A Survey on Software Coupling Relations and Tools

Enrico Fregnan*

University of Zurich, Switzerland

Tobias Baum

Leibniz Universität Hannover, Germany

Fabio Palomba and Alberto Bacchelli

University of Zurich, Switzerland

Abstract

Context: Coupling relations reflect the dependencies between software entities and can be used to assess the quality of a program. For this reason, a vast amount of them has been developed, together with tools to compute their related metrics. However, this makes the coupling measures suitable for a given application challenging to find.

Goals: The first objective of this work is to provide a classification of the different kinds of coupling relations, together with the metrics to measure them. The second consists in presenting an overview of the tools proposed until now by the software engineering academic community to extract these metrics.

Method: This work constitutes a systematic literature review in software engineering. To retrieve the referenced publications, publicly available scientific research databases were used. These sources were queried using keywords inherent to software coupling. We included publications from the period 2002 to 2017 and highly cited earlier publications. A snowballing technique was used to

*Corresponding author

Email addresses: fregnan@ifi.uzh.ch (Enrico Fregnan), tobias.baum@inf.uni-hannover.de (Tobias Baum), palomba@ifi.uzh.ch, bacchelli@ifi.uzh.ch (Fabio Palomba and Alberto Bacchelli)

retrieve further related material.

Results: Four groups of coupling relations were found: structural, dynamic, semantic and logical. A fifth set of coupling relations includes approaches too recent to be considered an independent group and measures developed for specific environments. The investigation also retrieved tools that extract the metrics belonging to each coupling group.

Conclusion: This study shows the directions followed by the research on software coupling: e.g., developing metrics for specific environments. Concerning the metric tools, three trends have emerged in recent years: use of visualization techniques, extensibility and scalability. Finally, some coupling metrics applications were presented (e.g., code smell detection), indicating possible future research directions.

Keywords: Software Engineering; Coupling relations; Software metrics

2010 MSC: 00-01, 99-00

---

1. Introduction

Software development is a complex task that requires careful planning and a high amount of time and energy [1]. Furthermore, maintainability [2, 3] and reliability [4] are important qualities that software should possess. To assess these properties, software complexity measures (coupling and cohesion) were introduced [5, 6]. As defined by Robbes et al. [7], coupling measures the amount of dependency between entities in a software. Over the years, different coupling measures have been proposed. Starting from structural metrics developed for procedural languages [5], new approaches were introduced to measure different relations in object-oriented environments [8]. Nonetheless, the central importance of these metrics for software engineering encouraged researchers to give birth to even more coupling measures in the attempt to evaluate further connections between software entities [9]. Excluding the already existing structural coupling, three new groups of coupling relations were created: *dynamic*, *se-*

mantic, and *logical* coupling. Dynamic coupling analyzes the run-time relations among different software entities [10]. Semantic coupling exploits the semantic relations among different elements in the source code using information retrieval techniques [11]. Finally, logical coupling approaches work by finding relations among system parts that are frequently changed together [12].

Due to the flourishing of this research field, a vast amount of original coupling measures have been proposed. However, all these different approaches can make it difficult for a software engineer to find the proper coupling relations to test the quality of the software on which he or she is working on. Some coupling relations can be applied only to particular groups of programming languages such as the object-oriented ones. Other metrics reveal themselves useful in specific situations: for example, evolutionary coupling is particularly helpful to highlight software changes. For these reasons, this work aims to provide a taxonomy of the coupling relations proposed so far, categorizing them in different groups and highlighting the commonalities and differences among them. Special attention has been given to the various trends that emerged in this field so far, highlighting the motivations behind the construction of new coupling relations. We argue that this study constitutes a good overview of software coupling relations and a starting point for further research in this field. Furthermore, we compare different tools developed by researchers to extract these relations in terms of output and required input information. The goal of this second part of our literature review is complementary to the first one. We argue that a researcher, having identified the coupling metrics of interest, may also be interested in which tools he/she may use to extract them. To the best of our knowledge, this constitutes a new contribution to the existing literature.

Other systematic reviews on coupling relations have been done by Kirbas et al. [12] and Nicolaescu et al. [13]. However, they have a different aim: Kirbas et al. limit their review to the field of logical coupling, while Nicolaescu et al. organize it in chronological order. The work of Kirbas et al. uses a measurement theory perspective to analyze logical coupling measures. This approach is reflected in the research questions identified in the study: questions like "*Do* *existing studies use a sound empirical relation system?*" or "*Do existing studies define measurement methods and procedures?*" show the authors' focus on evaluating how well logical coupling is currently captured by the different measures proposed. However, such an approach is not easily applicable to the broader perspective of our review. We consider logical coupling measures only as a subgroup of all the proposed coupling ones. Our review's goal is to give a general classification of all the possible coupling measures introduced in the software engineering field and not to analyze in details the properties of a specific subgroup. For this reason, the focus on logical coupling is limited to an overview of the different measures introduced to assess it.

Nicolaescu et al. propose an analysis of the main trends of coupling metrics for object-oriented systems, considering both the fundamental research done in the field and new directions that have been explored in recent years. Although their work constitutes an extraordinary attempt to present an overview of this complex research area, they report the different proposed coupling metrics in chronological order instead of dividing them into groups. In fact, Nicolaescu et al. analyzed 26 research papers dividing them in three time periods: *fundamental works* (1990-1999), *advanced approaches* (2000-2010) and *recent directions* (2011-2015). On the contrary, in our work the main focus is to give a conceptual subdivision of the different coupling relations. In fact, our main concern is not the period in which the considered metrics have been proposed (although, if possible, we keep a chronological order for clarity), but the different rationales behind them, which gave birth to their classification.

2. Research questions

Coupling relations have fundamental importance in software development since they are useful in activities such as, among others, maintenance and program comprehension [7]. For this reason, researchers explored links between software entities in the attempt to capture different characteristics of software to ensure its quality [9]. Nonetheless, a systematic classification of these different techniques is still missing. Therefore, in this work we first answer the following research question:

> ***RQ1*** *Which different coupling relations have been proposed by the software engineering research community?*

The goal of **RQ1** is to produce a taxonomy of the existing coupling relations, describing their core points together with the novelty that they introduce. Furthermore, the differences between them will be presented. Then, we will investigate different metric tools based on the relations found in **RQ1** with their outcome and input information. For this reason, our next research question is the following:

> ***RQ2*** *Which tools to extract coupling metrics have been developed by the software engineering research community?*

RQ2 aims to retrieve the tools that the software engineering research community has developed to extract different coupling relations. We will classify them based on the taxonomy produced by answering **RQ1**. Moreover, their different inputs and outputs will be highlighted, together with their limitations: e.g., the programming languages to which they are restricted.

3. Research strategy

In our investigation, we followed the guidelines given by Kitchenham [14]. Figure 1 shows the steps of our research strategy. To address **RQ1**, we conducted an initial query to evaluate the goodness of our approach. Based on the papers retrieved, in particular, the work by Bavota et al. [9], we refined our query including terms specific for each coupling group. Moreover, we checked for alternative spellings and synonyms. The terms identified were:

- Structural coupling
- Dynamic coupling

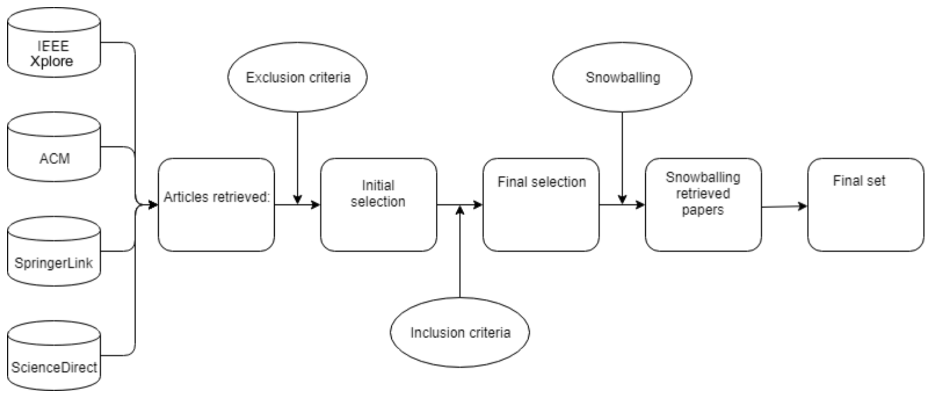

Figure 1: Methodology steps

- (Semantic or Conceptual) coupling
- (Logical or evolutionary or change) coupling

Finally, to investigate our first research question (**RQ1**) we combined using boolean operators all terms identified to create the following search string:

> *(Software AND coupling) OR (coupling AND object-oriented) OR (software AND coupling AND ((logical OR evolutionary OR change) OR (semantic OR conceptual) OR dynamic OR structural))*

We included the word "software" to reduce the number of results from research fields other than software engineering. However, this was not necessary when we used terms proper to the computer science area such as "object-oriented". The same procedure was applied to develop a search string for our second research question (**RQ2**).

> *(Coupling AND tool) AND (metrics OR (logical OR evolutionary OR change) OR (semantic OR conceptual) OR dynamic OR structural)*

These two research strings were used to investigate the following resources:

- IEEE Xplore Digital Library[1]
- ACM Digital Library[2]
- SpringerLink[3]
- ScienceDirect[4]

Table 1 shows the size of the papers set retrieved at each step of our investigation. *ScienceDirect* and *SpringerLink* returned a number of results too vast for an accurate analysis (more than 200’000 results). To restrict this set, we filtered the journals to the ones on computer science and software/software engineering and then we applied our queries to each of them (complete list available in Appendix A). In SpringerLink, we excluded the “preview-only” content.

Table 1: Data sources and search results

Source	**Total results retrieved**[5]	**Initial selection**	**Final selection**
IEEE Xplore Digital Library	10391 + 1639	69	65
ACM Digital Library	2838 + 1120	+13	+13
ScienceDirect	2887 + 2014	+9	+8
SpringerLink Digital Library	3733 + 2967	+2	+2
Total		93	88

The retrieved papers were evaluated based on a set of exclusion and inclusion criteria. The exclusion criteria were:

- Articles that do not focus on software coupling relations and/or tools.
- Articles that were not written in English.
- Articles whose full text is not available.

[1]IEEE, http://ieeexplore.ieee.org/Xplore/home.jsp
[2]ACM, https://dl.acm.org/
[3]SpringerLink, https://link.springer.com
[4]ScienceDirect, http://www.sciencedirect.com
[5]We reported the number of results obtained with the first query and the number of results obtained with the second one. We did not compute their sum since the two queries presented overlapping results.

- Not peer-reviewed articles, e.g., Ph.D. or M.Sc. thesis

To assess the quality of the retrieved research material, in other words, if the papers identified by our queries contained information useful to answer our research questions (respecting the first criterion), the following three-step procedure was applied. In the first step, the papers' titles and abstracts were carefully read to exclude the ones clearly irrelevant to the focus of our research. The second step consisted in skimming the whole text of the material left after the first selection to assure that it contained information related to coupling relations, measures and/or tools to extract them. Finally, the third step was an accurate reading through the whole text to ensure that this information was effectively helpful to address the two research questions: for the tools, we checked that their input, output and limitations were described.

At the same time, we applied the following inclusion criteria:

- Year of publication: only papers published between 2002 and 2017 were accepted;
  OR
- Number of citations: only papers referenced by more than 100 other publications were accepted;

The two criteria do not have to be valid simultaneously: they are connected with the logical operator OR. Therefore, a paper is selected if it meets at least one of the two criteria. The first criterion was selected to include recent publications on the topic. We included in our work only papers published in the last 15 years at the moment on which this research is conducted: from January 2002 to December 2017. However, we argue that this criterion may have excluded fundamental papers on coupling. Although they have been published before 2002, their contribution could be fundamental to obtain insights on trends and characteristics of more recent metrics. For this reason, we introduced the second criterion to augment the first one. We selected 100 citations as threshold because we were interested in the analysis of solid and well-established resources on the topic of interest. The number of citations have been attested using Google Scholar to have a verification system independent from the single data source. However, we recognize the potential limits of this approach: a paper with a high number of citations is not necessarily an important paper on the subject. To mitigate this problem, we complemented this research strategy with a snowballing technique [15]. We applied forward and backward snowballing on all the papers included in our final selection set until saturation was reached. The purpose of using the snowballing technique was to compensate for fundamental material that may have been left out by our previously mentioned search queries.

Table 2: Total of papers retrieved

Final set	**Snowballing**	**Total**
88	48	136

As a further check on the goodness of the retrieved material and its correct use, we contacted authors of other similar works on coupling or studies published in 2016 and 2017. We reached 16 authors and received 5 answers. We asked them to check if their papers were cited correctly and suggest us any other published work relevant for our review. However, we did not discover any new source that was not already included in the set of papers found by means of our search process.

4. RQ1: Coupling relations

In this section different existing coupling relations and techniques will be presented, based on the result obtained investigating our first research question.

As stated by Briand et al. "*Coupling refers to the degree on interdependence among the components of a software system*" [16]. A component can be a module of the system or a smaller entity such as a class or an object. Moreover, coupling can indicate a relation between two components but also a property of an entity compared to all the other related entities in the system: e.g., CCBC (Conceptual

Coupling Between Classes) and CoCC (Conceptual Coupling of a Class) [17]. Bavota et al. identified four different measures of coupling [9]:

- *Structural* coupling
- *Dynamic* coupling
- *Semantic* coupling
- *Logical* coupling

Structural coupling exploits the static relations in the source code: it focuses on finding patterns such as called methods, relations among classes (inheritance and friendship) and accessed variables. *Dynamic coupling* also reflects calls between classes and methods but it does that at run-time, instead of looking at the static code. *Semantic coupling* relies on Information Retrieval techniques to find relations in the code lexicon, while *logical coupling* intends to assess the entities that are frequently changed together, and therefore share a link, using historical information. Finally, other approaches try to combine these groups of relations in a complementary way or present coupling measures for domain-specific programming languages. Figure 2 shows an overview of the coupling relation taxonomy that we have constructed in our review. The goal of this first part of our work is to present the evolution of the coupling relations and metrics proposed, while keeping intact the categories presented by Bavota et al. [9].

4.1. Structural coupling

Structural coupling relations exploit static connections among software entities (modules, objects or classes). Measures to assess them have been initially developed for procedural languages, but, later, extended to the object-oriented paradigm. Furthermore, some structural coupling relations have been proposed specifically for object-oriented languages. In general, it is possible to divide them into two broad groups: procedural programming coupling measures and object-oriented coupling measures [18].

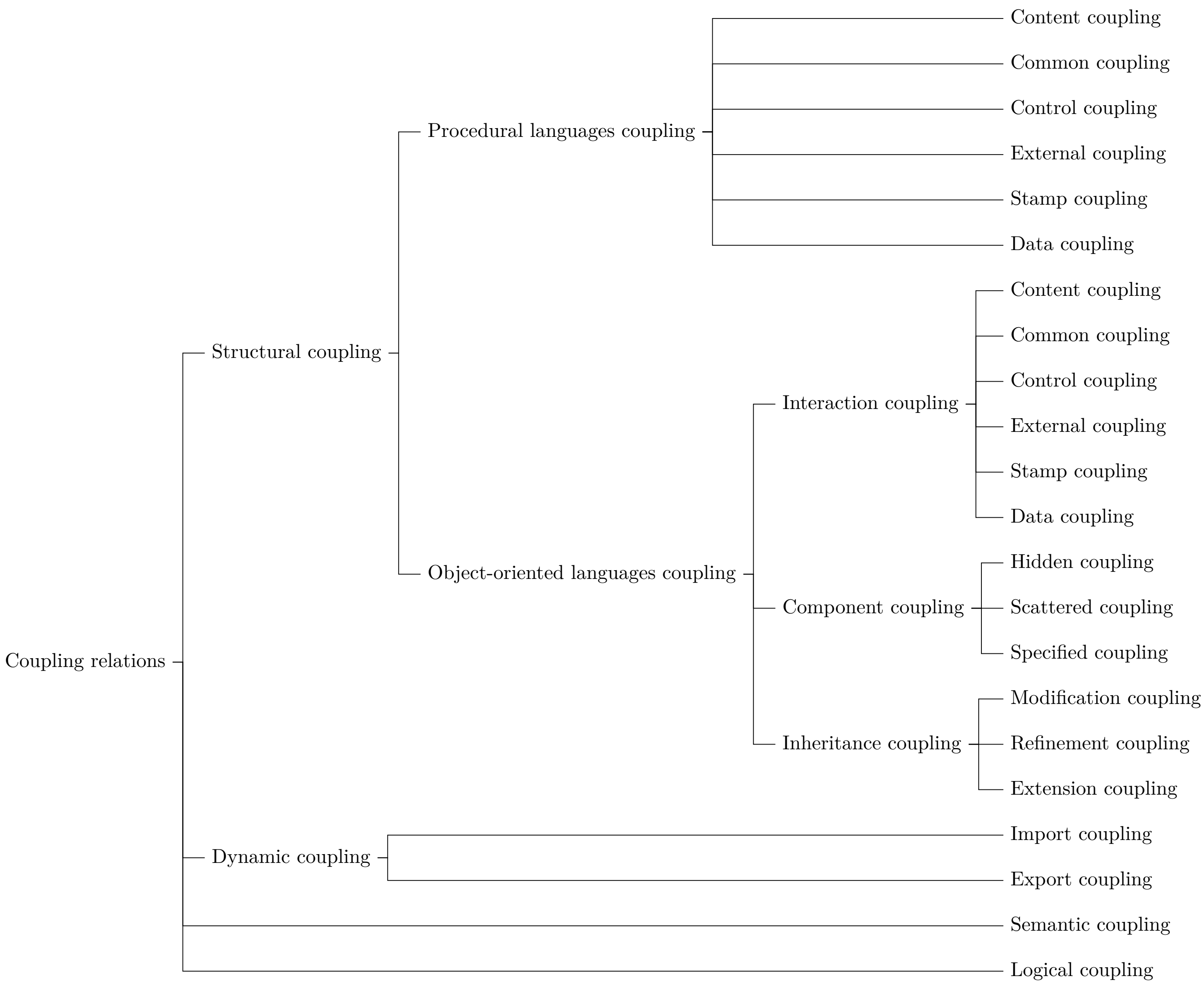

Figure 2: Coupling relation taxonomy

Myers divided the coupling for procedural programming languages in 6 different levels, reported in Table 3 ordered from the worst to the best in terms of consequences on the maintainability and quality of the software [19]. These coupling levels have been extended by Offutt et al. to include global and bidirectional coupling measures, previously left uncovered [5]. Although it has been originally introduced for procedural languages, this subdivision remains valid also for object-oriented ones.

In 1981, Henry and Kafura proposed an information-flow technique to construct different measures for a software system [21]. Their idea constitutes an interesting approach to compute coupling relations. In particular, the information flow metrics can recognize *content* coupling and *common* coupling. The authors argue that content coupling is equivalent to their *direct local flows*. Common coupling is equivalent to the *global flow* measure. Henry and Kafura developed two metrics *fan-in* and *fan-out*. *fan-in* counts the number of local flows to the considered procedure together with the number of data structures read by the procedure. *fan-out* measures the quantity of local flows from a selected procedure plus the number of data structures on which the procedure writes. At a later time, Allen et al. proposed to measure coupling metrics using the links and information obtainable building the system graph of a software [22, 23]. The strength of this approach is that it can be applied to many software design abstractions and to all kinds of programming languages (procedural and object-oriented).

The flourishing of the object-oriented paradigm brought the researchers to propose metrics and relations to cover its new aspects. Coupling relations for object-oriented systems have been investigated in the work done by Eder et al. in 1994. The authors identified three groups of coupling relations [24]: interaction, component and inheritance coupling. Their classification is shown in Table 4.

A fundamental structural coupling metric for object-oriented software is CBO (Coupling Between Object) [25], which belongs to the interaction coupling subgroup [24]. CBO reflects the degree to which an object acts upon another, excluding the parent-child relation. It constitutes one of the core cou-

Table 3: Structural coupling relations for procedural languages

Level	**Description**
Content coupling	Refers to two modules of which one directly accesses the contents of the other: e.g., module A modifies a statement or branches to a local label of module B.
Common coupling	Happens when two modules have access to the same global data: for example, both modules can read and write the same global record. Schach points out that if the access to the data is read-only, then this can not be considered common coupling [20].
Control coupling	Happens when a module explicitly controls the logic of another. However, this does not happen if the first module passes only data.
External coupling	Happens when two modules exchange information using an external element such as a file.
Stamp coupling	Exists between two modules if one of them passes a data structure as an argument to the second one, but the called module does not operate on all the components of the data structure.
Data coupling	Exists among two modules if the arguments that they pass to each other are all homogeneous data items: simple arguments or data structures in which all elements are used by the calling module [20].

Table 4: Structural coupling relations for object-oriented languages

Level	**Description**	**Applicability**	**Subdivision**
Interaction coupling	Defined as the invocations among different methods and their sharing of variables	Methods and Classes	Content coupling Common coupling External coupling Control coupling Stamp coupling Data coupling
Component coupling	A class A is related to another class B if and only if A is referenced in B: this happens when A is the domain of an instance or local variable, a method's parameter or a parameter of a method called inside a method of B	Classes	Hidden coupling Scattered coupling Specified coupling
Inheritance coupling	Relates two classes if one of them is a subclass of the other	Classes	Modification coupling Refinement coupling Extension coupling

pling metrics and it has been further refined and applied in different other domains. Chidamber and Kemerer also defined RFC (Response For a Class), a coupling measure related to CBO [26, 27] that measures the total communication potential. These two metrics were further analyzed by Briand et al. [28] and formalized to remove possible sources of ambiguities. Moreover, Briand et al. introduced CBO′ to include the ancestor classes in the metric computation, previously left excluded [25]. Other important metrics that they considered were Message Passing Coupling (MPC) and Data Abstraction Coupling (DAC), originally defined by Li and Henry [29]. Briand et al. further refined DAC in DAC′, a metric that counts the number of classes used as types of attributes. Finally, other important structural coupling measures are efferent and afferent coupling (C_e and C_a [30]), Coupling Factor (COF) [31, 32] and Information-flow-based coupling (ICP) [33]. Li defined two new coupling metrics to complement Chidamber and Kemerer's metrics suite: CTA (Coupling Through Abstract data type) and CTM (Coupling Through Message passing) [34]. Similarly to the DAC′ [28] metric, CTA measures the relation between two classes created when one of them uses the other in its data declaration. CTM (Coupling Through Message passing) measures the number of messages sent by a considered class to the other classes in the system, excluding the ones sent to objects used locally by the methods of the class. MPC, RFC and CBO were also modified to be applied to program slices [35], creating the new metrics SMPC, SRFC and SCBO. The method-level metrics proposed by Briand et al. have been adapted to a finer granularity by English et al. to distinguish the different types of constructs with which they might be related [36]. A specific focus has been given to the *friendship* relation.

To take into account indirect coupling relations and the strength of coupling between two classes, Li developed a new metric [37]. Indirect coupling has also been considered by Yang et al. [38, 39] (creating also a tool, *Indirect Coupling Detector*) and later by Almugrin et al. [40].

New measures have been introduced to allow an evaluation of the level of object-orientation of a program to estimate the possibility that an object-oriented fault happens. For this purpose, Tang et al. proposed new coupling metrics [41]: IC (Inheritance Coupling) and CBM (Coupling Between Methods). Gui and Scott focused instead on metrics specific for component reusability [42], defining measures for the direct coupling among two classes (CoupD), the transitive coupling among two classes (CoupT) and the total coupling of a software system (WTCoup).

An interesting approach is the one proposed by Aloysius and Arockiam, where a comprehensive coupling metric, CWCBO (Cognitive Weighted Coupling Between Objects), is defined to give an overall measure of different degrees of coupling [43]. This metric considers different kinds of coupling measures such as data coupling, control coupling, global coupling and interface coupling and applies to them a weighting factor. Using a comparative study, the authors supported their claim that CWCBO is a better indicator than CBO to measure the complexity of a class since it takes into consideration different coupling levels.

4.2. Dynamic coupling

Dynamic coupling rules were introduced to address problems left not completely answered by previous static coupling measures: e.g., dynamic binding and polymorphism [10]. In fact, these metrics lose precision when dealing with intensive use of inheritance and dynamic binding. Furthermore, they aim to evaluate software quality looking not only at the source code complexity, but also at its operational environment [44]. Further research confirmed that these metrics provide additional information to the structural metrics [45]. Dynamic coupling approaches can be further divided according to the coupling direction, import or export coupling, and their mapping level, object or class-level oriented [46]. Coupling direction captures the difference between a sending entity and a receiving entity [46, 10]. Considering the messages exchanged between entities, the distinction is:

- *Import coupling*: focus on the messages sent from an entity

- *Export coupling*: focus on the messages received by an entity

The mapping level reflects the applicability domain of dynamic coupling rules: *object-level* or *class-level* coupling. Moreover, Arisholm et al. proposed three different approaches to evaluate the strength of a coupling relation: the number of messages, the number of distinct method invocations and the number of distinct classes. The first one refers to the quantity of distinct messages exchanged between two entities. The other two represent the amount of methods called and classes used, respectively, by a method in an object. The classification of these metrics, as given by Arisholm [46], is summarized in Table 5.

Table 5: Dynamic Coupling Relations Summary [46]

Direction	Mapping	Strength	Name
Import Coupling	Object-level	Dynamic messages	IC_OD
		Distinct Methods	IC_OM
		Distinct Classes	IC_OC
	Class-level	Dynamic messages	IC_CD
		Distinct Methods	IC_CM
		Distinct Classes	IC_CC
Export Coupling	Object-level	Dynamic messages	EC_OD
		Distinct Methods	EC_OM
		Distinct Classes	EC_OC
	Class-level	Dynamic messages	EC_CD
		Distinct Methods	EC_CM
		Distinct Classes	EC_CC

The direction of coupling was taken into account also by Mitchell and Power: their idea was to expand the previously defined CBO metric [44]. The authors presented two new coupling relations, both divided to account for *internal* and *external* coupling. The first one, Run-time coupling between objects (R) (external or internal) counts the number of accesses made by/to a class divided by the total number of accesses. The second one, run-time import (or export) degree of coupling (RD), gives the strength of the coupling relation computed as the number of classes that access (or are accessed by) a selected class. In further research, Mitchell and Power, in their attempt to verify if CBO can be used efficaciously as a dynamic metric, defined two new coupling measures [47]: Run-time Coupling Between Objects (RCBO) counts the amount of classes that a specific class accesses at run-time. The second one, the Number of object class clusters (Noc), counts the number of clusters obtained considering a class in the system and studying the distribution of unique accesses per object.

Work at object-level was also done by Yacoub et al. [48]. The authors proposed two dynamic coupling metrics that operate on the export and import side, respectively. The former one, Export Object Coupling (EOC) measures the percentage of messages sent from an object to the other, compared with the total amount of messages exchanged in the complete execution. Import Object Coupling (IOC) works in the opposite way, reflecting the number of messages that an object receives from another. From EOC, Yacoub et al. defined OQFS (Object reQuest For Service) as the sum of the EOC among a selected object and all the other objects in the design. IOC was instead developed into OPFS (Object resPonse For Service), defined as the sum of IOC between the given object and all the other objects in the application during the execution of a specific scenario. These values affect the maintainability, understandability, reusability and the errors distribution in the code. Zaidman and Demeyer refined OQFS to work at class-level, introducing CQFS (Class reQuest For Service) [49]. CQFS counts all the methods that a given class calls during the application execution. Every method is counted once: calling the same method more than one time does not increase the count.

Hassoun et al. propose a general relation, called DCM (Dynamic Coupling Metric) to formalize the idea of dynamic coupling [50]. Their metric works at object-level and it can be used to analyze the coupling of a selected object or a system during the program execution.

Dynamic coupling metrics require analysis conducted at run-time, but the impact of the metrics is higher if they are computed at early stages of the development. To address this issue, pseudo-dynamic coupling metrics were proposed: static metrics that consider the expected usage profile (derived from UML graphs during the design phase) [51]. Referring to Chidamber and Kemerer's CBO, the pseudo-dynamic CBO metric was defined as the value of the static CBO multiplied by the value of the operational profile. It presents a strong correlation with the dynamic CBO metric. A similar static approach to compute dynamic metrics has been proposed by Liu and Milanova [52]. A different approach calculates dynamic metrics from the system use case maps and the interactions between different scenarios [53].

An interesting evolution of the metrics defined by Arisholm [46] has been introduced by Abualese et al. to evaluate the importance of a class in the understanding process that a developer has to face when approaching code written by a different person [54].

4.3. Semantic coupling

Classes can not only be structurally related to each other but also conceptually. Based on this idea, *semantic* coupling uses information from comments and identifiers to identify relations among software entities [11]. The technique proposed by Poshyvanyk and Marcus relies on Latent Semantic Indexing (LSI) [55]: a machine learning model developed to analyze relations between words and documents. To investigate coupling aspects left unaddressed by the previous metrics, the authors created four progressive coupling relations, each of them based on the previous one: CCM (Conceptual Coupling Between Methods), CCMC (Conceptual Coupling Between a Method and a Class), CCBC (Conceptual Coupling Between two Classes), also called CSBC (Conceptual Similarity Between two Classes) and CoCC (Conceptual Coupling of a Class) [11, 17]. Poshyvanyk and Marcus also considered the idea to exclude weak coupling links in the computation of the metrics defining a new metric called $\mathrm{CSMC_m}$. From it, they also recomputed CSBC and CoCC accordingly, producing the two new metrics $\mathrm{CSBC_m}$ and $\mathrm{CoCC_m}$. Újházi et al. have further improved this approach with a new metric called CCBO (Conceptual Coupling Between Object classes), which does not merely take the maximum but identifies a threshold to distinguish between a strong and weak semantic coupling [56].

All the aforementioned semantic coupling measures use LSI to create an initial semantic corpus for the analysis. Gethers and Poshyvanyk proposed a coupling approach based on a different technique: Relational Topic Model (RTM), a model that can find connections between documents based on the context [57]. The authors introduced a measure called Relational Topic-based Coupling (RTC). This metric individuates new aspects of coupling between classes compared to the metrics based on LSI, such as CoCC. Furthermore, a fundamental benefit of this model is that it does not need any previous knowledge about the links between classes.

Revelle et al. extended semantic coupling relations to work at feature level, aiming to identify which parts of a system are linked to a specific function [59, 58]. In fact, a feature represents the implementation of a functionality described in the requirements. Since features can be represented by structured information (source code and related artifacts) and unstructured information (textual information), two different metrics were proposed: SFC (Structural Feature Coupling) and TFC (Textual Feature Coupling). Furthermore, the authors introduced HFC (Hybrid Feature Coupling) to consider together the complementary information captured by SFC and TFC.

Semantic coupling has been combined with evolutionary coupling [60] or domain-based relations [61]. Domain-based coupling individuates relations among domain variables, functions and User Interface Components (UIC) [62, 63] and has been applied to fields such as code clone detection with promising results [64]. Gethers et al. defined CSE (Conceptual Similarity between Entities) and CSED (Conceptual Similarity between two UICs) to perform impact analysis in hybrid software systems [61]. Moreover, semantic and domain-based coupling relations have been checked to assure their orthogonality. This analysis confirmed that these relations capture different aspects of the analyzed system and, therefore, they can be efficaciously combined. Based on CBE, Kagdi et al. defined CSEMC and CSEBC [65]. Furthermore, semantic coupling has been combined with structural coupling to create a metric that takes into account both aspects at the same time [66]. The authors defined four coupling metrics, each of them based on the previous one (in a fashion similar to the one used by Poshyvanyk et al. [17]): MPC (Method Pair Coupling), HCMC (Hybrid Coupling between Method and a Class), HCCC (Hybrid Coupling between two classes) and SSCM (Coupling of a class in an object-oriented system). Moreover, they positively performed an evaluation to confirm that these metrics identify aspects not covered by structural and semantic coupling relations alone.

4.4. Logical coupling

Logical coupling (sometimes also called *evolutionary* or *change coupling*) works by finding similar change patterns in the release history: it aims to investigate "*the sequential dependencies such as if module A is changed in one release, module B is changed in the next release*" [67]. This approach has been further developed to be applied at class level in the research conducted by Gall et al. [68], with the aim to identify classes that share a common change behavior. The authors proposed a distinction between internal and external links: internal coupling happens between classes in the same module, while external coupling involves classes contained in different modules. Their approach works using data extracted from the CVS (Concurrent Versions System) release history. Further research focused on a finer-grained analysis of system evolution, compared to the description obtained using CVS. Robbes et al. [7] argued that this method is imprecise because it employs the commits as basic analysis blocks. For this reason, they defined coupling metrics to work using information collected during software development through a tool that saves all the changes made to a system in development together with the exact time at which they were made. Alali et al. proposed to further extend these metrics analyzing the contribution of age and pattern distance measures [69]. Age is defined as the period of time between the appearance of the specific evolutionary coupling relation and its disappearance. Pattern distance represents the tree distance between two files in a program. Another interesting approach is the one proposed by D'Ambros et al. [70]. They introduced two weighted change coupling measures EWSOC (Exponentially Weighted Sum of Coupling) and LWSOC (Linearly Weighted Sum of Coupling). They both emphasize recent changes over past ones, but the latter penalizes them less than the former.

4.5. Recent or isolated trends

Outside this classification, other coupling relations have been proposed. They are novel techniques, still too recent to be considered a proper subgroup of metrics, or relations developed for specific domains.

A first novel relation is *interaction coupling*. Interaction coupling aims to group artifacts that are likely to implement the same task. Zou et al. [71] worked on the task interaction history, defining the strength of the relation between two entities as the number of times they are accessed together. Although interaction coupling may be considered similar to logical coupling, the former requires information from the task interaction histories and involves not only artifacts that are changed together, but also entities that are viewed in the same portion of time. Interaction coupling and logical coupling have been combined by Bantelay et al. to predict future interactions [72].

The usefulness of general coupling relations led the researchers to tailor them to domain-specific applications: e.g., knowledge-based systems. Kramer and Kaindl proposed the Degree of Coupling of Frame (DCpF) metric to measure the number of slots in a frame connected to other slots in different frames by a rule [73]. Coupling measures have also been developed for Web Ontology Language to evaluate the complexity of the system [74]. Table 6 summarizes the metrics proposed for this application. Furthermore, coupling metrics have been modified to be applied to *Agent-oriented* software development. Jordan and Collier proposed a reformulation of the CBO metric as coupling between abstractions, defining Coupling Between Elements (CBE) [75]: two elements are coupled if any direct dependencies exist between any of their parts. If an element accesses or modifies the implementation details of another one, this leads to a dependency.

Table 6: Web Ontology coupling metrics

Metric	**Description**	**Ref**
NEC (Number of external classes)	Number of other classes outside the selected ontology	[74]
REC (References to External Classes)	Counts the number of references to external classes considering a selected ontology	[74]
RI (Referenced Includes)	Computes the number of includes used in an ontology	[74]
CBE-in (Coupling Between Entities)	Considers the class to be in the property domain	[76, 77]
CBE-out (Coupling Between Entities)	Considers the class to be in the property range	[76, 77]
SC (Self-Coupling)	Captures the properties with a class contemporary in the property range and domain	[76, 77]

Coupling relations have also been adapted for Aspect-Oriented (AO) software, where the basic entities are aspects and classes, to assess its reusability and maintainability [78]. Sant'Anna et al. [78] and Ceccato and Tonella [79] (further improved by Shen and Zhao [80] with the addition of other seven metrics) proposed an initial set of metrics to measure aspect-oriented coupling relations. An example is CBC (Coupling Between Components) [78], a general measure of coupling that accounts for different relations between classes and aspects in AO programs. However, these metrics have been criticized by the software engineering community for not taking into account finer dimensions of class-aspect coupling and their lack of empirical validation. Moreover, their adoption was disregarded by the software developers [81]. For this reason, Burrows et al. aimed to asses the quality of these metrics and, if necessary, how they might be improved. The authors defined a new AOP coupling measure called Base-Aspect Coupling (BAC) that quantifies the strength of the link between the base and the aspect code. In the same year, Bernardi and Lucca proposed a further set of coupling metrics based on their classification of aspects' interactions [82]. They proposed a metric for coupling due to interactions altering the static structure (CLSS), to interactions altering the control flow (CLCF) and to interactions altering the state of an object (CLSO). Finally, they considered all these interactions together in the metric CLA (Coupling Level of an Aspect). New metrics were proposed by Bennett and Mitropoulos to address the problem of aspect interference [83]: an aspect that causes unexpected changes to the flow of a class or a method. The authors argued that previously proposed AO coupling metrics did not cover all the interaction necessary to describe potential aspect interference. To solve this issue, two new metrics were introduced: IP (Interference Potential) and ICP (Interference Causality Potential). Furthermore, these two metrics have been combined in a new one, TIP (Total Interference Potential) [84]. At the same time, attempts have been made to create a comprehensive framework, independent from the language considered, to define AO coupling measures [85, 86, 87].

Moreover, specific coupling metrics have also been developed for Service-

Oriented Architectures (SOA) [88, 89, 90, 91, 92, 93, 94, 95]. A SOA is an architectural model to combine different services in one platform. It can be formed by a combination of technologies, products, APIs and various other components and is not related to a particular programming language [96]. Among the metrics proposed, we report ASSD (Average Service State Dependency), ASPD (Average Service Persistent Dependency), ARSD (Average Required Service Dependency) [88], SOCI (Service Operational Coupling Index), ISCI (Inter-Service Coupling Index) [89] and ASOU (Average Service Operation Coupling) [93]. ASSD and ASPD compute the average of the services' states and persistent state dependencies, respectively. A persistent state dependency happens between services that share a state, which all of them can use and update. Finally, ARSD measures the average number of services to which each service in the system provides its functionalities. SOCI (Service Operational Coupling Index) and ISCI (Inter-Service Coupling Index) measure the dependence of a service on other services and on messages, respectively. The former was adapted by the object-oriented metric RFC, the latter from the CBO metric. ISCI can be considered as the opposite of ARSD [88]. To measure the dependency based on messages, a new metric was formulated: SMCI (Service Message Coupling Index). Even if it may seem to have a dynamic nature, it is computed statically from the information model of the domain. Finally, ASOU computes the coupling of a service as the sum of its synchronous and asynchronous invocations divided by the total number of services in the domain. Karhikeyan and Geetha identified five types of dependencies that influence the coupling of a Service-Oriented system: direct, indirect, state, IO and delayed message dependency [94]. They developed a metric for each of them and proposed a fuzzy model to evaluate the overall coupling of a system.

The discussed coupling metrics for SOA are all static. Based on the promising results obtained by dynamic coupling in object-oriented systems, Quynh and Thang introduced a set of dynamic metrics for Service-Oriented systems [91]: CBS (Coupling Between Services), which has been derived from CBO, IMS (Instability Metric for Service), DC2S (Degree of Coupling between 2 Services) and

DCSS (Degree of Coupling within a given Set of Services).

Semantic coupling relations have also been further developed to deal with Service-Oriented Architecture (SOA). New metrics needed to be created since the ones proposed by Poshyvanyk et al. [17] could not be applied in this domain: comments and identifier names are not accessible for services and, furthermore, the required concepts can also be obtained using business level artifacts [97]. For these reasons, Kazemi et al. developed three coupling metrics [97]: CCO (Conceptual Coupling between Operations), CDSO (Conceptual Dependency of a Service to an Operation) and CCS (Conceptual Coupling of a Service).

An interesting application of coupling measures is to assess the information security of object-oriented designs [98]. To this purpose, a new metric CCC (Critical Classes Coupling) has been defined. CCC computes the degree of interconnection among classes and classified attributes in a given software design. Moreover, it is based on design graphs (e.g., UML). However, to extract security information these graphs need to be annotated using tools such as UMLsec [99] or SPARK's annotations [100].

Finally, coupling measures have been defined for Computational Science and Engineering (CSE) applications [101] and real-time application design [102]. In the context of real-time application design, Ahmed and Shoaib defined a set of metrics (e.g., MEF, Message Exchange Factor) to evaluate the system in its early development phases [102]. Kamble et al. investigated coupling in Computational Science and Engineering (CSE) software [101] to perform software integration. They claimed that this domain is different from others due to the complex algorithms and functions involved.

5. RQ2: Developed tools

Different tools have been proposed to extract the measures discussed in *Section 4*. Based on the previous classification, the aim is to identify how they work, the information that they require and their advantages and disadvantages. Table 7 shows a summary of the tools that we have considered divided based on the kind of metrics that they extract (structural, dynamic, semantic, logical coupling or relations belonging to smaller groups). For each tool its input and output are reported, together with its limitations: mainly its restriction to a particular programming language (or set of languages).

Some tools are stand-alone kits that simply extract a set of metrics: e.g., CCMETRICS [104]. However, during our analysis, two trends emerged clearly: the use of visualization techniques to improve users' understanding and the focus on extensibility. Moreover, in tools developed for dynamic coupling metrics, due to the significant amount of data that needs to be analyzed, researchers focused on scalability to improve the tools' performance.

5.1. Extensibility

A problem of metric tools proposed by the software engineering community is that the majority of them can not be extended to support new metrics or languages [132]. For this reason, some metrics tools focused on extensibility with the specific intent to support future metrics. Examples can be found in QScope [105], which provides an explicit mechanism to include new metrics and a framework to develop and test them, OOMeter [111] and AMT [106]. This last tool takes a further step towards extensibility, being expandable not only with new metrics but with new languages too. To achieve an implementation independent from the programming language analyzed it takes as input a representation of the source code using XML. However, this representation should be created from the source code using a parser. For this reason, at the time of the publication of their research, Kayarvizhy and Kanmani's tool AMT only supports Java and C#. A similar approach has also been implemented in WebMetrics [107]. The tool implements an architecture that includes an intermediate level of abstraction between the code and the metrics computation: the code is processed to extract a list of relations, which are analyzed in a second step to compute the desired metrics. This allows an easier implementation of new measures in the tool since the developer does not need to know how the

Table 7: Coupling Tools Analyzed

Coupling	Tool	Input	Output	Limitations	Ref
Structural coupling	ckjm	Java files or JAR files	CBO (Coupling Between Objects) RFC (Response For a Class) and C_a (Afferent coupling)	Restricted to Java applications	[103]
	CCMETRICS	Source code	DAC (Data Abstraction Coupling) MPC (Message Passing Coupling)	Restricted to object-oriented languages	[104]
	QScope	XML database of the program representation	CBO and RFC and their graphical visualization		[105]
	AMT	Source code	CBO, CBO′, RFC, MPC, DAC and DAC′	Restricted to Java and C# (At the time on which Kayarvizhy and Kanmani published)	[106]
	WebMetrics	Source code	CBO and RFC Fan-in and Fan-out	Supports C, C++, Java and Smalltalk	[107] [108] [109]
	DependencyViewer	Java files or JAR archives	C_a (afferent coupling) and C_e (efferent coupling)	Limited to Java applications	[110]
	OOMeter	Source code (Java or C#) or UML diagrams (in XMI)	CBO (Coupling Between Objects) (can be exported in XML, Microsoft Excel, html etc.)	Restricted to Java and C# Only supports UML in XMI format	[111]
	CLUSTERCHANGES	CodeFlow changeset	Clusters of diff-regions (visualized a tree graph)	Restricted to C#	[112]
	SCPA	UMLsec or SPARK graphs (generated with the tool)	CCC	Specific for assessing security (using UMLsec or SPARK's annotations)	[113]
	GMN tool	UML diagrams (in XMI format)	DAC, DAC′ Briand et al.'s metric suite [16] NASS and DCC		[114]
	AEA tool	Java source code	NOC, DIT	Restricted to Java	[115]
	JMCT	Java source files	CBO, RFC (refer to table in [116])	Restricted to Java	[116]
	JCAT	Java source files	PCC, ECC, GCC, ICC	Restricted to Java	[117]
	JCTIViz	Java source files	CBO, CTI	Restricted to Java	[118]
	Indirect Coupling Detector	Eclipse IDE	use-def indirect coupling	Eclipse plug-in	[38]
Dynamic coupling	JDissect	Running Java program	Dynamic coupling measures [10]	Restricted to Java	[10]
	SSS tool[6]	Running Java program	Total Dynamic Messages (TDM) Distinct Class Couples (DCC) Distinct Method Couples (DMC)	Restricted to Java applications	[119]
	DMA	Jar files	EC_CC, EC_CM and EC_CD IC_CC, IC_CM and IC_CD	Restricted to Java applications	[120]
	DynaMetrics	Running program	Set of static and dynamic measures [121]	Restricted to Java/C++	[121]
Semantic coupling	IRC^2M	Source code	CoCC and $CoCC_m$ CSBC and $CSBC_m$	Restricted to C++ programs	[11]
	$FLAT^3$	Source code Feature-Method Mapping	TFC SFC	Eclipse plug-in	[58]
Logical coupling	ROSE	CVS data	Locations for further changes Warnings about probable missing changes		[122]
	Evolution Radar	CVS data	Graphical visualization of coupling between modules and files		[123] [124] [125]
	YMNC tool	CVS data	List of files changed together with the selected one		[126]
	Hanakawa's tool	Java source code and CVS data	Module coupling measure Logical coupling measure (graphic visualization of them)	Module coupling is restricted to Java	[127]
Other couplings	OWL-VisMod	Ontologies code	Graphical visualization of CBE-in and CBE-out relations	Restricted to OWL language	[128]
	AJATO	Source code and Concern map (XML) and Rules (XML)	CBC (Coupling Between Components) Design Warnings	Restricted to Aspect-Oriented applications	[129] [130]
	AJMetrics	AspectJ files and Java files	CAM, CAA, CAI and CAE, CIM, CFA and a suite of structural coupling metrics	Restricted to Aspect-Oriented software	[80] [131]
	CT tool	AspectJ source code	CAE, CIM, CMC, CFA	Restricted to AspectJ	[79]
	SSP tool	UML diagrams	SOCI, ISCI and SMCI	Restricted to SOA	[89]

parser operates, but only the generated intermediate relations.

5.2. Visualization techniques

Applying visualization techniques to metric tools constitutes another important trend in research. The goal is not only to extract a set of software metrics but to support and improve the users' understanding. A step in this direction has been made by DependencyViewer [110] and OOMeter [111] in 2005. Both of them can show the metrics extracted using simple graphs: e.g., DependencyViewer reports the metrics computed for a package as a column graph.

In the field of logical coupling metrics, Evolution Radar [123, 124, 125] (2006) and the tool proposed by Hanakawa [127] (2007) implemented a visualization technique. D'Ambros et al. [124] argue that visualization techniques give immediateness to the user. *Evolution Radar* shows as output the coupling links existing between a selected module and the other system' modules. The visualization interface uses the distance between the center (where the selected module is located) as a measure of the strength of the coupling relation: the closer a module is to the center, the stronger is the link. Furthermore, due to the interactivity of this approach, it is possible to see more information related to the selected entity such as the author, timestamp, comments, lines added and removed and its source code and the logical coupling among entities over time. Hanakawa's visualization tool presents two maps: a module coupling map and a logical coupling one. Both of them can be shown at the same time. JCTIViz [118] (2008) uses a polymetric view to display software metrics. Each class or interface is represented with a node: the dimensions of the node (height, width and depth) represents the value of a metric. In particular, the node depth represents the CBO value. A different approach considers the creation of tools as plug-ins for existing IDEs. EPOSpix [133] exploits this idea showing related classes in Eclipse with a pixel map. eROSE [134] is an Eclipse plug-in that

[6] When the tool's name is not explicitly stated in the referenced research, we will call it with the initial letters of the authors' surnames

computes logical coupling to suggest related changes to the developer.

Visualization techniques have been applied also by Garcia et al. to coupling relations among OWL ontologies [76, 128]. Their tool, OWL-VisMod, requires as input the ontology's code and it shows the coupling CBE-in/out relations among the classes. Classes are displayed using a radial layout, where the selected class occupies the center. On the right and left side are displayed the classes coupled by a CBE-out or a CBE-in relation, respectively. An edge link couples classes and its color indicates the direction of the coupling relation.

5.3. Scalability and Dynamic coupling

Different ways exist to collect dynamic metrics: using run-time information or relying on simulating the execution behavior of a system using interaction diagrams, such as UML or Real-time Object Oriented Modeling (ROOM) language [135].

A first tool to find dynamic relations, proposed by Arisholm, is JDissect [46]. The tool works in two phases: in the first one, it gathers information from a running program, while in the second step the collected data are analyzed. However, its first limitation consists in its restriction to Java applications, due to its connection with the JVM (Java Virtual Machine). In fact, this tool uses the JVM interfaces to collect dynamic information. For what concerns the input required, JDissect needs a running Java program to extract the dynamic coupling relations in it. Another tool to extract dynamic coupling metrics is DynaMetrics, proposed by Singh and Singh [121]. It can compute both dynamic and static metrics, analyzing the data collected at run-time (specifically, event log files).

Extracting a significant amount of data from the execution of a program may require a vast amount of time and resources. To mitigate this problem, in 2015 Sarvari et al. proposed to parallelize this process using Hadoop MapReduce [119]. Hadoop MapReduce needs the XML file of the program to be executed. For this reason, the authors utilized JP2 [136]: an open source tool that creates CTT XML files from a running Java program. Furthermore, Hadoop

MapReduce can be used both locally and on the cloud: a cloud-based approach further helps in dealing with large quantities of data. For this reason, in 2017 Dogra et al. [120] proposed DMA (Dynamic Metric Analysis), a tool based on Platform as a Service (PaaS). Like Sarvari et al.'s tool, DMA relies on JP2 but adapts it to be streamed to the cloud. In this way, it allows the user to have a real-time analysis of the coupling metrics during the program execution. Sarvari et al.'s tool returns three dynamic coupling measures: TDM (Total Dynamic Messages), DCC (Distinct Class Coupling) and DMC (Distinct Method Couples). The authors introduced this nomenclature for the first time in the software engineering research field. However, these metrics are the same as the ones developed earlier by Arisholm et al. [10]: TDM corresponds to IC_CD (or EC_CD, depending on the considered direction of the relation), DCC is equal to IC_CC (or EC_CC) and DMC is the same as IC_CM (or EC_CM).

Another approach is to collect dynamic coupling data from the UML diagrams of the program [46]. On the one hand, since these diagrams are usually done in the early design phase, the main advantage of this approach resides in the possibility of using dynamic relations to take design decisions. On the other hand, the coupling measures collected tend to be underestimated due to the impossibility to distinguish the different messages in the set of possible messages in the system using UML. Unfortunately, our systematic review did not find any examples of tools that implemented this approach to extract dynamic metrics. Tools such as OOMeter [111] and the tool proposed by Girgis et al. [114] extract coupling metrics from design diagrams, but they are restricted to structural metrics.

6. Discussion

A vast quantity of coupling metrics and relations has been proposed for different paradigms and applications, starting with procedural languages and, later, object-oriented ones. Due to their importance in assessing the software

[7] at different granularity levels

Table 8: Coupling Metrics Summary

Coupling group	Metric	Tool(s)	Metric Ref	Tool ref
Structural coupling	fan-in, fan-out	WebMetrics	[21]	[107] [109]
	CBO	ckjm, QScope, AMT, WebMetrics, OOMeter, DynaMetrics	[8] [25]	[103], [105], [106], [107] [109], [111], [121]
	RFC	ckjm, QScope, AMT, WebMetrics	[8] [25]	[103], [105], [106], [107] [109]
	CBO′	AMT	[28]	[106]
	MPC, DAC	CCMETRICS, AMT	[29] [28]	[104], [106]
	DAC′	AMT	[28]	[106]
	SMPC, SRFC, SCBO	no tool	[35]	
	C_a	ckjm, DependencyViewer, DynaMetrics	[30]	[103], [110], [121]
	C_e	DepencencyViewer, DynaMetrics	[30]	[110], [121]
	COF	no tool	[32] [31]	
	ICP	no tool	[33]	
	NAS	no tool	[137]	
	CTA, CTM	no tool	[34]	
	IC, CBM	no tool	[41]	
	CWCBO	no tool	[43]	
	CCC	SCPA	[98]	[113]
	RMC, RIC	no tool	[138]	
	MPEC, MPIC, AFM	no tool	[139]	
	PLC	no tool	[140]	
	CIC, CNIC, MC, CC, AMC	no tool	[141]	
	CoupD, CoupT, WTCoup	no tool	[42]	
Dynamic coupling	IC and EC[7]	DMA, JDissect	[46] [10] [142]	[120], [10]
	R, RD	no tool	[44]	
	RCBO	DynaMetrics	[47]	[121]
	Noc	no tool	[47]	
	EOC, IOC, OQFS, OPFS	DynaMetrics	[48]	[121]
	CQFS	DynaMetrics	[49]	[121]
	DCM	no tool	[50] [143]	
	TDM, DCC, DMC	SSS tool	[119]	[119]
	EUC, EIUC	no tool	[54]	
	ICV	no tool	[53]	
Semantic coupling	CCM, CCMC, CSMC	no tool	[11] [17]	
	CSBC (CCBC), CoCC	IRC^2M	[11] [17]	[11]
	$CSBC_m$ ($CCBC_m$), $CoCC_m$	IRC^2M	[11] [17]	[11]
	CCBO	no tool	[56]	
	RTC	no tool	[57]	
	SFC, TFC	$FLAT^3$	[58]	[58]
	SFC′, HFC	no tool	[58]	[58]
	CSE, CSED	no tool	[61]	
	CSEMC, CSEBC	no tool	[65]	
Logical coupling	LC	ROSE, Evolution Radar, YMNC tool and Hanakawa's tool	[67] [67] [7]	[122], [123] [124], [126], [127]
	CC, TC	no tool	[7]	
	IC	no tool	[7] [71]	
	NOCC, SOC, EWSOC, LWSOC	no tool	[70]	
Semantic + structural coupling	MPC, HCMC, HCCC, SSCM	no tool	[66]	

Table 9: Other Coupling Metrics Summary

Applicability field	Metric	Tool(s)	Metric Ref	Tool ref
Knowledge-based Systems	DCpF	no tool	[73]	
Web Ontology Language	NEC, REC, RI	no tool	[74]	
	CBE-in, CBE-out	OWL-VisMod	[76]	[128]
	SC, iSC, iCBE-in, iCBE-out	no tool	[76]	
aspect-oriented Software	CBC	AJATO	[78]	[129] [130]
	CFA, CIM, CMC, CAE	AJMetrics, CT tool	[79]	[80] [79]
	CAM, CAA, CAI, RFM, RFP	AJMetrics	[80]	[80]
	BAC	no tool	[81]	
	IP, ICP	no tool	[83] [84]	
	TIP	no tool	[84]	
	CoAT, CoPT, CoAR, CoOI, CoI, CoHA	no tool	[87]	
	CLSS, CLCF, CLSO, CLA	no tool	[82]	
Agent Orientation Paradigm	CBE	no tool	[75]	
Service-Oriented Architecture	ASSD, ASPD, ARSD	no tool	[88]	
	SOCI, ISCI, SMCI	SSP tool	[89]	[89]
	WISCE, WESICE, WESOCE, ESICSI, EESIOC, SIIEC, SPARF, SPURF	no tool	[95]	
	ASOU	no tool	[93]	
	CBS, IMS, DC2S, DCSS	no tool	[91]	
	CCO, CDSO, CCS	no tool	[97]	
	DD, IDDT, IDDSD, IDD, SD, IOD, DMsgD	no tool	[94]	
Remote-component-based Systems	CCOF	no tool	[144]	
Real-time application design	MEF	no tool	[102]	

quality and analyzing programs' features, coupling relations were proposed to investigate aspects left uncovered by previous research and to be applied to specific application domains.

Our research showed how CBO (Coupling Between Objects), proposed by Chidamber and Kemerer as part of their metrics suite [25], became a fundamental coupling metric used as base for further metrics and refinements by other researchers: examples can be found in CBO′ [28], CWCBO [43] and CBE (Coupling Between Elements) [75]. Moreover, our investigation revealed that sometimes the researchers encountered difficulties in retrieving previously proposed metrics. Analyzing the material collected in our review, we noticed inconsistencies in the metrics names: e.g., afferent and efferent coupling have been proposed as C_a and C_e by Martin [30], but later referred by Singh and Singh [121] as AFC and EC. This is only a formal issue, but *different nomenclatures for referring to the same metric may undermine the coherence of the research corpus in this field.* The problem of formally defining the metrics and validate them led to the creation of many frameworks: e.g., the one defined by Tempero and Ralph [145].

Software metrics should undergo a theoretical and empirical validation when they are introduced. Our investigation revealed that coupling metrics are evaluated referring to the properties defined by Kitchenham et al. [146], Weyuker [147] and Briand et al. [28]. Metrics such as CTA and CTM [34], CWCBO [43] and CCBC [11] have been validated using this process. However, we noticed that a vast number of metrics have been proposed without undergoing a theoretical evaluation: for instance, based on properties like Representation condition [146]. Many studies performed only an empirical evaluation. Using a set of test cases, the goal of the studies was to assess that the newly proposed metric achieves better performance than a previous one as an indicator for a specific application: e.g., fault prediction. Moreover, a common trend is to apply correlation analysis techniques (Spearman correlation or Principal Component Analysis) to verify the orthogonality of a new metric compared to previously presented ones. More emphasis has been given by the researchers on this second aspect of the evaluation. The theoretical evaluation does not seem to be considered as fundamental as the empirical one since the latter contributes to highlight the novelty of the metric. Thus, we suggest *novel metrics to employ both a theoretical and empirical validation/correlation analysis.*

Table 8 shows a summary of the metrics retrieved in our systematic literature review. They are grouped based on the category of coupling relations to which they belong. Furthermore, they are associated with the tools that can be used to extract them (if any). While for the structural, dynamic and semantic coupling relations a set of metrics has been defined, for the logical coupling relations no strict metric definitions seem to exist. In the table, we referred to the classification given by Robbes et al. [7], but their definitions allow different interpretations of the same metric. Further efforts should be devoted to provide a *consistent formal definition of logical coupling metrics.* Table 9 contains an overview of the metrics belonging to the *Other coupling approaches* group. They are grouped based on the field of applicability (e.g., Aspect-Oriented software). As in the previous table, the tools that can be used to extract them (if retrieved in our systematic review) are reported.

Our analysis of the coupling metric tools proposed by the researchers revealed two interesting trends: the progressive use of visualization techniques as a means to show the information to the user and the focus on making easily extendible tools. Visualization techniques, used in tools such as Evolution Radar [124] or OWL-VisMod [128], help the user to have a better understanding of the considered software properties. Usually, this approach allows changing the considered entity interactively. D'Ambros et al. stated that the idea of recurring to visualization is based on the following motives: "*it provides effective ways to break down the complexity of information*" and "*it has been shown to be a successful means to study the evolution of software systems*" [124]. As the second trend, easily expandable tools want to overcome the problem of having metric tools that work only on a specific programming language (or groups of languages). Researchers proposed modular designs in which new metrics can be implemented without the need to understand the whole tool implementation. Examples can be found in tools such as AMT [106] and WebMetrics [107].

Tahir and MacDonell stated that dynamic metrics could be collected using a run-time analysis or executable modules and interaction diagrams (UML or ROOM) [135]. Although both of these approaches have been analyzed in the literature, in our review we did not find any tool that implemented a methodology based on interaction graphs. This could be caused by the lack of precision that dynamic coupling metrics computed during the design phase are likely to have, which may have discouraged further research attempts in this direction. However, it is also necessary to highlight that this may be caused by the limited scope of our review, as given by our procedure and especially the choice to restrict the analysis to academic-developed tools. Also the semantic coupling area suffers from a lack of tools to extract its correlated metrics: IRC^2M [17] and $FLAT^3$ [58] are the only ones retrieved in our systematic review. This can be explained by the fact that *semantic coupling relations have been investigated only by a restricted group of researchers.*

Coupling relations can be used to cluster related code changes, helping developers in the process of reviewing and modifying their software. Logical coupling is particularly suited for this task, due to its intrinsic nature: logical coupling relations were introduced to find similar change patterns in the code release history [67]. An example can be found in ROSE [122], which gives suggestions to the user regarding which portions of code are likely to have to be changed with the current one. However, also structural or semantic coupling relations can be effectively used with this intent. CLUSTERCHANGES [112] uses data coupling to cluster code diff-regions that influence each other and, therefore, should be inspected together when modifying one of them. On the contrary, we argue that dynamic coupling metrics are unsuitable for this task since they reflect run-time relations among software elements which can not be easily collected when dealing with code changes. An interesting way to approach the problem of grouping related code changes is given by the evolutionary coupling relations proposed by Zou et al. [71]. Information on which entities have been accessed together during the development phase may constitute a sound basis on which grouping together portions of code: in fact, these are likely to implement the same functionality.

7. Coupling Relations: A Research Roadmap

While the research community heavily investigated ways to measure coupling relations, we believe that future research directions should and will be devoted to the *application* of such coupling metrics as well as the definition of effective *combinations* of metrics that would allow a better estimation of the actual coupling of software classes. This section aims at reporting a (non-exhaustive) roadmap for further research in the field.

Applications. There are plenty of opportunities to use coupling metrics to support other software maintenance and evolution tasks. For instance, their use in the context of code review may represent an effective method to improve the way developers detect defective source code. Specifically, change-based code review constitutes an important trend in software development and improving the existing techniques may lead to a significant contribution to software engineering [148, 149, 150]. Coupling relations may be applied to analyze the code contained in different changes and, consequently, cluster similar changes together. Baum et al. proposed an ordering theory for code changes based on the relations that they share with each other [148]. In particular, they conducted a survey among developers to evaluate which relations were considered important. Among all of them, they mentioned the *similarity* relation. We argue that logical and semantic coupling relations may be applied as practical implementation of this relation. Still in the context of code review, coupling metrics might be exploited in conjunction with just-in-time defect prediction [151]: we envision the introduction of coupling-related information on top of the recommendations provided by defect prediction models, so that developers might be informed on the classes having relations with a defective file and possibly assess the risk of defect propagation over these classes.

Another promising research field in Software Engineering is Code Smell detection [152, 153, 154]. Recent works started to exploit it by using machine learning techniques [155, 156] and to classify the severity of a code smell issue [157]. While some structural and logical coupling metrics have already been used as features of these models, there is still room for improvement: as shown by our survey, the role of many complementary coupling metrics can be explored to improve the (not always good [158]) performance of currently available code smell prediction models. Still in the same area, the application of conceptual coupling metrics have been explored by Palomba et al. [159, 160]. The authors also suggested that the exploration of a combination between structural and conceptual metrics may lead to promising results. This is something that is still unknown and that might lead to new research directions on how to combine the output of different metrics. At the same time, it remains unclear what is the value of other coupling metrics in the context of code smell detection: for example, to the best of our knowledge, implications of using dynamic coupling metrics to detect code smells are still to be evaluated. This seems to be a natural fit for the identification of Message Chain instances [161], given its intrinsic dynamic nature: in fact, it occurs when a long chain of method invocations is required for the operations of a class [162].

Finally, coupling relations have found a major field of application in Change Prediction, a research area dealing with identifying the classes that are more prone to be modified in the future [163]. Most works rely on the use of structural coupling metrics (among others) as indicators of these classes [164]. A recent study conducted by Elish and Al-Rahman Al-Khiaty evaluates a set of evolution metrics for change prediction purposes [165]. The authors reported that these metrics measure different dimensions than the classical Chidamber and Kemerer's metrics suite [25] and that their combination improved the accuracy of their prediction model. Based on the promising results of their work, we argue that the application of logical coupling or conceptual coupling metrics to this context may be worthy. This metric may be combined with structural or dynamic ones and tested to see if the performance of a model that takes into account these different aspects increases: we expect so from the moment that recent studies [163, 166] showed how an improved description of the change prediction phenomenon, done through the addition of orthogonal information, can dramatically increase the overall ability of prediction models in discriminating the classes that are more likely to change in the future.

Coupling relations and metrics have been applied in many different contexts, of which the ones cited above (e.g., code review, code smells detection or change prediction) constitute just a small part. Depending on the application considered, combining two or more groups of coupling metrics may be worthwhile: the existing techniques could increase their performance. An example may be found in the research conducted by Palomba et al. to identify code smells with conceptual coupling metrics, where the authors argue that the possible combination of these metrics with others belonging to different groups (e.g., structural or conceptual) may lead to a further performance increase [159].

Combination. During our investigation, we noticed very few attempts to integrate previously proposed coupling metrics in an *ensemble* metric, i.e., a method able to combine the information coming from different sources. In our opinion, this represents an important research direction that might be worth to investigate to come up with more powerful solutions for measuring coupling relations. As an example, consider the application of machine learning approaches in this context: coupling metrics computed using different data sources (e.g., structural vs conceptual coupling) might be nicely adopted as features of a regressor able to estimate a combined form of coupling that may provide developers with a comprehensive view of the phenomenon, thus facilitating her ability to take informed decisions. At the same time, we envision a combination of those metrics through the use of search-based algorithms: a clear opportunity is represented by the possibility to apply them for refactoring purposes (e.g., to improve software re-modularisation by means of aggregate measures that optimize the locations of classes).

Furthermore, our work classifies new approaches or coupling metrics for specific domains in a generic group called *"recent or isolated trends"* (section **4.5**). The knowledge on those metrics is still poor and the way they can effectively complement existing measures is still unknown. This represents an opportunity for future research, as researchers are called to investigate further how these emerging metrics can be combined with the set of metrics for which way more information is available.

8. Conclusion

This work presented a systematic review of the coupling relations and metrics proposed until now by the software engineering research community. In the first part of our research, we analyzed the trends that emerged over time in the software coupling area. We developed a taxonomy, as complete as possible within the limitations of our approach, of these relations in the attempt to give a systematic classification of over thirty years of research in the field. Based on previous works, such as the one done by Bavota et al. [9], we divided the coupling relations into four main groups: structural, dynamic, semantic and logical. Furthermore, we included a fifth group of coupling metrics not listed with the previous ones, since they constitute new trends still in development or coupling metrics developed for a particular field of applicability (such as knowledge-based systems or aspect-oriented applications).

In the second part of our investigation, we presented the tools developed by the research community to extract (and sometimes even visualize) coupling relations. The tools retrieved have been summarized in Table 7 maintaining the structure used to answer our first research question: dividing the tools based on the coupling group of metrics that they extract. For each tool, we highlighted the input that it needs and the output that it produces, together with its possible limitations: e.g., a restriction to a particular programming language. Moreover, we analyzed three main trends noticed in the academic-proposed tools: application of visualization techniques, extensibility, and scalability (applied to dynamic coupling metrics). We proposed a discussion on our findings and a roadmap for future work. The complexity of this research field sometimes led to discrepancies among the introduced coupling metrics. As guidance for future work we highlighted interesting applications of the presented coupling relations and metrics (change clustering, code review, code smells detection and change prediction), reporting the groups of coupling metrics already applied in these fields together with the ones that are yet to be explored and that may constitute the starting point for future work. Furthermore, we discussed the possibility to combine existing coupling metrics to create *ensemble* metrics, able to combine information from different sources.

List of primary studies

- A. Offutt, M. Harrold, P. Kolte, A software metric system for module coupling, Journal of Systems and Software 20 (1993) 295–308.

- J. B. N. Fenton, Software Metrics: A rigorous and practical approach, 2014.

- R. Robbes, D. Pollet, M. Lanza, Logical Coupling Based on Fine-Grained Change Information, 15th Working Conference on Reverse Engineering, WCRE

’08 (2008) 42–46.

- S. Chidamber, C. Kemerer, Towards a metrics suite for object oriented design, Conference Proceedings on Object-oriented programming systems, languages, and applications (OOPSLA ’91) (1991) 197–211.

- G. Bavota, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, A. De Lucia, An Empirical Study Of The Developers’ Perception Of Software Coupling, Proceedings of the 2013 International Conference on Software Engineering (ICSE ’13) (2013) 692701.

- E. Arisholm, L. Briand, A. Foyen, Dynamic Coupling Measurement For Object –oriented Software, IEEE Transactions on Software Engineering 30 (8) (2004) 491–506.

- D. Poshyvanyk, A. Marcus, The Conceptual Coupling Metrics for Object –Oriented Systems, ICSM ’06 Proceedings of the 22nd IEEE International Conference on Software Maintenance (2006) 469–478.

- S. Kirbas, T. Hall, A. Sen, Evolutionary coupling measurement: Making sense of the current chaos, Science of Computer Programming 135 (2017) 4–19.

- A. Nicolaescu, H. Lichter, Y. Xu, Evolution of Object Oriented Coupling Metrics: A sampling of 25 years of research, IEEE/ACM 2nd International Workshop on Software Architecture and Metrics (SAM) (2015) 48–54.

- L. Briand, P. Devanbu, W. Melo, An Investigation into Coupling Measures for C++, Proceeding of the 1997 International Conference on Software Engineering (1997) 412–421.

- D. Poshyvanyk, A. Marcus, R. Ferenc, T. Gyimóthy, Using information retrieval based coupling measures for impact analysis, Empirical Software Engineering 14 (1) (2009) 5–32.

- J. Alghamdi, Measuring Software Coupling, in: Proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems, SEPADS’07, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2007.

- G. Myers, Reliable software through composite design, Mason and Lipscomb, 1975.

- S. Schach, Object-Oriented and Classical Software Engineering, 8th edition, McGraw-Hill, 2011.

- S. Henry, D. Kafura, Software Structure Metrics Based on Information Flow, IEEE Transactions on Software Engineering SE–7 (5) (1981) 510–518.

- E. Allen, T. Khoshgoftaar, Y. Chen, Measuring coupling and cohesion of software modules: an information-theory approach, Proceedings of the 7th International Software Metrics Symposium, 2011. METRICS 2001 (2001) 124–134.

- E. B. Allen, S. Gottipati, R. Govindarajan, Measuring size, complexity and coupling of hypergraph abstractions of software: An information theory approach, Software Quality Journal 15 (2) (2007) 179–212

- J. Eder, G. Kappel, M. Schrefl, Coupling and Cohesion in Object –Oriented Systems, 1992.

- S. Chidamber, C. Kemerer, A metrics suite for object oriented design, IEEE Transactions on Software Engineering 20 (6) (1994) 476–493.

- M. Hitz, B. Montazeri, Measuring Coupling and Cohesion In Object –Oriented Systems, Proceedings of International Symposium on Applied Corporate Computing 35 (1995) 25–27.

- L. Briand, S. Morasca, V. R. Basili, Property-based software engineering measurement, IEEE Transactions on Software Engineering 22 (1) (1996) 68–86.

- L. Briand, J. Daly, J. Wust, A unified framework for coupling measurement in object-oriented systems, IEEE Transactions on Software Engineering 25 (1999) 91–121.

- W. Li, S. Henry, Object-Oriented Metrics that Predict Maintainability, Journal of Systems and Software 23 (1993) 111–122.

- R. Martin, Object oriented design quality metrics: an analysis of dependencies, vol. 2, 2006.

- F. Abreu, R. Esteves, M. Goullão, Toward the Design Quality Evaluation of Object–Oriented Software, 1995.

- F. Abreu, R. Carapuça, Object-Oriented Software Engineering: Measuring and Controlling the Development Process, in: Proceedings of the 4th International Conference on Software Quality, 1994.

- Y. Lee, B. Liang, S. Wu, F. Wang, Measuring the Coupling and Cohesion of an Object-Oriented Program Based On Information Flow, in: Proceedings of the International Conference on Software Quality, 1995.

- W. Li, Another Metric Suite for Object-oriented Programming, Journal of System and Software 44 (2) (1998) 155–162.

- J. Rilling, W. J. Meng, O. Ormandjieva, Context driven slicing based coupling measures, in: 20th IEEE International Conference on Software Maintenance,2004. Proceedings., 2004.

- M. English, J. Buckley, T. Cahill, Fine-Grained Software Metrics in Practice, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), ISSN 1949–3770, 295–304, doi:10.1109/ESEM.2007.32, 2007.

- H. Li, A Novel Coupling Metric for Object-Oriented Software Systems, in: 2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop, 609–612, doi:10.1109/KAMW.2008.4810562, 2008.

- H. Y. Yang, E. Tempero, R. Berrigan, Detecting indirect coupling, in: 2005 Australian Software Engineering Conference, ISSN 1530–0803, 212–221, doi:10.1109/ASWEC.2005.22, 2005.

- H. Y. Yang, E. Tempero, Indirect Coupling As a Criteria for Modularity, in: First International Workshop on Assessment of Contemporary Modularization Techniques (ACoM '07), 10–10, doi:10.1109/ACOM.2007.5, 2007.

- S. Almugrin, W. Albattah, A. Melton, Using indirect coupling metrics to predict package maintainability and testability, Journal of Systems and Software 121 (2016) 298–310, ISSN 0164–1212, doi: https://doi.org/10.1060 1016/j.jss.2016.02.024, URL http://www.sciencedirect.com/science/article/pii/S016412121600056X.

- M. Tang, M. Kao, M. Chen, An empirical study on object-oriented metrics, in: Proceedings Sixth International Software Metrics Symposium (Cat. No.PR00403), 242–249, 1999.

- G. Gui, P. D. Scott, New Coupling and Cohesion Metrics for Evaluation of Software Component Reusability, in: 2008 The 9th International Conference for Young Computer Scientists, 1181–1186, doi:10.1109/ICYCS.2008.270, 2008.

- A. Aloysius, L. Arockiam, Coupling Complexity Metric: A Cognitive Approach, International Journal of Information Technology and Computer Science 4 (2012) 29–35.

- A. Mitchell, J. Power, An empirical investigation into the dimensions of run-time coupling in Java programs, Proceedings of the 3rd international symposium on Principles and practice of programming in Java (2004) 9–14.

- R. Geetika, P. Singh, Dynamic Coupling Metrics for Object Oriented Software Systems: A Survey, SIGSOFT Softw. Eng. Notes 39 (2) (2014) 1–8, ISSN 0163–5948, doi:10.1145/2579281.2579296, URL http://doi.acm.org/10.1145/2579281.2579296.

- E. Arisholm, Dynamic coupling measures for object-oriented software, Proceeding of the 8th IEEE Symposium on Software Metrics (2002) 33–42.

-V. Dixit, R. Vishwkarma, Comparison of class-level versus object–level static and dynamic coupling and cohesion measures in object oriented programming, 11th International Conference on Wireless and Optical Com1munication Networks (WOCN) (2014) 1–5.

-H. Abualese, P. Sumari, T. Al-Rousan, M. R. Al-Mousa, Utility classes detection metrics for execution trace analysis, in: 2017 8th International Conference on Information Technology (ICIT), 469–474, doi:10.1109/ICITECH.2017.8080044, 2017.

- A. Mitchell, J. Power, Using object-level run-time metrics to study coupling between objects, Proceedings of the 2005 ACM symposium on Applied computing (SAC ’05) (2005) 1456–1452.

- S. Yacoub, H. Ammar, T. Robinson, Dynamic metrics for object oriented design, Proceedings of the 6th Software Metrics Symposium (1999) 50–61.

-A. Zaidman, S. Demeyer, Analyzing large event traces with the help of coupling metrics, in: Proceedings of the Fourth International Workshop on OO Reengineering, 2004.

- Y. Hassoun, R. Johnson, S. Counsell, A dynamic runtime coupling for meta-level architectures, Proceedings of the 8th European Conference on Software Maintenance and Reengineering, CSMR 2004 (2004) 339–346.

- R. Gunnalan, M. Shereshevsky, H. H. Ammar, Pseudo dynamic metrics[software metrics], in: The 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005., ISSN 2161-5322, 117–, doi:10.1109/AICCSA.2005.1387106, 2005.

- Y. Liu, A. Milanova, Static Analysis for Dynamic Coupling Measures, in: Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative Research, CASCON ’06, IBM Corp., Riverton, NJ, USA, doi:10.1145/1188966.1188980, URL http://dx.doi.org/10.1145/1188966.1188980, 2006.

- J. Cleland-Huang, C. Chang, H. Kim, A. Balakrishnan, D. Nassar, H. Ammar, A. Mili, Requirements-based dynamic metrics in object-oriented systems, 5th IEEE International Symposium on Requirements Engineering (2001) 212–219.

- B. Újházi, R. Ferenc, D. Poshyvanyk, T. Gyimóthy, New conceptual coupling and cohesion metrics for object-oriented systems, 10th IEEE International Working Conference on Source Code Analysis and Manipulation 1120 (SCAM’10) (2010) 33–42.

- M. Gethers, D. Poshyvanyk, Using Relational Topic Models to capture coupling among classes in object-oriented software systems, 10th IEEE International Working Conference on Software Maintenance (ICSM) (2010) 1–10.

- M. Revelle, M. Gethers, D. Poshyvanyk, Using structural and textual information to capture feature coupling in object-oriented software, Empirical Software Engineering 16 (2011) 773–811.

- D. Poshyvanyk, Y. Guéhéneuc, A. Marcus, G. Antoniol, V. Rajlich, Feature Location Using Probabilistic Ranking of Methods Based on Execution Scenarios and Information Retrieval, IEEE Transactions on Software Engineering 33 (2007) 420–432.

- H. Kagdi, M. Gethers, D. Poshyvanyk, M. L. Collard, Blending Conceptual and Evolutionary Couplings to Support Change Impact Analysis in Source Code, in: 2010 17th Working Conference on Reverse Engineering, 1135 ISSN 2375-5369, 119–128, doi:10.1109/WCRE.2010.21, 2010.

- M. Gethers, A. Aryani, D. Poshyvanyk, Combining Conceptual and Domain-Based Couplings to Detect Database and Code Dependencies, IEEE 12th International Working Conference on Source Code Analysis and Manipulation (SCAM) (2012) 144–153.

- A. Aryani, I. Peake, M. Hamilton, Domain-based change propagation analysis: An enterprise system case study, IEEE International Conference on Software Maintenance (ICSM) (2010) 1–9.

- A. Aryani, F. Perin, M. Lungu, A. Mahmood, O. Nierstrasz, Can we predict depencencies using domain information?, 18th Working Conference on Reverse Engineering (WCRE) (2011) 55–64.

- M. Rahman, A. Aryani, C. Roy, F. Perin, On the relationships between domain-based coupling and code clones: an exploratory study, Proceedings of the 2013 International Conference on Software Engineering (2013) 1265–1268.

- H. Kagdi, M. Gethers, D. Poshyvanyk, Integrating conceptual and logical couplings for change impact analysis in software, Empirical Software Engineering 18 (5) (2013) 933–969.

- M. Alenezi, K. Magel, Empirical Evaluation of a New Coupling Metric: Combining Structural and Semantic Coupling, International Journal of Comput-

ers and Applications 36 (1) (2014) 34–44, doi:10.2316/Journal.202.2014.1.202-3902,
URL https://www.tandfonline.com/doi/abs/10.2316/Journal.202.2014.1.202-3902.

- H. Gall, K. Hajek, M. Jazayeri, Detection of Logical Coupling Based on Product Release History, Proceedings of the International Conference on Software Maintenance, 1998 (1998) 190–198.

- H. Gall, M. Jazayeri, J. Krajewski, CVS Release History Data For Detecting Logical Couplings, Proceedings of the 6th International Workshop on Principles of Software Evolution (2003) 13–23.

- A. Alali, B. Bartman, C. Newman, J. Maletic, A Preliminary Investigation of Using Age and Distance Measures in Detection of Evolutionary Couplings, 10th IEEE Working Conference on Mining Software Repositories (MSR) (2013) 169–172.

- M. D'Ambros, M. Lanza, R. Robbes, On the Relationship Between Change Coupling and Software Defects, in: 2009 16th Working Conference on Reverse Engineering, ISSN 1095-1350, 135–144, doi:10.1109/WCRE.2009.19, 2009.

- L. Zou, M. Godfrey, A. Hassan, Detecting Interaction Coupling from Task Interaction Histories, 15th IEEE Internation Conference on Program Comprehension, ICPC '07 (2007) 135–144.

- F. Bantelay, M. B. Zanjani, H. Kagdi, Comparing and combining evolutionary couplings from interactions and commits, in: 2013 20th Working Conference on Reverse Engineering (WCRE), ISSN 1095–1350, 311–320, doi:10.1109/WCRE.2013.6671306, 2013.

- S. Kramer, H. Kaindl, Coupling and cohesion metrics for knowledge-based systems using frames and rules, ACM Transactions on Software Engineering and Methodology 13 (2004) 332–358.

-A. Orme, H. Tao, L. Etzkorn, Coupling metrics for ontology –based system, IEEE Software 23 (2006) 102–108.

- H. Jordan, R. Collier, Evaluating Agent-Oriented Programs: Towards

Multi-paradigm Metrics, International Workshop on Programming Multi Agent Systems (2010) 63–78.

- J. Garcia, F. Garcia, R. Theron, Visualizing Semantic Coupling among Entities in an OWL Ontology, Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science, ONTOSE 2010 (2010) 90–106.

- J. García, F. García, R. Therón, Defining Coupling Metrics among Classes in an OWL Ontology, in: N. García-Pedrajas, F. Herrera, C. Fyfe, J. M. Benítez, M. Ali (Eds.), Trends in Applied Intelligent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-642-13025-0, 12–17, 1195 2010.

- C. Sant'Anna, A. Garcia, C. Chavez, C. Lucena, A. von Staa, On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework, in: Proceedings of Brazilian Symposium on Software Engineering, 19–34, 2003.

- M. Ceccato, P. Tonella, Measuring the Effects of Software Aspectization, 2004.

- H. Shen, J. Zhao, An evaluation of coupling metrics for aspect-oriented software, Tech. Rep., 2007.

- R. Burrows, F. C. Ferrari, A. Garcia, F. Taiani, An empirical evaluation of coupling metrics on aspect-oriented programs, Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics (WETSoM '10) (2010) 53–58.

- M. L. Bernardi, G. A. D. Lucca, A metric model for aspects' coupling, in: WETSoM, 2010.

- B. Bennett, F. Mitropoulos, New metrics for assessing aspect coupling, in: SoutheastCon 2016, 1–8, 2016.

- B. T. Bennett, Using hierarchical agglomerative clustering to locate potential aspect interference, in: SoutheastCon 2017, 1–8, 2017.

- J. Zhao, Measuring Coupling in Aspect-Oriented Systems, in: Information Processing Society of Japan (IPSJ), 14–16, 2004.

- T. Tonelli Bartolomei, A. Garcia, C. Sant'Anna, E. Figueiredo, Towards a Unified Coupling Framework for Measuring Aspect-Oriented Programs, in: 3rd InternationalWorkshop on Software Quality Assurance (SOQUA), ACM Press, ACM Press, New York, NY, USA, ISBN 1-59593-584-3, 46–53, doi:http://doi.acm.org/10.1145/1188895.1188907, 2006.

- A. Kumar, R. Kumar, P. S. Grover, Generalized Coupling Measure for Aspect-oriented Systems, SIGSOFT Softw. Eng. Notes 34 (3) (2009) 1–6, ISSN 0163-5948, doi:10.1145/1527202.1527209,
URL http://doi.acm. org/10.1145/1527202.1527209.

- K. Qian, J. Liu, F. Tsui, Decoupling Metrics for Services Composition, in: 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse (ICIS-COMSAR'06), 44–47, 2006.

- R. Sindhgatta, B. Sengupta, K. Ponnalagu, Measuring the Quality of Service Oriented Design, Springer Berlin Heidelberg, Berlin, Heidelberg, 485–499, 2009.

- A. A. M. Elhag, R. Mohamad, Metrics for evaluating the quality of service oriented design, in: 2014 8th. Malaysian Software Engineering Conference 1235 (MySEC), 154–159, doi:10.1109/MySec.2014.6986006, 2014.

- P. Quynh, H. Thang, Dynamic Coupling Metrics for Service –Oriented Software, International Journal of Computer, Electrical, Automation, Control and Information Engineering 3 (2009) 795–800.

- X. Wang, Metrics for Evaluating Coupling and Service Granularity in Service Oriented Architecture, in: 2009 International Conference on In-formation Engineering and Computer Science, ISSN 2156-7379, 1–4,
doi: 10.1109/ICIECS.2009.5362767, 2009.

- S. Alahmari, E. Zaluska, D. C. D. Roure, A Metrics Framework for Evaluating SOA Service Granularity, in: 2011 IEEE International Conference on Services Computing, 512–519, doi:10.1109/SCC.2011.98, 2011.

- T. Karhikeyan, J. Geetha, A metrics suite and fuzzy model for measuring coupling in Service Oriented Architecture, in: 2012 International Conference on Recent Advances in Computing and Software Systems, 254–259, doi:10.1109/RACSS.2012.6212677, 2012.

- M. Perepletchikov, C. Ryan, K. Frampton, Z. Tari, Coupling Metrics for Predicting Maintainability in Service-Oriented Designs, in: 2007 Australian Software Engineering Conference (ASWEC'07), ISSN 1530-0803, 329–340, doi:10.1109/ASWEC.2007.17, 2007.

- A. Kazemi, A. Azizkandi, A. Rostampour, H. Haghihi, P. Jamshidi, F. Shams, Measuring the Conceptual Coupling of Services Using Latent Semantic Indexing, IEEE International Conference on Service Computing 1260 (SCC) (2011) 504–511.

- B. Alshammari, C. Fidge, D. Corney, Security Metrics for Object-Oriented Designs, 21st Australian Software Engineering Conference (ASWEC) (2010) 55–64.

- S. Kamble, X. Jin, N. Niu, M. Simon, A Novel Coupling Pattern in Computational Science and Engineering Software, in: 2017 IEEE/ACM 12th International Workshop on Software Engineering for Science (SE4Science), 9–12, doi:10.1109/SE4Science.2017.10, 2017.

M. Ahmed, M. Shoaib, Novel Design Metrics to Measure Real Time Environment Application Design, Journal of American Science 7 (2011) 222.

- D. Spinellis, Tool writing: a forgotten art? (software tools), IEEE Software 22 (4) (2005) 9–11.

- S. Husein, A. Oxley, A Coupling and Cohesion Metrics Suite for Object-Oriented Software, in: Proceedings of the 2009 International Conference on Computer Technology and Development - Volume 01, ICCTD '09, IEEE Computer Society, Washington, DC, USA, ISBN 978-0-7695-3892-1, 421–425, 2009.

- M. Eichberg, D. Germanus, M. Mezini, L. Mrokon, T. Schafer, QScope: an open, extensible framework for measuring software projects, in: Conference on

Software Maintenance and Reengineering (CSMR'06), 10 pp.–122, 2006.

- N. Kayarvizhy, S. Kanmani, An Automated Tool for Computing Object Oriented Metrics Using XML, Springer Berlin Heidelberg, Berlin, Heidelberg, 69–79, 2011.

- M. Scotto, A. Sillitti, G. Succi, T. Vernazza, A Relational Approach to Software Metrics, in: Proceedings of the 2004 ACM Symposium on Applied Computing, SAC '04, ACM, New York, NY, USA, ISBN 1- 58113-812-1, 1536–1540, doi:10.1145/967900.968207,
URL http://doi.acm.org/10.1145/967900.968207, 2004.

- M. Scotto, A. Sillitti, G. Succi, T. Vernazza, Non-invasive Product Metrics Collection: An Architecture, in: Proceedings of the 2004 Workshop on Quantitative Techniques for Software Agile Process, QUTE-SWAP '04, ACM, New York, NY, USA, 76–78, doi:10.1145/1151433.1151444,
URL http://doi.acm.org/10.1145/1151433.1151444, 2004.

- M. Scotto, A. Sillitti, G. Succi, T. Vernazza, A non-invasive approach to product metrics collection, Journal of Systems Architecture 52 (11) (2006) 668–675, agile Methodologies for Software Production.

- M. Wilhelm, S. Diehl, Dependency Viewer - A Tool for Visualizing Package Design Quality Metrics, in: 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis, 1–2, doi: 1305 10.1109/VISSOF.2005.1684321, 2005.

- J. Alghamdi, R. Rufai, S. Khan, OOMeter: A Software Quality Assurance Tool, in: Ninth European Conference on Software Maintenance and Reengineering, ISSN 1534-5351, 190–191, doi:10.1109/CSMR.2005.44, 2005.

- B. Alshammari, C. Fidge, D. Corney, An Automated Tool for Assessing Security-Critical Designs and Programs, in: WIAR 2012; National Workshop on Information Assurance Research, 1–10, 2012.

- M. R. Girgis, T. M. Mahmoud, R. R. Nour, UML class diagram metrics tool, in: 2009 International Conference on Computer Engineering Systems, 423–428, doi:10.1109/ICCES.2009.5383226, 2009.

- J. Alghamdi, M. Elish, M. Ahmed, A tool for measuring inheritance coupling in object-oriented systems, Information Sciences 140 (3) (2002) 217–227, ISSN 0020-0255, doi:https://doi.org/10.1016/S0020-0255(01)00172-4, URL http://www.sciencedirect.com/science/article/pii/S0020025501001724, software Engineering: Systems and Tools.

- V. Bidve, P. Sarasu, Tool for Measuring Coupling in Object- Oriented Java Software 8 (2016) 812–820.

- J. Offutt, A. Abdurazik, S. R. Schach, Quantitatively measuring object oriented couplings, Software Quality Journal 16 (4) (2008) 489–512, ISSN 1573–1367, doi:10.1007/s11219-008-9051-x, URL https://doi.org/10.1007/s11219-008-9051-x.

- P. Rosner, S. Viswanathan, Visualization of Coupling and Programming to Interface for Object-Oriented Systems, in: 2008 12th International Conference Information Visualisation, ISSN 1550-6037, 575–581, 1335 doi:10.1109/IV.2008.96, 2008.

- S. Sarvari, P. Singh, G. Sikka, Efficient and Scalable Collection of Dynamic Metrics Using MapReduce, Asia-Pacific Software Engineering Conference (APSEC), 2015 (2015) 127–134.

- A. Dogra, H. Singh, P. Singh, Execution Trace Streaming Based Real Time Collection of Dynamic Metrics Using PaaS, 8th Workshop on Emerging Trends in Software Metrics (WETSoM) (2017) 43–48.

- P. Singh, H. Singh, DynaMetrics: a runtime metric-based analysis tool for object-oriented software systems, ACM SIGSOFT Software Engineering Notes 33 (2008) 1–6.

- T. Zimmermann, P. Weisgerber, S. Diehl, A. Zeller, Mining Version Histories to Guide Software Changes, in: Proceedings of the 26th International Conference on Software Engineering, ICSE '04, IEEE Computer Society, Washington, DC, USA, ISBN 0-7695-2163-0, 563–572, 2004.

- M. D'Ambros, M. Lanza, Reverse Engineering with Logical Coupling, in: 2006 13th Working Conference on Reverse Engineering, ISSN 1095-1350, 189–198, doi:10.1109/WCRE.2006.51, 2006.

- M. D'Ambros, M. Lanza, M. Lungu, Visualizing Co-Change Information with the Evolution Radar, IEEE Transactions on Software Engineering 35 (2009) 720–735.

- M. D'Ambros, M. Lanza, M. Lungu, The Evolution Radar: Visualizing Integrated Logical Coupling Information, in: Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR '06, ACM, New York, NY, USA, ISBN 1-59593-397-2, 26–32, doi:10.1145/1137983.1137992, URL http://doi.acm.org/10.1145/1137983.1137992, 2006.

- A. Ying, G. Murphy, R. Ng, M. Chu-Carroll, Predicting Source Code Changes by Mining Change History, IEEE Transcription Software Engineering 30 (9) (2004) 574–586, ISSN 0098-5589, doi:10.1109/TSE.2004.52.

- N. Hanakawa, Visualization for Software Evolution Based on Logical Coupling and Module Coupling, in: 14th Asia-Pacific Software Engineering 1365 Conference (APSEC'07), ISSN 1530-1362, 214–221, doi:10.1109/ASPEC.2007.36, 2007.

- J. Garcia, F. Garcia, R. Therón, Modelling Relationships among Classes as Semantic Coupling in OWL Ontologies, 2011.

- E. Figueiredo, A. Garcia, C. Sant'Anna, U. Kulesza, C. Lucena, Assessing aspect-oriented artifacts: Towards a tool-supported quantitative method, in: Proc. of the 9th ECOOP Workshop on Quantitative Approaches in OO Software, 2005.

- E. Figueiredo, A. Garcia, C. Lucena, AJATO: an AspectJ Assessment Tool, Proceedings of European Conference on Object Oriented Programming (ECOOP Demo).

- J. Zhao, S. Zhang, H. Shen, An Empirical Study of Maintainability in Aspect-Oriented System Evolution Using Coupling Metrics, in: 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of SoftwareEngineering(TASE), vol. 00, 233–236, doi:10.1109/TASE.2008.17, doi.ieeecomputersociety.org/10.1109/TASE.2008.17, 2008.

- A. S. Núñez-Varela, H. G. Pérez-González, F. E. Martínez-Pérez, J. Cuevas-Tello, Building a User Oriented Application for Generic Source Code Metrics Extraction from a Metrics Framework, in: 2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT), 27–32, doi:10.1109/CONISOFT.2016.13, 2016.

- P.Weißgerber, L. V. Klenze, M. Burch, S. Diehl, Exploring Evolutionary Coupling in Eclipse, 2005.

- T. Zimmermann, V. Dallmeier, K. Halachev, A. Zeller, eROSE: guiding programmers in eclipse, in: OOPSLA Companion, 2005.

- A. Tahir, S. MacDonell, A systematic mapping study on dynamic metrics and software quality, 28th IEEE International Conference on Software Maintenance (ICSM) (2012) 326–335.

- R. Harrison, S. Counsell, R. Nithi, Coupling metrics for object-oriented design, in: Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262), 150–157, doi:10.1109/METRIC. 1998.731240, 1998.

- P. Joshi, R. K. Joshi, Microscopic coupling metrics for refactoring, in: Conference on Software Maintenance and Reengineering (CSMR'06), ISSN 1534–5351, 8 pp.–152, doi:10.1109/CSMR.2006.32, 2006.

- M. English, T. Cahill, J. Buckley, Construct specific coupling measurement for C++ software, Computer Languages, Systems and Structures 38 (4) (2012) 300–319, ISSN 1477-8424, doi:https://doi.org/10.1016/j.cl.2012.06.002, URL http://www.sciencedirect.com/science/article/ pii/S1477842412000243.

- A. Tripathi, D. S. Kushwaha, A metric for package level coupling, CSI Transactions on ICT 2 (4) (2015) 217–233, ISSN 2277-9086, doi:10.1007/s40012-015-0061-0, URL https://doi.org/10.1007/s40012-015-0061-0.

- C. Rajaraman, M. R. Lyu, Some Coupling Measures for C++ Programs, in: Proc. TOOLS USA 92 Conference, 225–234, 1992.

- Y. Hassoun, S. Counsell, R. Johnson, Dynamic coupling metric: proof of concept, IEE Proceedings - Software 152 (6) (2005) 273–279, ISSN 1462-5970, doi:10.1049/ip-sen:20045067.

- H. Washizaki, T. Nakagawa, Y. Saito, Y. Fukazawa, A Coupling-based Complexity Metric for Remote Component-based Software Systems Toward Maintainability Estimation, in: 2006 13th Asia Pacific Software Engineering Conference (APSEC'06), ISSN 1530-1362, 79–86,
doi:10.1109/ APSEC.2006.3, 2006.

- E. Tempero, P. Ralph, A Model for Defining Coupling Metrics, in: 23rd Asia-Pacific Software Engineering Conference (APSEC), ISSN 1530-1362, 145–152, doi:10.1109/APSEC.2016.030, 2016.

- S. Eski, F. Buzluca, An Empirical Study on Object-Oriented Metrics and Software Evolution in Order to Reduce Testing Costs by Predicting Change-Prone Classes, in: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, 566–571, 2011.

- M. O. Elish, M. Al-Rahman Al-Khiaty, A suite of metrics for quantifying historical changes to predict future change–prone classes in object –oriented software, in: J. Softw.: Evol. and Proc., 407–437, 2013.

References

[1] M. M. Lehman, Programs, life cycles, and laws of software evolution, Proceedings of the IEEE 68 (9) (1980) 1060–1076.

[2] D. Coleman, B. Lowther, P. Oman, The application of software maintainability models in industrial software systems, Journal of Systems and Software 29 (1) (1995) 3 – 16, ISSN 0164-1212, doi:https://doi.org/10.1016/0164-1212(94)00125-7, URL `http://www.sciencedirect.com/science/article/pii/0164121294001257`, oregon Metric Workshop.

[3] M. Riaz, E. Mendes, E. Tempero, A Systematic Review of Software Maintainability Prediction and Metrics, in: Proceedings of the 2009 3rd

International Symposium on Empirical Software Engineering and Measurement, ESEM '09, IEEE Computer Society, Washington, DC, USA, ISBN 978-1-4244-4842-5, 367–377, doi:10.1109/ESEM.2009.5314233, URL `http://dx.doi.org/10.1109/ESEM.2009.5314233`, 2009.

[4] A. L. Goel, Software Reliability Models: Assumptions, Limitations, and Applicability, IEEE Transactions on Software Engineering SE-11 (12) (1985) 1411–1423, ISSN 0098-5589, doi:10.1109/TSE.1985.232177.

[5] A. Offutt, M. Harrold, P. Kolte, A software metric system for module coupling, Journal of Systems and Software 20 (1993) 295–308.

[6] J. B. N. Fenton, Software Metrics: A rigorous and practical approach, 2014.

[7] R. Robbes, D. Pollet, M. Lanza, Logical Coupling Based on Fine-Grained Change Information, 15th Working Conference on Reverse Engineering, WCRE '08 (2008) 42–46.

[8] S. Chidamber, C. Kemerer, Towards a metrics suite for object oriented design, Conference Proceedings on Object-oriented programming systems, languages, and applications (OOPSLA '91) (1991) 197–211.

[9] G. Bavota, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, A. De Lucia, An Empirical Study Of The Developers' Perception Of Software Coupling, Proceedings of the 2013 International Conference on Software Engineering (ICSE '13) (2013) 692–701.

[10] E. Arisholm, L. Briand, A. Foyen, Dynamic Coupling Measurement For Object-oriented Software, IEEE Transactions on Software Engineering 30 (8) (2004) 491–506.

[11] D. Poshyvanyk, A. Marcus, The Conceptual Coupling Metrics for Object-Oriented Systems, ICSM '06 Proceedings of the 22nd IEEE International Conference on Software Maintenance (2006) 469–478.

[12] S. Kirbas, T. Hall, A. Sen, Evolutionary coupling measurement: Making sense of the current chaos, Science of Computer Programming 135 (2017) 4–19.

[13] A. Nicolaescu, H. Lichter, Y. Xu, Evolution of Object Oriented Coupling Metrics: A sampling of 25 years of research, IEEE/ACM 2nd International Workshop on Software Architecture and Metrics (SAM) (2015) 48–54.

[14] B. Kitchenham, Procedures for Performing Systematic Reviews, 2004.

[15] C. Wohlin, Guidelines for Snowballing In Systematic Literature Studies and a Replication In Software Engineering, Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering (EASE 14) (2014) 38:1–38:10.

[16] L. Briand, P. Devanbu, W. Melo, An Investigation into Coupling Measures for C++, Proceeding of the 1997 International Conference on Software Engineering (1997) 412–421.

[17] D. Poshyvanyk, A. Marcus, R. Ferenc, T. Gyimóthy, Using information retrieval based coupling measures for impact analysis, Empirical Software Engineering 14 (1) (2009) 5–32.

[18] J. Alghamdi, Measuring Software Coupling, in: Proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems, SEPADS'07, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, ISBN 978-960-8457-59-1, 6–12, URL http://dl.acm.org/citation.cfm?id=1353801.1353803, 2007.

[19] G. Myers, Reliable software through composite design, Mason and Lipscomb, 1975.

[20] S. Schach, Object-Oriented and Classical Software Engineering, 8th edition, McGraw-Hill, 2011.

[21] S. Henry, D. Kafura, Software Structure Metrics Based on Information Flow, IEEE Transactions on Software Engineering SE-7 (5) (1981) 510–518.

[22] E. Allen, T. Khoshgoftaar, Y. Chen, Measuring coupling and cohesion of software modules: an information-theory approach, Proceedings of the 7th International Software Metrics Symposium, 2011. METRICS 2001 (2001) 124–134.

[23] E. B. Allen, S. Gottipati, R. Govindarajan, Measuring size, complexity, and coupling of hypergraph abstractions of software: An information-theory approach, Software Quality Journal 15 (2) (2007) 179–212, ISSN 1573-1367, doi:10.1007/s11219-006-9010-3, URL https://doi.org/10.1007/s11219-006-9010-3.

[24] J. Eder, G. Kappel, M. Schrefl, Coupling and Cohesion in Object-Oriented Systems, 1992.

[25] S. Chidamber, C. Kemerer, A metrics suite for object oriented design, IEEE Transactions on Software Engineering 20 (6) (1994) 476 – 493.

[26] M. Hitz, B. Montazeri, Measuring Coupling and Cohesion In Object-Oriented Systems, Proceedings of International Symposium on Applied Corporate Computing 35 (1995) 25–27.

[27] L. Briand, S. Morasca, V. R. Basili, Property-based software engineering measurement, IEEE Transactions on Software Engineering 22 (1) (1996) 68–86, ISSN 0098-5589.

[28] L. Briand, J. Daly, J. Wust, A unified framework for coupling measurement in object-oriented systems, IEEE Transactions on Software Engineering 25 (1999) 91–121.

[29] W. Li, S. Henry, Object-Oriented Metrics that Predict Maintainability, Journal of Systems and Software 23 (1993) 111–122.

[30] R. Martin, Object oriented design quality metrics: an analysis of dependencies, vol. 2, 2006.

[31] F. Abreu, R. Esteves, M. Goulão, Toward the Design Quality Evaluation of Object-Oriented Software, 1995.

[32] F. Abreau, R. Carapuça, Object-Oriented Software Engineering: Measuring and Controlling the Development Process, in: Proceedings of the 4th International Conference on Software Quality, 1994.

[33] Y. Lee, B. Liang, S. Wu, F. Wang, Measuring the Coupling and Cohesion of an Object-Oriented Program Based On Information Flow, in: Proceedings of the International Conference on Software Quality, 1995.

[34] W. Li, Another Metric Suite for Object-oriented Programming, Journal of System and Software 44 (2) (1998) 155–162, ISSN 0164-1212.

[35] J. Rilling, W. J. Meng, O. Ormandjieva, Context driven slicing based coupling measures, in: 20th IEEE International Conference on Software Maintenance, 2004. Proceedings., ISSN 1063-6773, 532–, doi:10.1109/ICSM.2004.1357874, 2004.

[36] M. English, J. Buckley, T. Cahill, Fine-Grained Software Metrics in Practice, in: First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), ISSN 1949-3770, 295–304, doi:10.1109/ESEM.2007.32, 2007.

[37] H. Li, A Novel Coupling Metric for Object-Oriented Software Systems, in: 2008 IEEE International Symposium on Knowledge Acquisition and Modeling Workshop, 609–612, doi:10.1109/KAMW.2008.4810562, 2008.

[38] H. Y. Yang, E. Tempero, R. Berrigan, Detecting indirect coupling, in: 2005 Australian Software Engineering Conference, ISSN 1530-0803, 212–221, doi:10.1109/ASWEC.2005.22, 2005.

[39] H. Y. Yang, E. Tempero, Indirect Coupling As a Criteria for Modularity, in: First International Workshop on Assessment of Contemporary Modularization Techniques (ACoM '07), 10–10, doi:10.1109/ACOM.2007.5, 2007.

[40] S. Almugrin, W. Albattah, A. Melton, Using indirect coupling metrics to predict package maintainability and testability, Journal of Systems and Software 121 (2016) 298 – 310, ISSN 0164-1212, doi:https://doi.org/10.1016/j.jss.2016.02.024, URL http://www.sciencedirect.com/science/article/pii/S016412121600056X.

[41] M. Tang, M. Kao, M. Chen, An empirical study on object-oriented metrics, in: Proceedings Sixth International Software Metrics Symposium (Cat. No.PR00403), 242–249, 1999.

[42] G. Gui, P. D. Scott, New Coupling and Cohesion Metrics for Evaluation of Software Component Reusability, in: 2008 The 9th International Conference for Young Computer Scientists, 1181–1186, doi:10.1109/ICYCS.2008.270, 2008.

[43] A. Aloysius, L. Arockiam, Coupling Complexity Metric: A Cognitive Approach, International Journal of Information Technology and Computer Science 4 (2012) 29–35.

[44] A. Mitchell, J. Power, An empirical investigation into the dimensions of run-time coupling in Java programs, Proceedings of the 3rd international symposium on Principles and practice of programming in Java (2004) 9–14.

[45] R. Geetika, P. Singh, Dynamic Coupling Metrics for Object Oriented Software Systems: A Survey, SIGSOFT Softw. Eng. Notes 39 (2) (2014) 1–8, ISSN 0163-5948, doi:10.1145/2579281.2579296, URL http://doi.acm.org/10.1145/2579281.2579296.

[46] E. Arisholm, Dynamic coupling measures for object-oriented software, Proceeding of the 8th IEEE Symposium on Software Metrics (2002) 33–42.

[47] A. Mitchell, J. Power, Using object-level run-time metrics to study coupling between objects, Proceedings of the 2005 ACM symposium on Applied computing (SAC '05) (2005) 1456–1452.

[48] S. Yacoub, H. Ammar, T. Robinson, Dynamic metrics for object oriented design, Proceedings of the 6th Software Metrics Symposium (1999) 50–61.

[49] A. Zaidman, S. Demeyer, Analyzing large event traces with the help of coupling metrics, in: Proceedings of the Fourth International Workshop on OO Reengineering, 2004.

[50] Y. Hassoun, R. Johnson, S. Counsell, A dynamic runtime coupling for meta-level architectures, Proceedings of the 8th European Conference on Software Maintenance and Reengineering, CSMR 2004 (2004) 339–346.

[51] R. Gunnalan, M. Shereshevsky, H. H. Ammar, Pseudo dynamic metrics [software metrics], in: The 3rd ACS/IEEE International Conference on-Computer Systems and Applications, 2005., ISSN 2161-5322, 117–, doi: 10.1109/AICCSA.2005.1387106, 2005.

[52] Y. Liu, A. Milanova, Static Analysis for Dynamic Coupling Measures, in: Proceedings of the 2006 Conference of the Center for Advanced Studies on Collaborative Research, CASCON '06, IBM Corp., Riverton, NJ, USA, doi:10.1145/1188966.1188980, URL `http://dx.doi.org/10.1145/1188966.1188980`, 2006.

[53] J. Cleland-Huang, C. Chang, H. Kim, A. Balakrishnan, D. Nassar, H. Ammar, A. Mili, Requirements-based dynamic metrics in object-oriented systems, 5th IEEE International Symposium on Requirements Engineering (2001) 212–219.

[54] H. Abualese, P. Sumari, T. Al-Rousan, M. R. Al-Mousa, Utility classes detection metrics for execution trace analysis, in: 2017 8th Interna-

tional Conference on Information Technology (ICIT), 469–474, doi:10.1109/ICITECH.2017.8080044, 2017.

[55] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, R. Harshman, Indexing by Latent Semantic Analysis, Journal of the American Society for Information Science 41 (1990) 391–407.

[56] B. Újházi, R. Ferenc, D. Poshyvanyk, T. Gyimóthy, New conceptual coupling and cohesion metrics for object-oriented systems, 10th IEEE International Working Conference on Source Code Analysis and Manipulation (SCAM'10) (2010) 33–42.

[57] M. Gethers, D. Poshyvanyk, Using Relational Topic Models to capture coupling among classes in object-oriented software systems, 10th IEEE International Working Conference on Software Maintenance (ICSM) (2010) 1–10.

[58] M. Revelle, M. Gethers, D. Poshyvanyk, Using structural and textual information to capture feature coupling in object-oriented software, Empirical Software Engineering 16 (2011) 773–811.

[59] D. Poshyvanyk, Y. Guéhéneuc, A. Marcus, G. Antoniol, V. Rajlich, Feature Location Using Probabilistic Ranking of Methods Based on Execution Scenarios and Information Retrieval, IEEE Transactions on Software Engineering 33 (2007) 420–432.

[60] H. Kagdi, M. Gethers, D. Poshyvanyk, M. L. Collard, Blending Conceptual and Evolutionary Couplings to Support Change Impact Analysis in Source Code, in: 2010 17th Working Conference on Reverse Engineering, ISSN 2375-5369, 119–128, doi:10.1109/WCRE.2010.21, 2010.

[61] M. Gethers, A. Aryani, D. Poshyvanyk, Combining Conceptual and Domain-Based Couplings to Detect Database and Code Dependencies, IEEE 12th International Working Conference on Source Code Analysis and Manipulation (SCAM) (2012) 144–153.

[62] A. Aryani, I. Peake, M. Hamilton, Domain-based change propagation analysis: An enterprise system case study, IEEE International Conference on Software Maintenance (ICSM) (2010) 1–9.

[63] A. Aryani, F. Perin, M. Lungu, A. Mahmood, O. Nierstrasz, Can we predict depencencies using domain information?, 18th Working Conference on Reverse Engineering (WCRE) (2011) 55–64.

[64] M. Rahman, A. Aryani, C. Roy, F. Perin, On the relationships between domain-based coupling and code clones: an exploratory study, Proceedings of the 2013 International Conference on Software Engineering (2013) 1265–1268.

[65] H. Kagdi, M. Gethers, D. Poshyvanyk, Integrating conceptual and logical couplings for change impact analysis in software, Empirical Software Engineering 18 (5) (2013) 933–969.

[66] M. Alenezi, K. Magel, Empirical Evaluation of a New Coupling Metric: Combining Structural and Semantic Coupling, International Journal of Computers and Applications 36 (1) (2014) 34–44, doi:10.2316/Journal.202.2014.1.202-3902, URL https://www.tandfonline.com/doi/abs/10.2316/Journal.202.2014.1.202-3902.

[67] H. Gall, K. Hajek, M. Jazayeri, Detection of Logical Coupling Based on Product Release History, Proceedings of the International Conference on Software Maintenance, 1998 (1998) 190–198.

[68] H. Gall, M. Jazayeri, J. Krajewski, CVS Release History Data For Detecting Logical Couplings, Proceedings of the 6th International Workshop on Principles of Software Evolution (2003) 13–23.

[69] A. Alali, B. Bartman, C. Newman, J. Maletic, A Preliminary Investigation of Using Age and Distance Measures in Detection of Evolutionary Couplings, 10th IEEE Working Conference on Mining Software Repositories (MSR) (2013) 169–172.

[70] M. D'Ambros, M. Lanza, R. Robbes, On the Relationship Between Change Coupling and Software Defects, in: 2009 16th Working Conference on Reverse Engineering, ISSN 1095-1350, 135–144, doi:10.1109/WCRE.2009.19, 2009.

[71] L. Zou, M. Godfrey, A. Hassan, Detecting Interaction Coupling from Task Interaction Histories, 15th IEEE Internation Conference on Program Comprehension, ICPC '07 (2007) 135–144.

[72] F. Bantelay, M. B. Zanjani, H. Kagdi, Comparing and combining evolutionary couplings from interactions and commits, in: 2013 20th Working Conference on Reverse Engineering (WCRE), ISSN 1095-1350, 311–320, doi:10.1109/WCRE.2013.6671306, 2013.

[73] S. Kramer, H. Kaindl, Coupling and cohesion metrics for knowledge-based systems using frames and rules, ACM Transactions on Software Engineering and Methodology 13 (2004) 332–358.

[74] A. Orme, H. Tao, L. Etzkorn, Coupling metrics for ontology-based system, IEEE Software 23 (2006) 102–108.

[75] H. Jordan, R. Collier, Evaluating Agent-Oriented Programs: Towards Multi-paradigm Metrics, International Workshop on Programming Multi-Agent Systems (2010) 63–78.

[76] J. Garcia, F. Garcia, R. Theron, Visualizing Semantic Coupling among Entities in an OWL Ontology, Ontology, Conceptualization and Epistemology for Information Systems, Software Engineering and Service Science, ONTOSE 2010 (2010) 90–106.

[77] J. García, F. García, R. Therón, Defining Coupling Metrics among Classes in an OWL Ontology, in: N. García-Pedrajas, F. Herrera, C. Fyfe, J. M. Benítez, M. Ali (Eds.), Trends in Applied Intelligent Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-642-13025-0, 12–17, 2010.

[78] C. Sant'Anna, A. Garcia, C. Chavez, C. Lucena, A. von Staa, On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework, in: Proceedings of Brazilian Symposium on Software Engineering, 19–34, 2003.

[79] M. Ceccato, P. Tonella, Measuring the Effects of Software Aspectization, 2004.

[80] H. Shen, J. Zhao, An evaluation of coupling metrics for aspect-oriented software, Tech. Rep., 2007.

[81] R. Burrows, F. C. Ferrari, A. Garcia, F. Taiani, An empirical evaluation of coupling metrics on aspect-oriented programs, Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics (WETSoM '10) (2010) 53–58.

[82] M. L. Bernardi, G. A. D. Lucca, A metric model for aspects' coupling, in: WETSoM, 2010.

[83] B. Bennett, F. Mitropoulos, New metrics for assessing aspect coupling, in: SoutheastCon 2016, 1–8, 2016.

[84] B. T. Bennett, Using hierarchical agglomerative clustering to locate potential aspect interference, in: SoutheastCon 2017, 1–8, 2017.

[85] J. Zhao, Measuring Coupling in Aspect-Oriented Systems, in: Information Processing Society of Japan (IPSJ), 14–16, 2004.

[86] T. Tonelli Bartolomei, A. Garcia, C. Sant'Anna, E. Figueiredo, Towards a Unified Coupling Framework for Measuring Aspect-Oriented Programs, in: 3rd International Workshop on Software Quality Assurance (SOQUA), ACM Press, ACM Press, New York, NY, USA, ISBN 1-59593-584-3, 46–53, doi:http://doi.acm.org/10.1145/1188895.1188907, 2006.

[87] A. Kumar, R. Kumar, P. S. Grover, Generalized Coupling Measure for Aspect-oriented Systems, SIGSOFT Softw. Eng. Notes 34 (3) (2009) 1–

6, ISSN 0163-5948, doi:10.1145/1527202.1527209, URL http://doi.acm.org/10.1145/1527202.1527209.

[88] K. Qian, J. Liu, F. Tsui, Decoupling Metrics for Services Composition, in: 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering,Software Architecture and Reuse (ICIS-COMSAR'06), 44–47, 2006.

[89] R. Sindhgatta, B. Sengupta, K. Ponnalagu, Measuring the Quality of Service Oriented Design, Springer Berlin Heidelberg, Berlin, Heidelberg, 485–499, 2009.

[90] A. A. M. Elhag, R. Mohamad, Metrics for evaluating the quality of service-oriented design, in: 2014 8th. Malaysian Software Engineering Conference (MySEC), 154–159, doi:10.1109/MySec.2014.6986006, 2014.

[91] P. Quynh, H. Thang, Dynamic Coupling Metrics for Service –Oriented Software, International Journal of Computer, Electrical, Automation, Control and Information Engineering 3 (2009) 795 – 800.

[92] X. Wang, Metrics for Evaluating Coupling and Service Granularity in Service Oriented Architecture, in: 2009 International Conference on Information Engineering and Computer Science, ISSN 2156-7379, 1–4, doi:10.1109/ICIECS.2009.5362767, 2009.

[93] S. Alahmari, E. Zaluska, D. C. D. Roure, A Metrics Framework for Evaluating SOA Service Granularity, in: 2011 IEEE International Conference on Services Computing, 512–519, doi:10.1109/SCC.2011.98, 2011.

[94] T. Karhikeyan, J. Geetha, A metrics suite and fuzzy model for measuring coupling in Service Oriented Architecture, in: 2012 International Conference on Recent Advances in Computing and Software Systems, 254–259, doi:10.1109/RACSS.2012.6212677, 2012.

[95] M. Perepletchikov, C. Ryan, K. Frampton, Z. Tari, Coupling Metrics for Predicting Maintainability in Service-Oriented Designs, in: 2007 Australian Software Engineering Conference (ASWEC'07), ISSN 1530-0803, 329–340, doi:10.1109/ASWEC.2007.17, 2007.

[96] T. Erl, SOA: Principles of Service Design, The Prentice Hall Service Technology Series from Thomas Erl, Pearson Education, ISBN 9780132715836, 2007.

[97] A. Kazemi, A. Azizkandi, A. Rostampour, H. Haghihi, P. Jamshidi, F. Shams, Measuring the Conceptual Coupling of Services Using Latent Semantic Indexing, IEEE International Conference on Service Computing (SCC) (2011) 504–511.

[98] B. Alshammari, C. Fidge, D. Corney, Security Metrics for Object-Oriented Designs, 21st Australian Software Engineering Conference (ASWEC) (2010) 55–64.

[99] J. Jrjens, Secure Systems Development with UML, Springer-Verlag, Berlin, Heidelberg, ISBN 3642056350, 9783642056352, 2010.

[100] J. Barnes, High Integrity Software: The SPARK Approach to Safety and Security, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, ISBN 0321136160, 2003.

[101] S. Kamble, X. Jin, N. Niu, M. Simon, A Novel Coupling Pattern in Computational Science and Engineering Software, in: 2017 IEEE/ACM 12th International Workshop on Software Engineering for Science (SE4Science), 9–12, doi:10.1109/SE4Science.2017.10, 2017.

[102] M. Ahmed, M. Shoaib, Novel Design Metrics to Measure Real Time Environment Application Design, Journal of American Science 7 (2011) 222.

[103] D. Spinellis, Tool writing: a forgotten art? (software tools), IEEE Software 22 (4) (2005) 9–11.

[104] S. Husein, A. Oxley, A Coupling and Cohesion Metrics Suite for Object-Oriented Software, in: Proceedings of the 2009 International Conference on Computer Technology and Development - Volume 01, ICCTD '09, IEEE Computer Society, Washington, DC, USA, ISBN 978-0-7695-3892-1, 421–425, 2009.

[105] M. Eichberg, D. Germanus, M. Mezini, L. Mrokon, T. Schafer, QScope: an open, extensible framework for measuring software projects, in: Conference on Software Maintenance and Reengineering (CSMR'06), 10 pp.–122, 2006.

[106] N. Kayarvizhy, S. Kanmani, An Automated Tool for Computing Object Oriented Metrics Using XML, Springer Berlin Heidelberg, Berlin, Heidelberg, 69–79, 2011.

[107] M. Scotto, A. Sillitti, G. Succi, T. Vernazza, A Relational Approach to Software Metrics, in: Proceedings of the 2004 ACM Symposium on Applied Computing, SAC '04, ACM, New York, NY, USA, ISBN 1-58113-812-1, 1536–1540, doi:10.1145/967900.968207, URL `http://doi.acm.org/10.1145/967900.968207`, 2004.

[108] M. Scotto, A. Sillitti, G. Succi, T. Vernazza, Non-invasive Product Metrics Collection: An Architecture, in: Proceedings of the 2004 Workshop on Quantitative Techniques for Software Agile Process, QUTE-SWAP '04, ACM, New York, NY, USA, 76–78, doi:10.1145/1151433.1151444, URL `http://doi.acm.org/10.1145/1151433.1151444`, 2004.

[109] M. Scotto, A. Sillitti, G. Succi, T. Vernazza, A non-invasive approach to product metrics collection, Journal of Systems Architecture 52 (11) (2006) 668 – 675, agile Methodologies for Software Production.

[110] M. Wilhelm, S. Diehl, Dependency Viewer - A Tool for Visualizing Package Design Quality Metrics, in: 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis, 1–2, doi: 10.1109/VISSOF.2005.1684321, 2005.

[111] J. Alghamdi, R. Rufai, S. Khan, OOMeter: A Software Quality Assurance Tool, in: Ninth European Conference on Software Maintenance and Reengineering, ISSN 1534-5351, 190–191, doi:10.1109/CSMR.2005.44, 2005.

[112] M. Barnett, C. Bird, J. Brunet, S. Lahiri, Helping Developers Help Themselves: Automatic Decomposition Of Code Review Changesets, Proceedings of the 37th International Conference on Software Engineering 1 (1) (2015) 134–144.

[113] B. Alshammari, C. Fidge, D. Corney, An Automated Tool for Assessing Security-Critical Designs and Programs, in: WIAR 2012; National Workshop on Information Assurance Research, 1–10, 2012.

[114] M. R. Girgis, T. M. Mahmoud, R. R. Nour, UML class diagram metrics tool, in: 2009 International Conference on Computer Engineering Systems, 423–428, doi:10.1109/ICCES.2009.5383226, 2009.

[115] J. AlGhamdi, M. Elish, M. Ahmed, A tool for measuring inheritance coupling in object-oriented systems, Information Sciences 140 (3) (2002) 217 – 227, ISSN 0020-0255, doi:https://doi.org/10.1016/S0020-0255(01)00172-4, URL `http://www.sciencedirect.com/science/article/pii/S0020025501001724`, software Engineering: Systems and Tools.

[116] V. Bidve, P. Sarasu, Tool for Measuring Coupling in Object- Oriented Java Software 8 (2016) 812–820.

[117] J. Offutt, A. Abdurazik, S. R. Schach, Quantitatively measuring object-oriented couplings, Software Quality Journal 16 (4) (2008) 489–512, ISSN 1573-1367, doi:10.1007/s11219-008-9051-x, URL `https://doi.org/10.1007/s11219-008-9051-x`.

[118] P. Rosner, S. Viswanathan, Visualization of Coupling and Programming to Interface for Object-Oriented Systems, in: 2008 12th Interna-

tional Conference Information Visualisation, ISSN 1550-6037, 575–581, doi:10.1109/IV.2008.96, 2008.

[119] S. Sarvari, P. Singh, G. Sikka, Efficient and Scalable Collection of Dynamic Metrics Using MapReduce, Asia-Pacific Software Engineering Conference (APSEC), 2015 (2015) 127–134.

[120] A. Dogra, H. Singh, P. Singh, Execution Trace Streaming Based Real Time Collection of Dynamic Metrics Using PaaS, 8th Workshop on Emerging Trends in Software Metrics (WETSoM) (2017) 43–48.

[121] P. Singh, H. Singh, DynaMetrics: a runtime metric-based analysis tool for object-oriented software systems, ACM SIGSOFT Software Engineering Notes 33 (2008) 1–6.

[122] T. Zimmermann, P. Weisgerber, S. Diehl, A. Zeller, Mining Version Histories to Guide Software Changes, in: Proceedings of the 26th International Conference on Software Engineering, ICSE '04, IEEE Computer Society, Washington, DC, USA, ISBN 0-7695-2163-0, 563–572, 2004.

[123] M. D'Ambros, M. Lanza, Reverse Engineering with Logical Coupling, in: 2006 13th Working Conference on Reverse Engineering, ISSN 1095-1350, 189–198, doi:10.1109/WCRE.2006.51, 2006.

[124] M. D'Ambros, M. Lanza, M. Lungu, Visualizing Co-Change Information with the Evolution Radar, IEEE Transactions on Software Engineering 35 (2009) 720–735.

[125] M. D'Ambros, M. Lanza, M. Lungu, The Evolution Radar: Visualizing Integrated Logical Coupling Information, in: Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR '06, ACM, New York, NY, USA, ISBN 1-59593-397-2, 26–32, doi:10.1145/1137983.1137992, URL http://doi.acm.org/10.1145/1137983.1137992, 2006.

[126] A. Ying, G. Murphy, R. Ng, M. Chu-Carroll, Predicting Source Code Changes by Mining Change History, IEEE Transcription Software Engineering 30 (9) (2004) 574–586, ISSN 0098-5589, doi:10.1109/TSE.2004.52.

[127] N. Hanakawa, Visualization for Software Evolution Based on Logical Coupling and Module Coupling, in: 14th Asia-Pacific Software Engineering Conference (APSEC'07), ISSN 1530-1362, 214–221, doi:10.1109/ASPEC.2007.36, 2007.

[128] J. Garcia, F. Garcia, R. Therón, Modelling Relationships among Classes as Semantic Coupling in OWL Ontologies, 2011.

[129] E. Figueiredo, A. Garcia, C. Sant'Anna, U. Kulesza, C. Lucena, Assessing aspect-oriented artifacts: Towards a tool-supported quantitative method, in: Proc. of the 9th ECOOP Workshop on Quantitative Approaches in OO Software, 2005.

[130] E. Figueiredo, A. Garcia, C. Lucena, AJATO: an AspectJ Assessment Tool, Proceedings of European Conference on Object Oriented Programming (ECOOP Demo) .

[131] J. Zhao, S. Zhang, H. Shen, An Empirical Study of Maintainability in Aspect-Oriented System Evolution Using Coupling Metrics, in: 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering(TASE), vol. 00, 233–236, doi:10.1109/TASE.2008.17, URL `doi.ieeecomputersociety.org/10.1109/TASE.2008.17`, 2008.

[132] A. S. Núñez-Varela, H. G. Pérez-González, F. E. Martínez-Pérez, J. Cuevas-Tello, Building a User Oriented Application for Generic Source Code Metrics Extraction from a Metrics Framework, in: 2016 4th International Conference in Software Engineering Research and Innovation (CONISOFT), 27–32, doi:10.1109/CONISOFT.2016.13, 2016.

[133] P. Weißgerber, L. V. Klenze, M. Burch, S. Diehl, Exploring Evolutionary Coupling in Eclipse, 2005.

[134] T. Zimmermann, V. Dallmeier, K. Halachev, A. Zeller, eROSE: guiding programmers in eclipse, in: OOPSLA Companion, 2005.

[135] A. Tahir, S. MacDonell, A systematic mapping study on dynamic metrics and software quality, 28th IEEE International Conference on Software Maintenance (ICSM) (2012) 326–335.

[136] A. Sarimbekov, A. Sewe, W. Binder, P. Moret, M. Mezini, JP2: Call-site aware calling context profiling for the Java Virtual Machine, Science of Computer Programming 79 (Supplement C) (2014) 146–157.

[137] R. Harrison, S. Counsell, R. Nithi, Coupling metrics for object-oriented design, in: Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262), 150–157, doi:10.1109/METRIC.1998.731240, 1998.

[138] P. Joshi, R. K. Joshi, Microscopic coupling metrics for refactoring, in: Conference on Software Maintenance and Reengineering (CSMR'06), ISSN 1534-5351, 8 pp.–152, doi:10.1109/CSMR.2006.32, 2006.

[139] M. English, T. Cahill, J. Buckley, Construct specific coupling measurement for C++ software, Computer Languages, Systems and Structures 38 (4) (2012) 300 – 319, ISSN 1477-8424, doi:https://doi.org/10.1016/j.cl.2012.06.002, URL `http://www.sciencedirect.com/science/article/pii/S1477842412000243`.

[140] A. Tripathi, D. S. Kushwaha, A metric for package level coupling, CSI Transactions on ICT 2 (4) (2015) 217–233, ISSN 2277-9086, doi:10.1007/s40012-015-0061-0, URL `https://doi.org/10.1007/s40012-015-0061-0`.

[141] C. Rajaraman, M. R. Lyu, Some Coupling Measures for C++ Programs, in: Proc. TOOLS USA 92 Conference, 225–234, 1992.

[142] V. Dixit, R. Vishwkarma, Comparison of class-level versus object-level static and dynamic coupling and cohesion measures in object oriented pro-

gramming, 11th International Conference on Wireless and Optical Communication Networks (WOCN) (2014) 1–5.

[143] Y. Hassoun, S. Counsell, R. Johnson, Dynamic coupling metric: proof of concept, IEE Proceedings - Software 152 (6) (2005) 273–279, ISSN 1462-5970, doi:10.1049/ip-sen:20045067.

[144] H. Washizaki, T. Nakagawa, Y. Saito, Y. Fukazawa, A Coupling-based Complexity Metric for Remote Component-based Software Systems Toward Maintainability Estimation, in: 2006 13th Asia Pacific Software Engineering Conference (APSEC'06), ISSN 1530-1362, 79–86, doi:10.1109/APSEC.2006.3, 2006.

[145] E. Tempero, P. Ralph, A Model for Defining Coupling Metrics, in: 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), ISSN 1530-1362, 145–152, doi:10.1109/APSEC.2016.030, 2016.

[146] B. Kitchenham, S. L. Pfleeger, N. Fenton, Towards a framework for software measurement validation, IEEE Transactions on Software Engineering 21 (12) (1995) 929–944, ISSN 0098-5589, doi:10.1109/32.489070.

[147] E. J. Weyuker, Evaluating Software Complexity Measures, IEEE Trans. Softw. Eng. 14 (9) (1988) 1357–1365, ISSN 0098-5589, doi:10.1109/32.6178, URL `https://doi.org/10.1109/32.6178`.

[148] T. Baum, K. Schneider, A. Bacchelli, On The Optimal Order Of Reading Source Code Changes for Review, IEEE International Conference on Software Maintenance and Evolution (ICSME) (2017) 329–340.

[149] T. Baum, O. Liskin, K. Niklas, K. Schneider, A Faceted Classification Scheme for Change-Based Industrial Code Review Processes, 2016 IEEE International Conference on Software Quality, Reliability and Security (QRS) (2016) 74–85.

[150] L. Pascarella, D. Spadini, F. Palomba, M. Bruntink, A. Bacchelli, Information Needs in Contemporary Code Review .

[151] Y. Kamei, E. Shihab, B. Adams, A. E. Hassan, A. Mockus, A. Sinha, N. Ubayashi, A large-scale empirical study of just-in-time quality assurance, IEEE Transactions on Software Engineering 39 (6) (2013) 757–773.

[152] M. Tufano, F. Palomba, G. Bavota, R. Oliveto, M. Di Penta, A. De Lucia, D. Poshyvanyk, When and why your code starts to smell bad (and whether the smells go away), IEEE Transactions on Software Engineering 43 (11) (2017) 1063–1088.

[153] F. Palomba, G. Bavota, M. Di Penta, R. Oliveto, D. Poshyvanyk, A. De Lucia, Mining version histories for detecting code smells, IEEE Transactions on Software Engineering 41 (5) (2015) 462–489.

[154] F. Palomba, G. Bavota, M. Di Penta, F. Fasano, R. Oliveto, A. De Lucia, On the diffuseness and the impact on maintainability of code smells: a large scale empirical investigation, Empirical Software Engineering 23 (3) (2018) 1188–1221.

[155] A. Maiga, N. Ali, N. Bhattacharya, A. Sabané, Y. Guéhéneuc, E. Aimeur, SMURF: A SVM-based Incremental Anti-pattern Detection Approach, in: 2012 19th Working Conference on Reverse Engineering, 466–475, 2012.

[156] F. Khomh, S. Vaucher, Y. Guéhéneuc, H. Sahraoui, BDTEX: A GQM-based Bayesian approach for the detection of antipatterns, Journal of Systems and Software 84 (4) (2011) 559 – 572.

[157] F. Arcelli Fontana, M. Zanoni, Code smell severity classification using machine learning techniques, Knowledge-Based Systems 128 (2017) 43 – 58.

[158] D. Di Nucci, F. Palomba, D. A. Tamburri, A. Serebrenik, A. De Lucia, Detecting code smells using machine learning techniques: are we there yet?, in: 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE, 612–621, 2018.

[159] F. Palomba, A. Panichella, A. De Lucia, R. Oliveto, A. Zaidman, A textual-based technique for Smell Detection, in: 2016 IEEE 24th International Conference on Program Comprehension (ICPC), 1–10, 2016.

[160] F. Palomba, A. Panichella, A. Zaidman, R. Oliveto, A. De Lucia, The scent of a smell: An extensive comparison between textual and structural smells, IEEE Transactions on Software Engineering .

[161] M. Fowler, K. Beck, J. Brant, W. Opdyke, D. Roberts, Refactoring: improving the design of existing code, Addison-Wesley Professional, 1999.

[162] F. Khomh, M. Di Penta, Y. Guéhéneuc, G. Antoniol, An exploratory study of the impact of antipatterns on class –change and fault –proneness, Empirical Software Engineering 17 (3) (2012) 243–275, ISSN 1573-7616, doi:10.1007/s10664-011-9171-y, URL `https://doi.org/10.1007/s10664-011-9171-y`.

[163] G. Catolino, F. Palomba, A. De Lucia, F. Ferrucci, A. Zaidman, Developer-related Factors in Change Prediction: An Empirical Assessment, in: Proceedings of the 25th International Conference on Program Comprehension, ICPC '17, IEEE Press, Piscataway, NJ, USA, 186–195, 2017.

[164] S. Eski, F. Buzluca, An Empirical Study on Object-Oriented Metrics and Software Evolution in Order to Reduce Testing Costs by Predicting Change-Prone Classes, in: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops, 566–571, 2011.

[165] M. O. Elish, M. Al-Rahman Al-Khiaty, A suite of metrics for quantifying historical changes to predict future change –prone classes in object –oriented software, in: J. Softw.: Evol. and Proc., 407–437, 2013.

[166] G. Catolino, F. Palomba, A. De Lucia, F. Ferrucci, A. Zaidman, Enhancing Change Prediction Models using Developer-Related Factors, Journal of Systems and Software 143 (9) (2018) 14–28.

Appendix A. List of selected ScienceDirect Journals

- AASRI Procedia
- Advances in Engineering Software
- Astronomy and Computing
- Computer Fraud and Security
- Computer Languages
- Computer Languages, Systems and Structures
- Computer Methods and Programs in Biomedicine
- Computer Programs in Biomedicine
- Computer Standards and Interfaces
- Data Processing
- Digital Investigation
- Egyptian Journal of Basic and Applied Sciences
- Entertainment Computing
- Environmental Modelling and Software
- Environmental Software
- Euromicro Newsletter
- Future Computing and Informatics Journal
- Future Generation Computer Systems
- Information and Software Technology
- Information Systems
- Integration

- Intelligent Data Analysis
- Journal of Computational Science
- Journal of Innovation in Digital Ecosystems
- The Journal of Logic and Algebraic Programming
- The Journal of Logic Programming
- Journal of Logical and Algebraic Methods in Programming
- Journal of Parallel and Distributed Computing
- Journal of Systems Architecture
- Journal of Systems and Software
- Journal of Web Semantics
- Microprocessing and Microprogramming
- Microprocessors
- Microprocessors and Microsystems
- Network Security
- Performance Evaluation
- Robotics
- Science of Computer Programming
- SoftwareX