Evolution of Object Oriented Coupling Metrics: A Sampling of 25 Years of Research

Ana Nicolaescu
RWTH Aachen University
52074 Aachen, Germany
ana.nicolaescu@swc.rwth-aachen.de

Horst Lichter
RWTH Aachen University
52074 Aachen, Germany
horst.lichter@swc.rwth-aachen.de

Yi Xu
RWTH Aachen University
52074 Aachen, Germany
yi.xu@rwth-aachen.de

Abstract—**Coupling is one of the most important properties that affect the quality of the design and implementation of a software system. In the context of object oriented software development, coupling metrics and their impact on quality attributes have been investigated for a quarter of a century. In this work we review and critically analyze the developments in this domain by considering 26 of the most influential research papers addressing object oriented coupling. Our analysis reveals that a very strong theoretical background has been already developed but unfortunately without a clear impact on the industry practices and software analysis tooling. Even more, recent developments fail to address this problem and seem to even contribute to increasing this gap. We argue that the direction of current research should be shifted towards systematizing and evaluating existing results rather than exploring new applicability domains and defining new metric suites.**

I. Introduction

In the 1990s, researchers realized that traditional metrics were only barely suitable for object-oriented (OO) programs and could not be adapted to OO concepts such as classes, inheritance and message passing (e.g. Wilde and Huitt [1]). Tegarden et al. and Bilow were among the first researchers who proposed to take OO concepts into account when designing new metrics [2], [3]. To the best of our knowledge, the first set of metrics developed for OO software was proposed by Morris [4] as early as in 1989 and already contained a coupling metric called *Degree of Coupling between Objects*. In the following, researchers started to develop metrics for OO software. For example, Coplien [5] as well as Pfleeger and Palmer [6] proposed some initial metrics for C++ programs and Rajaraman and Lyu were the first who proposed four coupling metrics for C++ programs in 1992 [7]. In 1994 Lorenz and Kidd published one of the first books on OO metrics presenting a set of eleven very different metrics [8].

Although there is a variety of properties that can be measured in OO programs, we focused in this paper on coupling metrics and aimed to present how OO coupling metrics have evolved since they were introduced. We chose this focus because coupling is by all means an important design property (also included in the Quality Model for Object-Oriented Design (QMOOD) [9]) that has a strong impact on several software quality attributes such as maintainability, testability or reusability. Meaningful coupling measurements should therefore be a necessary information to support both developers and architects to take the right decisions on, e.g., refactoring or re-engineering projects.

The remainder of this paper is organized as follows. In section 2 we briefly present how we have designed and performed our literature search. Based on the chosen timely grouping, the fundamental works on coupling metrics published in the 1990s will be discussed in section 3. In section 4 we present papers published in the 2000s mainly focusing on dynamic coupling metrics. Some recent directions are presented in section 5. Section 6 contains a (critical) discussion on about 25 years of research in this field and concludes the paper.

II. Search Strategy

Our search process consisted of defining the search expression, selecting digital libraries, executing a pilot search, refining the search expression, retrieving an initial list of papers matching the search expression and finally filtering the results.

In order to retrieve relevant published research works, we defined the following search expression ”software AND coupling OR (measurement OR metrics) AND (object OR class OR system OR component OR architecture) AND (OO OR object oriented)”. We restricted the publication period from 1990 until 2014. We applied the search expression to 10 databases and finally chose the following 6 databases based on the relevance of the results: IEEE Xplore Digital Library[1], ACM Digital Library[2], CiteSeerX[3], ScienceDirect[4], Google Scholar[5] and The Collection of Computer Science Bibliographies[6]. We sorted the results according to two criteria: number of citations and relevance. The “relevance” is typically defined as “the extent to which the retrieved articles match the search query”. We did not use a relevance algorithm of our own, but relied on the relevance computations offered by each considered database.

First, we searched the two citation databases, and got 210 results from The Collection of Computer Science Bibliographies and 418 papers from CiteSeerX. We read the top 50 titles and

[1]IEEE, http://ieeexplore.ieee.org/search/advsearch.jsp
[2]ACM, http://dl.acm.org/dl.cfm
[3]CiteSeerX, http://citeseerx.ist.psu.edu
[4]ScienceDirect, http://www.sciencedirect.com/
[5]Google Scholar, http://scholar.google.de/
[6]Collection, http://liinwww.ira.uka.de/bibliography

abstracts of each database, and divided the papers into three periods: papers published in the 1990s (fundamental works), the 2000s (advanced approaches), and from 2010 until 2014 (recent directions). Then, we queried the other four databases and sorted the results according to relevance. We retrieved 10625 results from IEEE Xplore Digital Library, 247 results from ACM Digital Library, 27300 results from Google Scholar and 42510 results from ScienceDirect. Finally, we filtered the results according to the different time periods and chose for our sampling 11, 9 and 6 papers for the three considered time periods. This choice is partially subjective. We chose more papers from the first time period, because it revealed itself as being very fruitful and impacting for the later research. The articles often build on top of each other, by systematizing and extending the previous work in the field. In contrast, we considered only 6 articles from the most recent time period because this period is shorter, the articles didn't yet have time to impact later research and, in our opinion, these latest developments are somewhat lacking pragmatism, as we will detail later.

III. Fundamental Works

In the following, we present *11 most influential coupling metrics papers* published in the 1990s, selected based on the citations counts and the relevance for our topic - as we deduced it by reading the title and abstract of the papers. First, we give an overview of the selected papers by shortly depicting their essence. Afterwards we summarize and present an initial analysis regarding the developments during this period of time.

A. Selected Papers

In 1991, Chidamber and Kemerer (CK) proposed a Metric Suite [10], [11] containing the most influential coupling metric *Coupling Between Object Classes* (CBO) as well as *Response For a Class* (RFC). The CBO value of a class is the number of classes that it is using and that it is used by. Hence, a class A is coupled to another class B, if A uses B's methods or attributes or vice versa. CBO does not take inheritance between classes into consideration. The publication of the CK Metric Suite was a landmark in the development of OO coupling metrics.

In 1994, Martin [12] published a set of metrics to measure stability (or instability) of a group of classes (called category). A highly stable category of classes is supposed to be reused easier in other contexts, as the classes are strongly interdependent and cannot be separated from each other. To measure instability, he introduced the two base metrics *Afferent Couplings* (Ca) - the number of classes outside a category that depend upon classes within this category, and *Efferent Couplings* (Ce) - the number of classes inside a category that depend upon classes outside this category. Based on these metrics, *instability* (I) is calculated as Ce / (Ca+Ce). Moreover, he presented a simple model using the properties instability and abstractness (defined as number of abstract classes in a category / total number of classes) to assess categories and to determine those categories whose abstractness is balanced with stability.

In the same year, Brito e Abreu and Carapuca [13] proposed the MOOD Metric Set containing the *Coupling Factor* (CF) metric. CF is a system-level metric, that considers the percentage of directed pairs of classes that have a "client-supplier relation" in comparison with the total number of possible (directed) connections in the overall system. A directed pair of classes is connected if the first one is "a client" of the second one, i.e., it accesses one of its methods or attributes.

Also in 1994, Eder et al. [14] introduced a uniform taxonomy or framework for coupling and cohesion, motivating that these concepts were not clearly defined in the OO context. They distinguished three types of coupling: *interaction coupling* (achieved through method invocation and/or sharing of data between methods), *component coupling* (one class contains a variable or parameter of another class) and *inheritance coupling*. Each of these types was further refined in so-called "degrees" that were mapped on an ordinal scale to show their contributions towards coupling increase.

In 1995, Hitz and Montazari [15] argued that internal product attributes (e.g. Number of Children) used to control important external quality attributes (e.g. maintainability) have to be carefully selected. They classified internal product attributes as *fundamental* or useful (having a causal relationship to external quality attributes), *auxiliary* (having a statistical correlation to external quality attributes) and *useless*. Finally they presented some examples of general design and coupling metrics (e.g. Message Passing Coupling and Change Dependency Between Classes), that measure useful internal product attributes and thus can be used to control respective external quality attributes.

As early as 1993, Lee and Henry [16] evaluated whether a set of metrics consisting of the CK Metric Suite, two size indicators (number of semicolumns in a class, and number of attributes and methods of a class) and an additional set of metrics proposed by themselves can be used to predict maintainability effort. The evaluation was performed on two commercial software systems developed in Ada. The results were positive: maintainability "seams to be predictable" using these metrics. Regarding the metrics themselves, the authors proposed two additional coupling metrics to complement the CK suite: *number of send-statements* in a class (to quantify "message passing coupling") and *number of abstract data types* defined in a class (to quantify the so-called "coupling through ADT").

Three years later, Basili et al. [17] performed an experiment to validate the ability of the CK Metric Suite to predict fault-proneness of classes. To this end they analyzed eight functionally equivalent C++ systems developed by teams of students. Their results indicated that the CBO value of a class significantly correlates with its fault-proneness, especially in the case of UI classes: the higher the CBO value is, the more error-prone the class might be. Furthermore, they also raised the question whether future work should focus on developing programming-language dependent metrics because these could correlate even better with the quality attributes of the systems developed with them.

Also in 1996, Brito e Abreu and Walcelio [18] evaluated the impact of OO design on software quality characteristics using the MOOD Metric Set [13]. They performed a controlled experiment presumably using Basili's eight C++ systems. The results showed that the MOOD metrics can potentially be used to predict defect density (a reliability measure) and rework (a maintainability measure). The authors concluded with emphasizing the need of further evaluation. They also mentioned the importance of studying whether the used programming language influences the results of such experiments.

One year later, Briand et al. provided a new suite of coupling metrics especially developed to address "the different design mechanisms provided by C++ " [19]. Similar as in the case of [17], one of the most important hypotheses that was explored and also confirmed was that coupling metrics expose a significant correlation with class fault-proneness. The metrics were designed to differentiate between the various modalities that can contribute to coupling when implementing in C++. These are *relationship*, *locus* and *type*. Relationship refers to the relationship-type between a pair of classes: none, inheritance or friendship. Locus refers to the flow of change impact: towards a class (import), i.e., a class has to be changed as a consequence of a change in another class, or away from a class (export), i.e., a change in a class leads to changes in other classes. Type refers to the interactions-type between classes: class-attribute, class-method, or method-method interaction. Given that there are 3 possible values of the relationship facet, 2 for the locus and 3 for the type, the authors defined 18 coupling metrics that consider the interactions of these different facets. While their evaluation (using again Basili's C++ systems) showed that their hypotheses might hold (high export coupling leads to domino effects when changing a class, high import coupling makes a class error-prone and hard to understand and friendship increases error-proneness), the authors emphasized the stringent need of further evaluations in industrial contexts.

As early as 1999, Briand et al. already remarked an explosion of the number of OO coupling metrics, their heterogeneity and lack of maturity: "many measures are not defined in a fully operational form, and relatively few of them are based on explicit empirical models" [20]. Thus, they proposed a framework to facilitate the comparison, evaluation and definition of already existing and new metrics, based on a common formalism and 6 classification criteria: *connection type* (method-class, method-attribute, etc.), *locus* (import vs. export), *measure granularity* (class, object, system, etc. and the strategy used to count the connections), *server stability* (unstable classes vs. stable classes), *direct vs. indirect connections* (counting only direct connections vs. transitively derived ones as well) and *inheritance* (inheritance- vs. non-inheritance-based coupling). The authors showed the applicability of the framework by using it to formalize the well known CK metrics.

In 1999, Allen and Khoshgoftaar [21] proposed an information theory approach to compute coupling and cohesion on system-level. A system is represented as a graph and its subsystems as partitions of the initial graph. They defined three metrics: *inter-module coupling*, *intra-module coupling*, and *cohesion*. These are computed by treating the pattern of edges incident on a given node in the graph as a random variable and then applying entropy and information measures. They argued that their approach is more useful than the counting-based approaches proposed previously and systematized in Briand's framework, described above. But, no evaluation of the approach in a real-life context was performed.

B. Summary

Since coupling and cohesion were not defined in the OO context, most of the influential works from this time period concentrated on defining and structuring these concepts and proposing metrics for their quantification. The well-known CK Metric Suite with its CBO metric and the MOOD metrics had a great impact on later research.

Furthermore, even at this early stage, researchers already remarked the explosion of new coupling metrics and proposed common meta-models, frameworks or taxonomies to compare already existing metrics and to define new future ones. Looking back from today's perspective, unfortunately these proposals did not hinder nor systematized the further expansion of this research domain, as our search has revealed.

It is remarkable to note that relatively many articles also discussed preliminary evaluations of the first published OO coupling metrics (e.g., the CK and the MOOD suites) and obtained positive results indicating that maintainability can be estimated using coupling measures. Nonetheless, a lot of emphasis was put on the need of further evaluations in industrial contexts.

Most of the approaches presented in the 1990s refer to the different types of coupling between classes, methods and attributes. However, coupling was also analyzed as a system-level metric and approaches regarding the measurement of coupling between subsystems were also considered. But, in all the cases the focus was on the static view of the system, so the measures were based either on analyzing design documents or actual source code.

IV. Advanced Approaches

In this section, we present *9 representative papers* on coupling metrics mostly published in the 2000s. Again we chose the papers based on the citations counts and the relevance for our topic.

A. Selected Papers

As early as 1999, Yacoub et al. [22] remarked that the "complex dynamic behavior of many real-time applications motivates a shift in interest from traditional static metrics to dynamic metrics". They introduced a suite of dynamic coupling metrics, sometimes also called run-time coupling metrics, to assess OO designs. To be able to capture dynamic metrics during the initial design time, they used the Real-Time Object Oriented Modeling (ROOM), an alternative to UML that provides means to define executable designs. The suite consists of one dynamic complexity metric and of two dynamic object-level coupling metrics: *Export Object Coupling* (EOC) and *Import Object Coupling* (IOC). The export and import coupling of an object A with respect to an object B is the percentage of the number of messages sent from A to B and received by A from B respectively with respect to the total number of exchanged messages during one execution scenario. They evaluated the metrics on the executable models of a cardiac pacemaker and showed that significant differences between static and dynamic coupling metrics can be identified. Therefore, they strongly militated for further empirical studies to analyze this difference as well as to understand the impact of the various types of metrics on external design.

In 2004 several proposals for new dynamic coupling metrics were published. Arisholm et al. [23] discussed about the existence of two types of run-time coupling: *object-level* and *class-level* coupling. These concepts do not completely overlap semantically, because of the intrinsic features of OO, e.g. polymorphism: "the class of the object sending or receiving a message may be different from the class implementing the corresponding method". Based on this remark they proposed 3 classification criteria for dynamic coupling metrics: *entity of measurement*, *granularity* and *scope*. The entity of measurement is either a class or an object. According to the chosen entity of measurement, the results can be aggregated to different granularities: the coupling values of all instances of a class can be aggregated to compute its coupling; the coupling values of some or all classes can be aggregated to further levels, e.g., sub-system or system. The scope determines which entities are to be accounted for or ignored when measuring dynamic coupling. In accordance with the framework proposed by Briand et al. [20] they then extended their previous work [24] and the work of Yacoub et al. [22] and formally defined and evaluated a series of export and import dynamic coupling metrics from the different perspectives mentioned above. According to their evaluation results on an open-source Apache system, the static and dynamic coupling metrics, although not unrelated, measure also different facets of the system. Especially dynamic export coupling measures were regarded as very strong indicators of class change-proneness especially when complemented by size and static coupling measures.

Mitchell and Power [25] also argued that "static metrics fail to quantify all the underlying dimensions of coupling". They proposed 4 run-time extensions of the CBO metric. The *run-time import* and *export coupling metrics* denote the number of times a class accesses and is being accessed by other classes respectively. The *run-time import* and *export degree metrics* denote the percentage of outgoing and incoming accesses w.r.t. the total number of accesses. As in the previous cases, their results obtained from an evaluation performed on several Java programs from the SPEC JVM98 benchmark "seem to suggest that the run-time coupling metrics are not redundant with the static CBO metric".

Hassoun et al. [26] observed object coupling as it evolves during program execution. They proposed the so-called *Dynamic Coupling Metric* (DCM). The DCM value of an object P depends on two factors: a time-factor (1) expressed as a set of discrete execution steps during which the state of the object P changes and the complexity (2) of the objects that are coupled with P during each execution step in this set. The DCM can be extended to compute the coupling of the entire system at run-time. The authors specifically stressed the importance of analyzing object coupling in reflective systems, because in such systems "objects' coupling are run-time dependent and may change due to customizations of objects' behavior or modifications of their structure". However, apart from relatively small examples and an analytical evaluation in which the properties of the DCM are compared with those recommended by Briand et al. [20], no evaluation of the metric on a real-world software system was provided.

Zaidman and Demeyer [27] defined the *Object Request For Service* (OQFS) and the *Class Request for Service* (CQFS) metrics - as two variants of the EOC metric proposed by Yacoub et al. [22] - to suggest entry points for further collection and analysis of event traces in the context of large-scale industrial systems. The OQFS metric gives the total number of unique messages that an object sends during run-time, while the CQFS metric aggregates the unique messages at the class-level. The authors argue that the objects or classes with the highest values for OQFS or CQFS respectively should be proposed as entry points for system comprehension. An evaluation of the validity of these heuristics was missing and proposed as future work.

In 2005, Mitchell and Power proposed a further set of CBO-based dynamic metrics to study the coupling at object-level [28]. The purpose of their study was to validate or invalidate the hypothesis that objects belonging to the same class have similar couplings during run-time. To this end, they proposed two dynamic coupling metrics at two different abstraction levels. *Run-time CBO* (RCBO) is a class-class level metric that simply computes the CBO for each object based on the number of accesses at run-time. The second metric operates on the object-class level and applies agglomerative hierarchical clustering to check if clusters of objects can be identified that although belonging to the same class, exhibit non-uniform behavior coupling at run-time. Their evaluation performed on the JOlden benchmark invalidated their hypothesis: objects belonging to the same class are not necessarily coupled similarly during run-time. The authors drew the attention on the need of further evaluation and reiterated on the importance of the chosen input data for the results of any dynamic metrics.

One year later, Mitchell and Power further explored the difference between the CBO metric, its dynamic counterparts and the influence that the instruction coverage has on this difference [29]. While previous results ([25], [28], [23]) showed that static and dynamic metrics complement each other, the influence of the actually executed code was not thoroughly tested. The evaluation was conducted using the dynamic metrics proposed by Arisholm et al. [23] and analyzing 14 Java programs from the SPEC JVM98 and JOlden benchmarks. The results showed that the static CBO together with the instruction coverage are better predictors for some of the dynamic coupling metrics than static CBO alone. In particular, the dynamic import-based coupling metrics can be better predicted than the export-based coupling metrics by this combination of metrics. As future research directions, the authors suggest to use dynamic metrics for quantifying the effectiveness of various test strategies, further analyzing the connection between static and dynamic metrics with respect to external design quality and sustaining re-engineering and program-understanding endeavors.

Deviating from the main research stream busy with defining and evaluating dynamic coupling metrics, Gyimothy et al. [30] applied the CK metrics in 2005 to identify fault-prone source code in the Mozilla system. They compared their predictions with the actually registered bugs in Bugzilla and used statistical and machine-learning techniques to validate their usefulness. Among other evaluation results, the authors showed that among all the CK metrics, the CBO metric correlates best with fault-proneness and can be used as a reliable predictor.

Already in 2002, Bansiya and Davis [9] proposed a hierarchical quality model called QMOOD to assess the high-level design quality attributes (reusability, flexibility, understandability, functionality, extensibility and effectiveness) during design-time. QMOOD defines four levels. The first level, L1, consists of the considered 6 design quality attributes. The L2 level contains 11 design properties (e.g., coupling, cohesion, inheritance) that positively or negatively influence the elements on L1 to different, weighted extents. Level L3 consists of metrics to determine the values of the L2 properties. These metrics operate on elements contained in level L4 - the basic OO design elements (i.e., methods, attributes, classes, etc.). QMOOD defines a new coupling metric since the authors argued that the previously developed ones require an implemented system, while QMOOD assesses the system at design time. The resulting coupling value is then weighted to negatively influence reusability, flexibility, understandability and extensibility. The very positive evaluation results of applying QMOOD to validate and compare a set of industrial systems can be seen as yet another indicator that coupling does correlate with the above mentioned quality attributes.

B. Summary

Related research in the 2000s decade was clearly predominated by proposals and evaluations of new, dynamic metrics. The initial, widely-agreed upon motivation to develop dynamic coupling metrics was that the "features of object-oriented programming such as polymorphism, dynamic binding and inheritance render the static coupling metrics imprecise" [25]. Interestingly, most of the new metrics ([23], [26], [25], [28], [29]) were proposed according to the principles previously formulated in Briand's framework [20].

Several dynamic variants of already known static coupling metrics and especially of CBO were developed. These typically extended CBO by augmenting it with direction (export vs. import coupling), and/or normalized strength notions, [25] and/or by applying it on different granularity levels: object-level, class-level or system-level ([23], [26]). In this context, it was even stated that the name of CBO is misleading since it does not measure the coupling between objects, but between classes [22].

Efforts were also invested in analyzing if the static and dynamic metrics are redundant or complement each other. By applying techniques from, e.g., descriptive statistics, correlation and principal component analysis, different papers ([22], [23], [25], [29]) indicated similar results: while a relation between static and dynamic metrics exists, they also measure different facets of the system and can be used in conjunction to better predict important properties, such as class fault-proneness.

Interestingly, in parallel with the development of more sophisticated dynamic coupling metrics, very good fault-proneness prediction results were achieved using the original CK metrics and especially CBO [30]. The quality model proposed by Bansiya and Davis [9] is another important milestone, since it considered the interplay of more design properties (among which coupling is as well) to assess various design qualities and proved within an extensive evaluation that a hierarchical model is very feasible for this goal.

V. Recent Directions

Finally, we present several recent papers on OO coupling metrics published in the last 5 years. Obviously, these papers are not so often cited than the older papers. Therefore, we sorted the query result according to relevance and we chose *6 papers* that we considered suitable to depict the trend of recent advances.

A. Selected Papers

Aloysius and Arockiam [31] presented a new complexity metric called *Cognitive Weighted Coupling Between Objects* (CWCBO) for measuring the complexity of classes. It considers among others the cognitive complexity (based on the average comprehension time) of the five types of coupling introduced by Berard [32] in 1993 (control coupling, global data coupling, internal data coupling, data coupling and lexical content coupling). In order to calibrate the weight values of these types of coupling they applied psychological experiments. Furthermore they performed a study to validate the metric compared to the CBO metric and showed that the CWCBO metric is a much better complexity indicator than CBO.

Kebir et al. [33] used coupling and cohesion metrics to identify components based on three properties of components. First, a component is autonomous if it has no required interface. Second, a component can be composed by means of its provided and required interfaces. Third, a component which provides many interfaces may provide various functionalities. Then they matched these properties to a set of new and existing metrics and validated their component identification approach using open source systems.

Rathore and Gupta [34] performed a controlled experiment to identify correlations between design properties (measured by means of class-level metrics) and fault-proneness of classes. Beside existing coupling metrics (e.g., CBO, RFC) they applied cohesion, inheritance, size and complexity metrics as well. They were able to confirm one of their hypotheses that classes with high coupling are more likely to be fault prone.

Alshammari et al.[35] proposed a suite of security related metrics including a metric called *Critical Classes Coupling* (CCC) to measure the coupling between classes based on design models such as UML class diagrams. The aim was to deliver indicators of strongly coupled systems, because the hypothesis of the authors was that systems with strong coupling are a greater target for successful attacks than systems with loose coupling. CCC aims to calculate the degree of coupling between classes and classified attributes in a given software design. They illustrated the application of the metric suite in a case study where an initial design was stepwise improved (regarding security issues) based on the calculated metric values.

Gethers and Poshyvanyk [36] found out that many existing coupling metrics lack the ability to identify conceptual dependencies, which could specify underlying relationships encoded by developers in identifiers and comments of source code. They proposed the class-level coupling metric *Relational Topic-based Coupling* (RTC), using the Relational Topic Model (RTM) to identify latent topics associated with source code. If RTM identifies a link between two classes with a high probability, these classes are considered to be coupled. Finally they validated the RTC metric against nine well-known static coupling metrics, e.g. CBO, RFC and DAC, using a large set of open source software. They could show that RTC is able to explain a dimension in the data that is ignored by existing coupling metrics.

Chen et al. [37] argued that traditional metrics are not suitable to measure complexity in component-based software system (CBSS). Therefore, they provided new component-level metrics to measure coupling (MV), cohesion (COM), and interface density (AIM) of components, used to calculate complexity. Two components are coupled if and only if at least one of them acts upon the other. The introduced coupling metric (MV) takes into account the usage of methods and variables. Based on the component-level metrics respective system-level metrics are presented. Unfortunately, the application of their metrics is shown only using an artificial toy-like CBSS example.

B. *Summary*

Although new advanced coupling metrics have been proposed (e.g., [36] for conceptual coupling), researchers now mainly focus on applying existing metrics for very different purposes such as the identification of components in large code bases or the discovery of security issues.

VI. Discussion

In this paper, we first conducted a literature review of 26 relevant and often cited research papers published in the last 25 years regarding the topic of object oriented coupling metrics. We divided the selected papers into three time periods: the "fundamentals" works era (1990 - 1999), the "advanced approaches" era (2000 - 2010) and the "recent directions". For each era, we chose a set of representative works, presented an overview thereof and finally conducted an analysis regarding the major identified trends.

In the first two eras, a relatively extensive knowledge base was constructed and the relevance of both static and dynamic coupling metrics was often acknowledged. Despite this apparent strong academic background, our practical experience with defining metric systems ([38], [39]) for a variety of industrial cooperation partners showed that both static and dynamic coupling metrics are ignored and not used to drive important decisions despite their proven predictability for quality attributes such as maintainability and reliability. Commonly used static analysis tools also at most simply list the computed values of some static coupling metrics but do not further use them to predict quality (e.g., JDepend[7] and Sonargraph[8] list the values of the Martin's metrics, Stan4J[9] additionaly also compute the CK metrics, while other tools such as Clover[10] or SonarQube[11] do not offer such measures at all). The situation is even more acute in the case of the dynamic coupling metrics. Most of the tools for dynamic analysis, no matter if industrial or academic, ignore these metrics (e.g., Kieker[12]). Those who do regard them, either define new ones and/or use them fust for system comprehension (e.g., ARAMIS [40]), but do not offer any impact analysis of the results.

Unfortunately, the later developments are not seeking to alleviate the situation described above but are often just concerned with exploring new applicability domains of existing metrics (e.g., security) or defining new, sometimes "very exotic" metrics, rarely probed in real-life software projects. This trend is very worrying as it embodies one of the most crucial problems of today's software engineering research: its lack of relevance for the software development industry! Probably just as worrying is the fact that this situation has resulted, although the importance of proper evaluation has been highlighted as early as during the emergence of the first, fundamental OO metrics "metrics must be collected and analyzed throughout time in as many different projects as possible in order to establish comparisons and derive conclusion". [13]. However, recent developments expose the same problem as almost 20 years ago: "our understanding of existing coupling measures is not what it should be" [20]. The present situation is even more dramatic, if we take into consideration that the previous cited statement was made in a context where only 30 OO coupling metrics had been developed by the research community, which is almost insignificant compared to the number of currently available OO coupling metrics, as our search process has revealed.

[7]http://clarkware.com/software/JDepend.html
[8]https://www.hello2morrow.com/products/sonargraph/architect
[9]http://stan4j.com/metrics/quality-metrics.html
[10]https://de.atlassian.com/software/clover/overview
[11]http://docs.sonarqube.org/display/SONAR/Metric+definitions
[12]http://kieker-monitoring.net/features/

VII. Future Work

An overview of all relevant coupling metrics is still missing. In the future, we strongly believe that more effort should be shifted towards reviewing and re-evaluating already existing metrics rather than investing further resources in defining new ones. A coupling metrics catalog similar to catalogs of design patterns, would be helpful both for researchers and practitioners to guide the selection of coupling metrics that best fit their purpose.

References

[1] N. Wilde and R. Huitt, "Maintenance support for object-oriented programs," *IEEE Transactions on Software Engineering*, vol. 18, no. 12, pp. 1038–1044, Dec. 1992.

[2] D. P. Tegarden, S. D. Sheetz, and D. E. Monarchi, "Effectiveness of traditional software metrics for object-oriented systems," in *Proceedings of the 25th Hawaii International Conference on System Sciences*, vol. 4. IEEE, 1992, pp. 359–368.

[3] S. C. Billow, "Applying graph-theoretic analysis models to object oriented system models," in *Workshop on Metrics for Object Oriented Software Engineering*, 1992.

[4] K. Morris, "Metrics for object oriented software development," Master Thesis, MIT, Sloan School of Management, Cambridge, MA, 1989.

[5] J. Coplien, "Looking over one's shoulder at a c++ program," *AT&T Bell Labs. Tech. Memo*, 1993.

[6] S. Pfleeger and J. Palmer, "Software estimation for object-oriented systems," in *Proceedings of International Function Point Users Group Fall Conference*, 1990, pp. 181–196.

[7] C. Rajaraman and M. R. Lyu, "Some coupling measures for c++ programs," in *Proceedings of the 8th International Conference on Technology of Object Oriented Languages and Systems*, ser. TOOLS 8, 1992, pp. 225–234.

[8] M. Lorenz and J. Kidd, *Object-Oriented Software Metrics*. Englewood Cliffs, New Jersey, USA: Prentice-Hall, Inc., 1994.

[9] J. Bansiya and C. G. Davis, "A hierarchical model for object-oriented design quality assessment," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 4–17, 2002.

[10] S. R. Chidamber and C. F. Kemerer, "Towards a metrics suite for object oriented design," in *Conference Proceedings on Object-oriented Programming Systems, Languages, and Applications*, ser. OOPSLA '91. New York, NY, USA: ACM, 1991, pp. 197–211.

[11] ——, "A metrics suite for object-oriented design," *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476–493, 1994.

[12] R. Martin, "Oo design quality metrics - an analysis of dependencies," in *Proceedings of Workshop on Pragmatic and Theoretical Directions in Object-Oriented Software Metrics, OOPSLA*, 1994.

[13] F. Brito e Abreu and R. Carapuca, "Object-oriented software engineering: Measuring and controlling the development process," in *Proceedings of 4th International Conference On Software Quality*, 1994.

[14] J. Eder, G. Kappel, and M. Schrefl, "Coupling and cohesion in object-oriented systems," *Technical Reprot, University of Klagenfurt, Austria*, 1994.

[15] M. Hitz and B. Montazeri, "Measuring product attributes of object-oriented systems," in *Proceedings of 5th European Software Engineering Conference*. Springer Verlag, 1995, pp. 124–136.

[16] W. Li and S. Henry, "Maintenance metrics for the object oriented paradigm," in *Proceedings of International Symposium on Software Metrics*. IEEE, 1993, pp. 52–60.

[17] V. R. Basili, L. C. Briand, and W. L. Melo, "A validation of object-oriented design metrics as quality indicators," *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 751–761, Oct. 1996.

[18] F. Brito e Abreu and W. Melo, "Evaluating the impact of object-oriented design on software quality," in *Proceedings of 3rd International Symposium on Software Metrics: From Measurement to Empirical Results*, ser. METRICS '96. Washington, DC, USA: IEEE Computer Society, 1996, pp. 90–99.

[19] L. Briand, P. Devanbu, and W. Melo, "An investigation into coupling measures for c++," in *Proceedings of 19th International Conference on Software engineering*. ACM, 1997, pp. 412–421.

[20] L. Briand, J. Daly, and J. Wüst, "A unified framework for coupling measurement in object-oriented systems," *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91–121, Jan. 1999.

[21] E. B. Allen and T. M. Khoshgoftaar, "Measuring coupling and cohesion: An information-theory approach," in *Proceedings of 6th International Symposium on Software Metrics*. IEEE, 1999, pp. 119–127.

[22] S. M. Yacoub, H. H. Ammar, and T. Robinson, "Dynamic metrics for object oriented designs," in *Proceedings of 6th International Symposium on Software Metrics*. IEEE, 1999, pp. 50–61.

[23] E. Arisholm, L. C. Briand, and A. Foyen, "Dynamic coupling measurement for object-oriented software," *IEEE Transactions on Software Engineering*, vol. 30, no. 8, pp. 491–506, 2004.

[24] E. Arisholm, "Dynamic coupling measures for object-oriented software," in *Proceedings of 8th IEEE International Software Metrics Symposium*, 2002, pp. 33–42.

[25] Á. Mitchell and J. F. Power, "An empirical investigation into the dimensions of run-time coupling in java programs," in *Proceedings of 3rd International Symposium on Principles and Practice of Programming in Java*. Trinity College Dublin, 2004, pp. 9–14.

[26] Y. Hassoun, R. Johnson, and S. Counsell, "A dynamic runtime coupling metric for meta-level architectures," in *Proceedings of 8th European Conference on Software Maintenance and Reengineering*. IEEE, 2004, pp. 339–346.

[27] A. Zaidman and S. Demeyer, "Analyzing large event traces with the help of coupling metrics," in *Proceedings of 5th International Workshop on OO Reengineering*, 2004.

[28] Á. Mitchell and J. F. Power, "Using object-level run-time metrics to study coupling between objects," in *Proceedings of ACM Symposium on Applied computing*. ACM, 2005, pp. 1456–1462.

[29] ——, "A study of the influence of coverage on the relationship between static and dynamic coupling metrics," *Science of Computer Programming*, vol. 59, no. 1, pp. 4–25, 2006.

[30] T. Gyimothy, R. Ferenc, and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Transactions on Software Engineering*, vol. 31, no. 10, pp. 897–910, October 2005.

[31] A. Aloysius and L. Arockiam, "Coupling complexity metric: A cognitive approach," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 4, no. 9, p. 29, 2012.

[32] E. V. Berard, *Essays on Object-oriented Software Engineering (Vol. 1)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[33] S. Kebir, A.-D. Seriai, S. Chardigny, and A. Chaoui, "Quality-centric approach for software component identification from object-oriented code," in *Proceedings of the 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture*, ser. WICSA-ECSA '12, 2012, pp. 181–190.

[34] S. Rathore and A. Gupta, "Investigating object-oriented design metrics to predict fault-proneness of software modules," in *Proceedings of 6th International Conference Series on Software Engineering (CONSEG)*, 2012, pp. 1–10.

[35] B. Alshammari, C. Fidge, and D. Corney, "Security metrics for object-oriented designs," in *Proceedings of 21st Australian Software Engineering Conference (ASWEC)*, 2010, pp. 55–64.

[36] M. Gethers and D. Poshyvanyk, "Using relational topic models to capture coupling among classes in object-oriented software systems," in *Proceedings of IEEE International Conference on Software Maintenance (ICSM)*, 2010, pp. 1–10.

[37] J. Chen, H. Wang, Y. Zhou, and S. D. Bruda, "Complexity metrics for component-based software systems," *International Journal of Digital Content Technology and its Applications*, vol. 5, no. 3, pp. 235–244, 2011.

[38] M. Vianden, H. Lichter, and S. Jeners, "History and Lessons Learnt from a Metrics Program at a CMMI Level 3 Company," in *Proceedings of 20th Asia-Pacific Software Engineering Conference, APSEC 2013, Vol. 2*, no. CMMI, 2013.

[39] A. Hutter, "Business Integration of Metric-Dashboards and Dashboard-Templates for existing Software-Projects (in German)," Diploma Thesis, RWTH Aachen University, 2013.

[40] A. Dragomir, H. Lichter, J. Dohmen, and H. Chen, "Run-time monitoring-based evaluation and communication integrity validation of software architectures," in *21th Asia-Pacific Software Engineering Conference (to be published)*, 2014.