yes
[19]	B	Customizability, Interface complexity, Portability, Understandability	-	Artificial Neural Network	Yes
[15]	G, W	Modularity, Coupling, Cohesion, Interface complexity, Documentation, Complexity, Size	Modularity = f(Lack of Coupling, Cohesion), Interface size = Adjusted Number of public methods, Documentation = A degree of documented lines of code, methods, attributes, Complexity = f(Size, Complexity of Methods)	Reusability = f(Modularity, Interface Size, Documentation, Complexity)	Yes
[49]	G, W	Coupling, Cohesion, Inheritance	CK metrics (WMC, DIT, NOC, CBO, LCOM)	Neuro-fuzzy inference system	Yes
[18]	G, W	Adaptability, Completeness, Maintainability, Understandability, Complexity, Volume, Inheritance, Coupling	McCable's Complexity, Halstead's Volume, NPSS (Number of Physical Source Statements, Depth of Nested Statements, WMC (Weighted Methods per Class), Depth of Inheritance, Number of immediate children, RFC (Response for a Class), Coupling between objects (CBO).	-	Yes
[28]	B	Usability, Portability, Confidence	-	-	Yes

When dealing with reusability, Zahara et al. [29] are considering coupling, cohesion and inheritance. Precisely, they use well-known OO-based CK metrics and four regression algorithms to estimate reusability.
Aggarwal et al. [16] are also targeting only object-oriented systems, but with different and unique approach. They evaluate reusability according to the extent of template classes and template methods used in component.

Another metric based on fuzzy approach is proposed by Sandhu and Singh. [30]. The factors they consider in evaluating reusability are: coupling, volume, complexity, regularity, reuse frequency. After obtaining values for these factors, they constructed Segueno-Type Fuzzy Inference system in order to estimate reusability.

In their other papers [31] and [20] the same authors built reusability evaluation system based on well-known CK metrics and complexity, regularity, volume, reuse frequency and coupling. With these values and constructed Neuro-fuzzy inference system they estimated reusability.

In [32] Sandhu et al. devised framework of metrics (McCabe's Cyclomatic Complexity metric, Regularity metric, Halstead Software Science Indicator, Reuse Frequency metric and Coupling metrics) to calculate reusability. They compared 3 approaches to evaluate reusability: Fuzzy, Neuro-Fuzzy and Fuzzy-GA. Results showed that hybrid Fuzzy-GA algorithm performs the best.

Manhas et al. [33] in their paper conducted similar research. They devised framework of the same metrics and experimented with different Neural Network approaches in order to identify reusability of function oriented systems. The resilient backpropagation algorithm (RB) proved to be the best among five algorithms.

Sandhu et al. [34] took well-known CK metrics and paired them with 14 different neural network algorithms with the goal to predict reusability of object-oriented systems. Among different algorithms Levenberg-Marguardt algorithm proved to be best.
Shri et al. [35] in their paper use CK metrics, K-means clustering algorithm and Decision approach in order to predict reusability of object-oriented software systems.

Lee and Choi [21] in their paper propose new reusability measure that is in proportion to reuse frequency and component understanding.

Gui and Scott [36] developed new measures for coupling and cohesion in order to assess the reusability of Java components. The newly developed transitive measures were then compared with eight existing measures, and performed superior.

Al Dallal [37] proposes similarity-based functional cohesion metric of modules in procedural or object-oriented software. Author claims that highly cohesive modules are desirable because of their high reusability and maintainability.

Kim et al. [38] deal with Component-as-a-Service as one of various cloud services. Their evaluation framework for CaaS assesses reusability with eight quality attributes, each of them measured with one or more metrics. Overall reusability value is given by integrating all eight attributes.

Kim and Chang [17] in their paper developed a systematic, UML-based method to identify software components with high cohesion and low coupling. They propose four-step model: calculate functional dependencies, cluster use cases, allocate classes to components and select optimal components identification.

Sandhu and Chhabra et al. [39] took a framework of metrics (McCabe's Cyclomatic Complexity metric, Regularity metric, Halstead Software Science Indicator, Reuse Frequency metric and Coupling metrics) in order to calculate reusability. They conducted comparative analysis of Conjugate Gradient Algorithms and Particle Swarm Optimization Neural network approaches.

Choi and Lee [40] developed component-based coupling metrics using static and dynamic dependency, which can assist developers in designing more independent components.

Washizaki et al. [41] developed a quality framework for evaluating software source code (focused on C programming language), which can be used to effectively evaluate software's reliability, maintainability, reusability and portability.

Ko and Park [42] claim the best reuse is reuse of design rather than implementation. In their paper they propose component architecture redesigning approach using component coupling and cohesion metrics.

Yu and Ramaswamy [43] defined component dependency metric based on types of coupling and types of components. According to authors dependency of software reflects external quality characteristics of components, such as reusability and maintainability.

Gupta and Chhabra [44] in their paper proposed new metric for package cohesion. They claim that quantification of package cohesion can be useful in assessing reusability and overall quality of package.
Price et al. [45] introduce a set of reusability metrics, based on coupling count, to be applied in both design and implementation phase of component development. They also provide the guidelines in order to increase reusability.

Wang [46] formed decomposition model of component reusability based on functionality, reliability, maintainability, ease-of-use, and transplanting. Aiming to reduce shortcomings of traditional AHP method, the author proposed calculation model based on Colony decision, which proved to be more adequate.

Yingmei et al. [47] define reusability measure value (RMV) based on 5 component's sub-features: functionality, reliability, maintainability, utilizability and portability. Influence of particular sub-feature on overall reusability of component is managed by coefficients (weights) which vary according to component's application field.

Sant' Anna et al. [11] constructed An assessment framework, composed of two parts: a suite of metrics and a quality model. The purpose of the framework is to increase understanding of SoC, coupling, cohesion and size attributes as predictors of maintainability and reusability.

Etzkorn et al. [15] introduced an approach to automatically evaluate the reusability of legacy object-oriented software. They estimate reusability by calculating quality factors (modularity, interface size, documentation and complexity) and sub-factors influencing reusability.

Dandashi and Rine [18] demonstrated a method for assessing reusability of C++ components. The method consists of two phases: assessing direct quality attributes and assessing indirect quality attributes.

4.4. Answering research question

The research question we stated at the beginning of our research was: *What are the proposed metrics for reusability of software components?*

By performing research method of SLR we looked through the databases and identified 39 scientific sources that contributed to this research question (see **Table 10**). Overall, we found 36 reusability factors (see **Table 7** and **Table 10**) ranging from those that are mentioned in only data source to those that were used in 46% of all included papers.

These factors are used in 24 identified reusability metrics, presented in **Table 11**, which can be used to measure reusability of software components. This answers our research question.

Table 11 Identified reusability metrics

Reusability metric	Source
COR - Component Overall Reusability	[12]
R(C) - Reusability of component	[23]
Reusability using Segueno-Type Fuzzy Inference System	[30]
CR – Component reusability	[13]

R = f(Reuse Frequency),
R = f(Reuse Frequency, Understandability),
R = f(Reuse Frequency, Understandability, Rate of modification)
	[21]

R = f (Domain Context)	[24]
IR – Integrated reusability (average value)	[38]
Quality Framework	[41]
Artificial Neural Network	[19]
Neural Network - Resilient Backpropagation algorithm (RB)	[33]
Neural Network - Levenberg & Marguardt algorithm	[34]
Feed Forward Neural Network (with SCG algorithm)	[39]
K-means clustering algorithm & Decision Approach	[47]
RV - Reusability Value based on Colony Decision	[45]
RMV (Reusability Measure Value)	[46]
Soft Computing Fuzzy approach	[26]
Fuzzy inference system	[27]
Neuro-fuzzy inference system	[31], [20], [49]
Fuzzy-based approach, Neuro-fuzzy based approach, Hybrid fuzzy-GA based approach	[32]
R = f(Modularity, Interface Size, Documentation, Complexity)	[15]

5 Conclusions

The main goal of this paper was to systematically review existing metrics for estimating reusability of components. Conducted review showed that in the past 15 years a number of papers have dealt with reusability and the means to quantify reusability. Reusability has been acknowledged as one of the major quality aspects of software, but also as one that is difficult to measure. This can be seen as well from the fact that 36 factors and attributes were extracted from relevant papers which influence reusability. Also, a majority of metrics use more than one factor to estimate reusability.

We answered our research question by identifying 24 reusability metrics that build upon identified factors. The relationship between reusability and aforementioned factors, as well as between factors themselves is often vague and cannot be simply calculated and accurately expressed. This is why in 13 papers estimation of reusability included the use of neural networks, fuzzy logic, genetic algorithms, regression algorithms, soft computing, clustering algorithms etc.

Although by term components we often imply black-box components, such as COTS components, this study showed that still majority of reusability metrics are for white box/glass box components. Also, the factors mostly used to estimate reusability (coupling, cohesion, complexity) are provided for white box/glass box components. This is a confirmation that internal characteristics of component heavily influence the reusability.

With black box components we must rely on available external characteristics, such as interface complexity, interface size, documentation, reuse frequency etc., since the internal structure of component is not known. With white box/glass box components, however, we have the possibility to utilize both external and internal characteristics of components. By combining both black box and white box/glass box metrics we can surely get more credible results.

While majority of metrics relies on measuring various characteristics of software code, a few authors emphasize the importance of design artifacts of component or context/domain where the component will be deployed.

The purpose of this paper was not to evaluate or compare the metrics, but to identify them. Thus, there are many possibilities for further improvements and research including the validation, evaluation or comparison of proposed metrics, construction of frameworks and quality models, development of tools for automatic assessment of components' reusability, development of guidelines and suggestions for improving existing components or components under development and similar.

References

[1] W.B. Frakes and Kyo Kang, "Software reuse research: status and future," *IEEE Trans. Softw. Eng.*, vol. 31, no. 7, pp. 529–536, Jul. 2005.

[2] A. Sharma, Rajesh Kumar, and P. S. Grover, "Critical Survey of Reusability Aspects for Component-Based Systems" presented at the Enformatika, 2007.

[3] G. Shanmugasundaram, V. Prasanna Venkatesan, and C. Punitha Devi, "Research opportunities in service reusability of service oriented architecture" in *2012 International Conference on Emerging Trends in Science, Engineering and Technology (INCOSET)*, Tamilnadu, India, 2012, pp. 396–403.

[4] Bojana Koteska and Goran Velinov, "Component-Based Development: A Unified Model of Reusability Metrics" in *ICT Innovations 2012*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 335–344.

[5] Vijai Kumar, Arun Sharma, Rajesh Kumar, and P.S. Grover, "Quality aspects for component-based systems: A metrics based approach" *Softw. Pract. Exp.*, 42 (12), pp. 1531–1548, Dec. 2012.

[6] Y. Lee, "Automated source code measurement environment for software quality" Doctoral, Auburn Univeristy, Auburn, USA, 2007.

[7] A. Sharma, R. Kumar, and P. S. Grover, "Managing component-based systems with reusable components" *Int. J. Comput. Sci. Secur.*, pp. 52–57, 2007.

[8] P.K. Suri and Neeraj Garg, "Software Reuse Metrics: Measuring Component Independence and its applicability in Software Reuse" *Int. J. Comput. Sci. Netw. Secur.*, vol. 9, no. 5, pp. 237–248, 2009.

[9] B. Kitchenham, "Procedures for Performing Systematic Reviews" Eversleigh, 2004.

[10] Z. Stapić, E. López, A. Cabot, L. de Marcos Ortega, and V. Strahonja, "Performing systematic literature review in software engineering" in *Proceedings of 23rd Central European Conference on Information and Intelligent Systems*, Varaždin, Croatia, 2012.

[11] Claudio Sant'Anna, Alessandro F. Garcia, Christina Chavez, Carlos Lucena, and Arndt Staa, "On the reuse and maintenance of aspect-oriented software: An assessment framework" in *Proceedings of Brazilian Symposium on Software Engineering*, 2003.

[12] H. Washizaki, H. Yamamoto, and Y. Fukazawa, "A Metrics Suite for Measuring Reusability of Software Components" in *9th International Software Metrics Symposium Proceedings*, Sydney, Australia, 2003, pp. 211–223.

[13] E. S. Cho, M. S. Kim, and S. D. Kim, "Component metrics to measure component quality" in *Eighth Asia-Pacific Software Engineering Conference, 2001. APSEC 2001.*, Macau, China, 2001, pp. 419–426.

[14] M.A.S. Boxall and S. Araban, "Interface metrics for reusability analysis of components" in *Australian SE Conference Proceedings.*, Melbourne, Australia, 2004, pp. 40–51.

[15] L. H. Etzkorn, W. E. Hughes, and C. G. Davis, "Automated reusability quality analysis of OO legacy software" *Inf. Softw. Technol.*, vol. 43, no. 5, pp. 295–308, 2001.

[16] K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, "Software reuse metrics for object-oriented systems" in *Third ACIS International Conference on Software Engineering Research, Management and Applications, 2005*, Mount Pleasant, Michigan, USA, 2005, pp. 48–54.

[17] S. D. Kim and S. H. Chang, "A Systematic Method to Identify Software Components" in *11th Asia-Pacific Software Engineering Conference, 2004.*, Busan, Korea, 2004, pp. 538–545.

[18] F. Dandashi, "A method for assessing the reusability of object-oriented code using a validated set of automated measurements", 2002.

[19] A. Sharma, P. S. Grover, and R. Kumar, "Reusability assessment for software components", *ACM SIGSOFT Softw. Eng. Notes*, vol. 34, no. 2, p. 1, Feb. 2009.

[20] P. Sandhu and H. Singh, "Automatic Reusability Appraisal of Software Components using Neuro-fuzzy Approach", *Int. J. Inf. Technol.*, vol. 1, no. 8, pp. 2407–2413, 2007.

[21] Seungwon Lee and Ho-Jin Choi, "Software component reusability measure in component grid" in *11th International Conference on Advanced Communication Technology, 2009. ICACT 2009*, PyeongChang, South Korea, 2009, vol. 1, pp. 576–578.

[22] O.P. Rotaru and M Dobre, "Reusability metrics for software components" presented at the The 3rd ACS/IEEE International Conference on Computer Systems and Applications, Cairo, Egypt, 2005.

[23] Juan Wang and You-An Wang, "Teaching software reuse with JavaBeans" in *30th Annual Frontiers in Education Conference, 2000*, Kansas City, USA, 2000, vol. 1, pp. 7–8.

[24] S. Bhattacharya and D.A. Perry, "Contextual reusability metrics for event-based architectures" in *2005 International Symposium on Empirical Software Engineering*, Australia, 2005.

[25] S. D. Kim and Jihwan Park, "C-QM: A practical quality model for evaluating cots components" in *21st IASTED International Multi-Conference on Applied Informatics*, Innsbruck, Austria, 2003, vol. 21, pp. 991–996.

[26] N. W. Nerurkar, A. Sharma, and S. Sagar, "A soft computing based approach to estimate reusability of software components" *ACM SIGSOFT Softw. Eng. Notes*, vol. 35, no. 4, p. 1, Jul. 2010.

[27] Aman Jatain and Deepti Gaur, "Estimation of component reusability by identifying quality attributes of component: a fuzzy approach" in *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology*, Coimbatore, India, 2012, pp. 738–742.

[28] V. P. Venkatesan and M. Krishnamoorthy, "A Metrics Suite for Measuring Software Components" *J. Converg. Inf. Technol.*, vol. 4, no. 2, Jun. 2009.

[29] S. I. Zahara, M. Ilias, and T. Zia, "A study of comparative analysis of regression algorithms for reusability evaluation of object oriented based software components" in *2013 International Conference on Open Source Systems and Technologies (ICOSST)*, Lahore, Pakistan, 2013.

[30] P. S. Sandhu and H. Singh, "A Fuzzy-Inference System Based Approach for the Prediction of Quality of Reusable Software Components" in *ADCOM 2006. International Conference on Advanced Computing and Communications, 2006.*, Surathkal, 2006, pp. 349–352.

[31] P. Sandhu and H. Singh, "A Neuro-Fuzzy Based Software Reusability Evaluation System with Optimized Rule Selection" in *ICET '06. International Conference on Emerging Technologies, 2006.*, 2006, pp. 664–669.

[32] P. S. Sandhu, S. S. Dalwinder, and H. Singh, "A Comparative Analysis of Fuzzy, Neuro-Fuzzy and Fuzzy-GA Based Approaches for Software Reusability Evaluation." in *Proceedings of World Academy of Science: Engineering & Technology*, 2008.

[33] S. Manhas, P.S. Sandhu, V. Chopra, and N. Neeru, "Identification of reusable software modules in function oriented software systems using neural network based technique" *World Acad. Sci. Eng. Technol.*, vol. 43, pp. 823–827, 2010.

[34] P. S. Sandhu, H. Kaur, and A Singh, "Modeling of reusability of object oriented software system" *World Acad. Sci. Eng. Technol.*, vol. 56, pp. 162–165, 2009.

[35] A. Shri, P. S. Sandhu, V. Gupta, and S. Anand, "Prediction of reusability of object oriented software systems using clustering approach" *World Acad. Sci. Eng. Technol.*, vol. 43, pp. 853–856, 2010.

[36] G. Gui and P. D. Scott, "Measuring software component reusability by coupling and cohesion metrics" *J. Comput.*, 4 (9), 2009.

[37] J. Al Dallal, "Software similarity-based functional cohesion metric" *IET Softw.*, vol. 3, no. 1, pp. 46–57, 2009.

[38] Hyun Jung La, Jin Sun Her, and Soo Dong Kim, "Framework for evaluating reusability of Component-as-a-Service (CaaS)" in *2013 ICSE Workshop on Principles of Engineering Service-Oriented Systems (PESOS)*, San Francisco, USA, 2013, pp. 41–44.

[39] P. S. Sandhu and S. Chhabra, "A comparative analysis of conjugate gradient algorithms & PSO based neural network approaches for reusability evaluation of procedure based software systems" *Chiang Mai J. Sci.*, vol. 38, no. 2, pp. 123–135, 2011.

[40] M. Choi and S. Lee, "A Coupling Metric Applying the Characteristics of Components" in *Computational Science and Its Applications - ICCSA 2006*, vol. 3983, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 966–975.

[41] H. Washizaki, R. Namiki, T. Fukuoka, Y. Harada, and H. Watanabe, "A framework for measuring and evaluating program source code quality" in *Product-Focused Software Process Improvement*, vol. 4589, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 284–299.

[42] B. Ko and J. Park, “Component Architecture Redesigning Approach Using Component Metrics” in *Artificial Intelligence and Simulation*, vol. 3397, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 449–459.

[43] L.-G. Yu and S. Ramaswamy, “Component Dependency in Object-Oriented Software” *J. Comput. Sci. Technol.*, vol. 22, no. 3, pp. 379–386, May 2007.

[44] Varun Gupta and Jitender Kumar Chhabra, “Package level cohesion measurement in object-oriented software” *J. Braz. Comput. Soc.*, vol. 18, no. 3, pp. 251–266, Sep. 2012.

[45] M.W. Price, D.M. Needham, and S.A. Demurjian, “Producing reusable object-oriented components: A domain-and-organization-specific perspective” in *Proceedings of SSR’01 2001 Symposium on Software Reusability*, Toronto, Canada, 2001, pp. 41–50.

[46] Qi Wang, “Research on the Quantifying and Calculating Model of the Software Component Reusability” in *Future Control and Automation*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 103–109.

[47] Li Yingmei, Shao Jingbo, and Xia Weining, “On Reusability Metric Model for Software Component” in *Software Engineering and Knowledge Engineering: Theory and Practice*, vol. 114, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 865–870.

[48] Z. Wang, D. Zhan, and X. Xu, “A General Approach for Optimizing Degree of Variability of Software Components” *Inf. Technol. J.*, vol. 7, no. 3, pp. 474–481, Mar. 2008.

[49] P. S. Sandhu and H. Singh, “A Reusability Evaluation Model for OO-Based Software Components” *Int. J. Comput. Sci.*, vol. 1, no. 4, pp. 259–264, 2006.