Evolutionary coupling measurement: Making sense of the current chaos

Serkan Kirbas, Tracy Hall, Alper Sen

PII:	S0167-6423(16)30158-7
DOI:	http://dx.doi.org/10.1016/j.scico.2016.10.003
Reference:	SCICO 2058
To appear in:	*Science of Computer Programming*
Received date:	24 March 2016
Revised date:	19 September 2016
Accepted date:	5 October 2016

Evolutionary Coupling Measurement: Making Sense of the Current Chaos

Serkan Kirbas[a,b,*], Tracy Hall[a], Alper Sen[b]

[a] *Department of Computer Science, Brunel University London, London, United Kingdom*
[b] *Computer Engineering Department, Bogazici University, Istanbul, Turkey*

Abstract

Objective: The aim of this research is to evaluate the measurement of evolutionary coupling (EC) in software artefacts from a measurement theory perspective.

Background: Evolutionary coupling (EC) can be defined as the implicit relationship between two or more software artefacts which are frequently changed together. Previous studies on EC show that EC measures which are based on software change history information play an important role in measuring software quality and predicting defects. The many previous EC measures published are disparate and no comprehensive evaluation of the current EC measures exists. Therefore it is hard for researchers and practitioners to compare, choose and use EC measures.

Methods: We define 19 evaluation criteria based on the principles of measurement theory and metrology. We evaluate previously published EC measures by applying these criteria.

Results: Our evaluation results revealed that current EC measurement has the particular weaknesses around establishing sound empirical relation systems, defining detailed and standardised measurement procedures as well as scale type and mathematical validation.

Conclusions: We provide information about the quality of existing EC measures and measurement methods. The results suggest that there is more work to be done to put EC measurement on a firm footing that will enable the reliable measurement of EC and the accurate replication of EC measurement.

Keywords: Evolutionary coupling, Measurement, Measurement Theory

[*] Corresponding Author
Email addresses: Serkan.Kirbas@boun.edu.tr (Serkan Kirbas), Tracy.Hall@brunel.ac.uk (Tracy Hall), Alper.Sen@boun.edu.tr (Alper Sen)

1. Introduction

The aim of this paper is to evaluate previously used evolutionary coupling (EC) measures from a measurement theory perspective and present recommendations for measuring EC in software artefacts. Evolutionary coupling[1] is the relationship between parts of software systems which are frequently changed together during the evolution of a system. Understanding the EC in software systems is important, as it has been shown to provide insight into architectural problems [1, 2, 3, 4, 5], cross-cutting concerns [6, 7, 8], software defects [9, 10, 11, 12, 13, 14] and the impact of change [15, 16].

Although there has been considerable work in establishing a measurement theory basis for software measurement [17, 18, 19, 20], it is not clear that these measurement principles have been used in EC measurement. Measures that do not adhere to these principles may be flawed. Empirical results obtained using flawed measures are likely to be unreliable and decisions and conclusions based on these results could be misleading. Such results may lead to time, money and resources being waste as well as contaminated scientific knowledge generated.

Evolutionary coupling is measured based on the co-changes of software artefacts in the version history. There are a variety of different **co-change** characteristics such as locality and change type. One flaw of current EC measures is treating these different characteristics as equivalent during the aggregation of co-change measurement results. For example; in terms of locality the distance between co-changes, specifically cross-system vs. within-system co-changes, there is evidence [11] that a cross-system evolutionary coupling *is not equal* to a within-system evolutionary coupling in the context of *defect prediction*. In particular that cross-system co-changes are more valuable to defect prediction. Unreliable results are possible where EC measurement does not take such characteristics into account, and the validity of these measures is difficult to establish.

In addition, there is no consistent definition of EC with studies using a variety of different measures. This is problematic as it means that the phenomenon that one study is reporting as EC may be very different to another study. Some studies [2, 12, 15] require a minimum number of evolutionary coupled artefacts while others [1, 16, 21] do not. This inconsistency is problematic because it is hard to compare the results of different studies.

In this paper we present EC measurement evaluation criteria. We use measurement theory and

[1]Evolutionary coupling is also known as change coupling or logical coupling.

metrology principles in the development of our criteria. Measurement must start with an objective [22, 23, 20, 18]. Objectives put measurement in a context and determine the design of measures and interpretation of the measurement results. So by following this basic principal of the measurement theory we first find the objectives of EC measurement. This constitutes our first research question (RQ1) in Table 1. Measurement captures information about attributes of entities [23, 20, 18]. So both the entity and the attribute to be measured should be identified clearly. This basic principle [23, 20, 18] is the base of our second research question (RQ2) in Table 1. We characterise the concept of evolutionary coupling by specifying the entities to be measured and the characteristics (attributes) that should be taken into account. We develop a meta-model of EC and its sub-concepts including the relationships across them as suggested by Abran [20]. In order to be valid, a measurement method must satisfy the representation condition of measurement theory [17, 23, 18]. Based on this principle and the meta-model developed in the previous step, we define the empirical relations and concerns to be considered in establishing a sound empirical relation system, which captures all generally accepted ideas about evolutionary coupling. We formulate our third question (RQ3) in Table 1 based on this principle of measurement theory. We check whether EC measures preserve the empirical relations while mapping the empirical relation system into the numerical one. Abran in his software metrology work [20] points out that mature engineering disciplines have well established measurement methods and the detailed procedures with a large international consensus. For example, the World Meteorological Organization (WMO) defines the temperature measurement setup in a very detailed and quantitative manner such as that thermometers should be positioned between 1.25 and 2m above the ground. Well-defined, standard and detailed measurement method and procedures are mandatory to ensure the accuracy, repeatability, and repetitiveness of measurement results. Based on this principle, we formulate our fourth research question (RQ4) to cover the detailed procedures and practicalities of EC measurement. Determining the scale type of measures and doing mathematical validation are also one of the basic practices of measurement. Scale types are very important to determine the limitations on the kind of mathematical manipulations that can be performed on measurement results. This forms our fifth research question (RQ5).

We evaluate current EC measures based on these research questions and criteria formulated for each research question. We reveal the picture of existing EC measurement practices and provide recommendations to be used in future uses of EC measurement.

We provide the following contributions in this paper. Firstly, we show the weaknesses and

Table 1: Research Questions

No	Research Question
RQ1	What are the objectives of EC measurement?
RQ2	Do existing EC studies identify entities and attributes to be measured?
RQ3	Do existing EC studies use sound empirical relation systems?
RQ4	Do existing EC studies define the measurement method and procedures?
RQ5	Do existing EC studies use scale type and mathematical validation?

strengths of current evolutionary coupling measures based on measurement theory principles. Secondly, we provide recommendations for practitioners and researchers about what/when EC measure to use and not to use. Furthermore, we develop a meta-model for the EC concepts, which are essential in understanding how the measure is derived and how to interpret the behaviour of the numerical values when returned to the real world. To the best of our knowledge, this is the first work that applies measurement theory [23, 18] and metrology [20] principles to EC measurement.

This paper is organised as follows: In the next section, we summarise related work. In Sections 3 and 4, we present our methodology for study selection and EC measurement evaluation, respectively. Section 5 shows the results of applying our methodology to the evolutionary coupling measures. The limitations of this study are addressed in Section 6 and the results are then discussed in Section 7. Finally, in Section 8, we summarise and present our conclusions.

2. Background

2.1. Evolutionary Coupling

Evolutionary coupling was first identified in 1997 by Ball et al. [1]. Early studies [1, 2, 3, 4] on EC focused on the relation between EC and architectural problems and EC was used as an indicator of architectural weaknesses and modularity problems. Classes that were frequently changed together during the evolution of a system were presented visually using EC information by Ball et al. [1]. Clusters of classes were identified according to EC measures. Ball et al. showed that classes belonging to the same cluster were semantically relevant. EC among different clusters was used as an indicator of ineffective class partitioning. Gall et al. analysed EC at a module-level and reported that EC provides useful insights into system architecture [2]. They identified potential structural shortcomings and detected modules and programs which should undergo restructuring or even reengineering. Another study by Gall et al. analysed evolutionary coupling at a class level on an industrial software system [3]. This study was important since it demonstrated that EC could be used to identify architectural weaknesses such as poorly designed interfaces and inheritance hierarchies. Pinzger et al. showed that candidate modules for refactoring could be detected by showing ECs between modules on Kiviat diagrams [4]. Beyer and Hassan explored EC data in the calculation of the distance between two files in a version control system (VCS) and displayed results as a series of animated panels [5]. They showed how the structure of a software system decayed or remained stable over time.

Besides detecting architectural problems, EC has also been used to predict possible co-changes and to recommend them to developers. In a study by Ying et al., an approach using data mining techniques was developed to recommend related source code parts to software developers assigned modification requests (MRs) [15]. Applying the approach to open source projects revealed important dependencies. A study by Zimmermann et al. presented a technique which predicted the parts of source code likely to change given the already changed parts of source code (at file, class, property, method levels) [16]. Association rule mining was used to detect evolutionary couplings.

EC has also proven useful in detecting the cross-cutting concerns scattered across systems. Breu et al. [6] leverage EC information to mine aspect candidates (identifying cross-cutting concerns). Eaddy et al. [7] argued that cross-cutting concerns are harder to implement and change consistently because multiple (possibly unrelated) locations in the code have to be found and updated simultaneously. Their study suggested that increased cross-cutting concerns may cause or even contribute to defects. Adams et al. [8] developed an aspect mining technique based on co-addition or co-removal of dependencies on program entities over time. They suggest that detailed knowledge about cross-cutting concerns in the source code is crucial for the cost-effective maintenance and successful evolution of large systems.

EC measures have also been used in defect prediction studies. Steff and Russo created sequential commit graphs of evolutionary coupled classes [9]. They showed that the graphs could be used for defect prediction. A study by Tantithamthavorn et al. proposed improvements to existing bug localisation methods by utilising EC information [10]. The proposed method was applied and verified on two open-source projects (Eclipse SWT and Android ZXing). Another study reported by

Kouroshfar concluded that EC between sub-systems was more effective on defects than EC within the sub-systems [11]. D'Ambros et al. [12] analysed three large software systems and detected correlation between EC and software defects, although no defect prediction was performed. This study considered only evolutionary coupling between classes within a project. Our own previous work [24, 25] suggests there is significant correlation between EC and defects in a commercial banking system.

No study has looked at the quality of EC measurement and the way in whether EC is measured inconsistently across studies.

2.2. Software Measurement and Metrology

Fenton and Bieman[23] provides information about measurement fundamentals, data collection and data analysis as well as the mathematical and statistical background of software measurement. Measurement is based on the following two concepts [23]:

Entity: An object (such as a person, a room or a program) or an event (such as a journey or the testing phase of a software project) in the real world.

Attribute: A characteristic or property of an entity (such as the area or color (of a room), the cost (of a journey) or the elapsed time (of the testing phase).

So measurement captures information about attributes of entities. Both the entity and the attribute to be measured should be specified for measurement, entities or attributes should not be used alone. Entities in software measurement can be divided into three categories:

Processes: Collections of software-related activities

Products: Any artifacts, deliverables or documents that result from a process activity

Resources: Entities required by a process activity

Within each category, there are two types of attributes:

Internal Attributes: Attributes that can be measured purely in terms of the product, process or resource itself.

External Attributes: Attributes (of an entity) that depend on the entity and its context. For example, reliability of a program depends on both the program and the environment in which the program is used.

Fenton and Bieman [23] differentiates these two kinds of measurement:

Direct (Base) Measurement: A direct quantification of an attribute of an entity such as the height or the weight of a person. No other attribute or entity is involved in the measurement.

Indirect (or Derived) Measurement: Direct measurement results are combined into a quantified item that reflects some the attribute of the entity to be measured. For example; density of a physical object can be measured only indirectly in terms of mass and volume. So base measurements of mass and volume attributes of an object are used to quantify density of the object.

Zuse [18] also contributed to the use of measurement theory in the area of software measurement. He extended classic measurement theory to the needs of software measurement [18]. Both Fenton and Zuse used the Representational Measurement Theory (RMT) approach. RMT [26] formalizes the intuitive, empirical knowledge about an attribute of a set of entities and the quantitative, numerical knowledge about the attribute. The intuitive knowledge is captured via the so-called empirical relational system and the quantitative knowledge via the so-called numerical relational system. Both the empirical and the numerical relational systems are built by means of set algebra. A measure links the empirical relational system with the numerical relational system in such a way that no inconsistencies are possible, as formalized by the Representation Condition. For example, intuition about the weight of an object is quantified by different measures in kilograms, grams, pounds, ounces, etc.

We also use the software metrology perspective proposed by Abran [20]. Abran [20] discusses the software measurement from a measurement method point of view. Metrology is the science of measurement and includes all theoretical and practical aspects of measurement according to the definition by the International Bureau of Weights and Measures (BIPM) [27]. His work is complementary to the previous work on the measurement theory. Abran suggests the following steps for designing a measurement method [20]:

1. Determination of the measurement objectives.

2. Characterization of the concept to be measured by specifying the entities to be measured and the characteristics (attributes) to take into account.
3. The setting up of the measurable construct, or of the meta - model, including the relationships across the concept to be measured (entity and attribute).
4. Definition of the numerical assignment rules.

Abran [20] indicates that not all software measures have strong designs and explains the quality criteria that should be expected from software measures. He provides what to look for when analysing a software measurement method. Cheikhi et al. [28] applies the metrology concepts to the Chidamber and Kemerer measures suite. They provide an investigation of the extent to which this set of measures addresses the metrology concepts related to the software measurement design. To help in understanding and clarifying the Chidamber and Kemerer measures, each measure is analysed from metrology perspectives and mapped to the relevant concepts. In this way it identifies the metrology concepts that are not tackled in Chidamber and Kemerer measures and helps to improve the design of these well-known measures.

3. Study Selection Methods

To identify the set of EC studies to evaluate we used a cut-down SLR approach based on that proposed by Kitchenham et al. [29]. We did not include all stages of an SLR, for example we did not include an explicit quality assessment, as the objectives of our study did not require this. We were aiming to identify all studies that use EC measurement, in order to evaluate that EC measurement. It could be argued that this evaluation is an extended quality check.

First we prepared a protocol for our literature review which specifies the methods that will be used to undertake a specific systematic review. Such a pre-defined protocol reduces the possibility of researcher bias and allows validation and replication. For example, without a protocol, it is possible that the selection of individual studies or the analysis may be driven by researcher expectations. We now summarise our protocol.

We identified previous studies of EC using the search facilities in the SCOPUS tool. SCOPUS is a general indexing system, which indexes most of the main internationally recognised software engineering journals that regularly publish empirical studies [29]. The following search string is used in our searches:

(evolutionary coupling OR change coupling OR logical coupling) AND (software) anywhere in study

We used these terms as our preliminary reading shows that three different terms are used to refer to EC: Evolutionary Coupling [16], Change Coupling [12] and Logical Coupling [2]. We added Software as a keyword to reduce false positives from other domains. Our initial searches returned many false positives from other domains such as Genetics, Molecular Biology, Electronics, etc. We follow Kitchenham et al.'s recommendation of using fairly simple search strings based on the main topic [29]. Simple strings are more likely to work on a variety of different digital libraries without extensive refinement.

Our searches covered 1997 to 2014. We chose 1997 as the start date since there is general agreement [16] [12] that EC is first introduced in 1997 by Ball et al. [1]. After obtaining the potentially relevant primary studies by digital library and manual searches, each must be assessed for relevance. The following inclusion criteria (a paper must be) and exclusion criteria (a paper must not be) were identified in the SLR protocol:

Inclusion criteria:

- A study measuring Evolutionary Coupling
- Published in English
- A Journal paper or Conference proceedings

Exclusion Criteria:

- Papers not subject to peer-review
- Grey literature (i.e. technical reports, work in progress)

The first author assessed each candidate paper by applying the inclusion/exclusion criteria on all primary studies found. Then a randomly selected three papers were also assessed by the two other senior researchers, plus one other researcher. Disagreements among assessors were resolved with a meeting. This process identified fifteen included papers.

4. EC Measurement Evaluation Methods

Our evaluation methods are based on the software measurement literature. From the literature we drawn on the principles of measurement theory [23, 18] and on software metrology [20] develop evaluation criteria that will allow us to answer our research questions. Our evaluation criteria are based on established software measurement theory (Fenton [23] and Zuse [18]) and, in particular, are based on representation conditions, the mathematical properties of the manipulation of numbers and the proper conditions for such manipulations. Our evaluation criteria also draw on software metrology [20] by including criteria which evaluate the measurement method.

The set of evaluation criteria that we develop are not of equal importance. Consequently, we weight the importance of each criteria using three gradations (3 or 2 or 1, from high to low). These weights have been arrived at by the authors interpreting the literature on software measurement and applying to the measurement of EC.

The following subsections detail the principles of measurement theory and software metrology that we have used in the development of our evaluation criteria within the context of our Research Questions.

4.1. Research Question 1: What are the objectives of EC measurement?

This research question was motivated by Abran [20] and Fenton and Bieman [23] emphasising the importance of determining measurement objectives. It is important to know the intended use of the measure and the intended users of the measurement results (software user, software designer, etc.). As a measure can be very useful for a specific measurement objective while the same measure may not be appropriate for another measurement objective. Consequently our first evaluation criteria is:

ID	Criteria	Category	Weight	RQ
C0	Objectives and Users of Measurement	Measurement Objectives	3	RQ1

This criteria is essential to all studies and so has a weighting of 3.

4.2. Research Question 2: Do existing EC studies identify entities and attributes to be measured?

This research question was included because to evaluate a base measure the concept to be measured must be first clearly defined as stated by Fenton ([23] and by Abran [20]). An (empirical) operational definition of the entity and the attribute must be given; that is, the concept must be characterised. This characterisation can be made by first stating explicitly how the concept is decomposed into sub-concepts as suggested by Abran [20]. This decomposition describes what role each sub-concept plays in the constitution of the concept to be measured and how these sub-concepts are themselves defined. We start the decomposition by analysing the definition of evolutionary coupling:

”The implicit relationship between two or more software artefacts which are frequently changed together.” [2, 3, 12]

The sub-concepts extracted from this definition are 1) software artefact, 2) relationship and 3) co-change (changed together). We further decompose these three concepts into their sub-concepts (following) and identify our detailed evaluation criteria.

EC sub-concept 1: Software artefact. We decompose this sub-concept into different types of software artefact: System, Module, Package, File, Class, Method. The granularity of software artefacts increases from System to Method level. Different EC studies in the literature use different types of software artefacts in their studies: System [2, 11], Module [2, 4, 11], Package [30], File [15, 5, 31, 21, 32, 24], Class [1, 3, 16, 6, 12] and Method [16, 6]. Studies show the importance of granularity in measuring EC. For example, [16] showed that information on the coupling between methods (low granularity) rather than classes or modules (higher granularity) is more useful for developers during impact analysis. Consequently, we add the following evaluation criterion:

Granularity of Software Artefact: The unit of code granularity of coupled items must be reported.

As granularity is fundamental to the interpretation of any results we assign a high weight (3) to this criterion, as below.

ID	Criteria	Category	Weight	RQ
C1	Granularity of the Entity	Identifying Entities and Attributes	3	RQ2

EC sub-concept 2: Co-Change

Next, we decompose the sub-concept **co-change** by considering various characteristics of co-change. These characteristics are based on the authors' existing knowledge of changing software and on a brain-storm of the issues between the authors. As these co-change characteristics are candidates for different base measurement as discussed later in the paper, we assign a high weight (3) to these criteria. The characteristics identified are:

1. **Grouping Characteristic:** EC measurement starts with the direct measurement of co-

Table 2: Criteria related with RQ2 and RQ3

ID	Criteria	Category	Weight	RQ
C2	Grouping Approach - Entity Level	Identifying Entities and Attributes	3	RQ2
C3	Locality Aspects	Sound Empirical Relation Systems	3	RQ3
C4	Change Scope	Sound Empirical Relation Systems	3	RQ3
C5	Change Type	Sound Empirical Relation Systems	3	RQ3
C6	Distinct Committers	Sound Empirical Relation Systems	3	RQ3
C7	Temporal Aspect-Regularity	Sound Empirical Relation Systems	3	RQ3
C8	Temporal Aspect-Recency	Sound Empirical Relation Systems	3	RQ3
C9	Change Size	Sound Empirical Relation Systems	3	RQ3

change of software artefact pairs. To identify pairs of software artefacts, it must be clear how these pairs are related as this will make a difference to the conclusion that can be drawn. Three different grouping approaches are common, pairs based on: Modification Requests (MR), Commit Transactions or Tasks. Using different grouping approaches can result in measuring different attributes of entities. Consequently we introduce a criteria requiring that the grouping approach should be reported (**C2** in Table 2). This criteria has been given a weight of 3.

2. **Locality Characteristics:** The distance between co-changes is important. Co-changes that include files from different subsystems have been shown to result in more bugs than co-changes that include files only from the same subsystem [11]. Alali et al. [33] also show that the distance among artefacts co-changed can increase the detection accuracy of EC, suggesting that most evolutionary couplings occur within the same sub-folder and remain localised. Consequently we include the locality of changes in our evaluation criteria (**C3** in Table 2).

3. **Change Characteristics:** The reason for a change may also affect EC. In particular whether a change is a bug-fix or enhancement [34]. The process involved and the teams can be very different for these different changes with different scope. Therefore, we distinguish the reason for a change in our evaluation criteria (**C4** in Table 2).

4. **Change Type Characteristics:** There are three basic types of changes that can be made: **addition**, **modification** and **deletion**. These different changes have been shown to impact on the results of studies. For example, [16] suggest that different types of changes enhance the ability to predict related or missing changes during development. Consequently our evaluation criteria distinguish between changes types (**C5** in Table 2).

5. **Committer Characteristics:** Madeyski et al. [35] found out that defect prediction models trained on a data set containing Number of Distinct Committers (NDC) were significantly better than the simple models without NDC. Distinct committers can also effect EC measures especially when they are used in defect prediction. Consequently our evaluation criteria include distinct committers (**C6** in Table 2).

6. **Temporal Characteristics:** The work of Alali et al. [33] highlights the temporal aspects in the detection of evolutionary couplings. Their results show that the age of evolutionary couplings can increase the detection accuracy of EC. They observed that there was a 40% chance that a pattern will occur again in a few days or less. Furthermore, it appears that the older a coupling is, the more rooted it is. Consequently our evaluation criteria include age of EC (**C7 and C8** in Table 2).

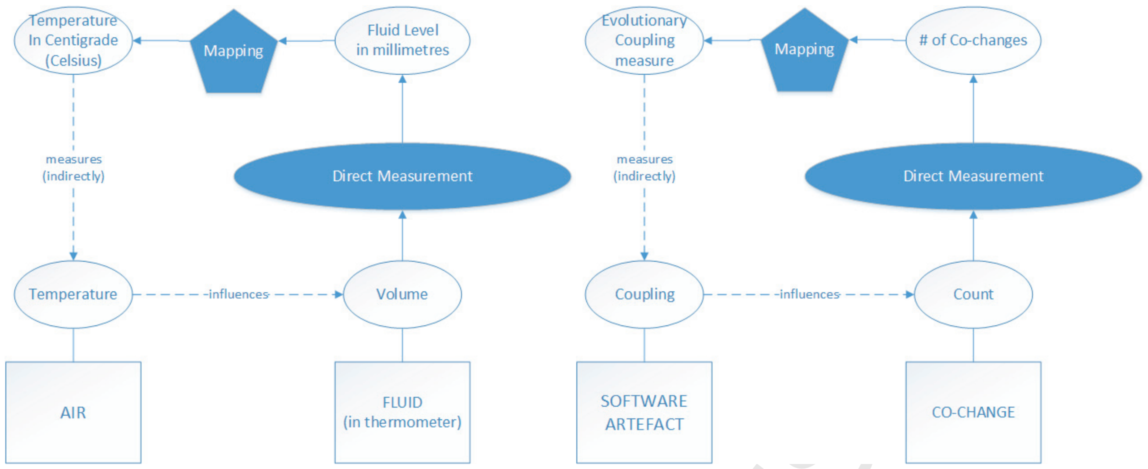
7. **Change Size Characteristics:** The size of changes involved in EC is likely to affect the impact of that EC. Large changes are not likely to have the same impact on EC as small changes. Consequently our evaluation criteria includes the size of changes (**C9** in Table 2).

EC sub-concept 3: Relationship

There are many different reasons for two artefacts to need changing at the same time, this means that there are a variety of reasons for EC such as structural, dynamic and semantic couplings. We suggest that evolutionary coupling is not a direct relationship between software artefacts, but it is a representation of other types of underlying direct relationships. This is very similar to temperature measurement as we generally do not measure air temperature directly. Instead it is measured via its effect on a fluid in a glass thermometer. The level of the fluid represents the air temperature as the strength of the evolutionary coupling represents the underlying direct couplings and relationships.

4.3. Research Question 3: Do existing studies use a sound empirical relation system

Establishing a Sound Empirical Relation System



(a) Temperature Measurement (b) Evolutionary Coupling Measurement

Figure 1: Temperature Analogy

In order to be valid, a measurement method must satisfy the representation condition of measurement theory [17, 23, 18]. The representational theory of measurement seeks to formalize our intuition about the way the world works. That is, the data we obtain as measures should represent attributes of the entities we observe, and the manipulation of the data should preserve empirical relationships that we observe among entities. A measurement mapping M must map entities into numbers and empirical relations into numerical relations.

Following Fenton's example we take the example of measuring the height of a person. In this specific case, person is the *entity* and height is an *attribute* of the entity, which is to be measured. Let's take two instances of the people entity as Tyrion Lannister (the dwarf in the HBO series Game of Thrones) and Shaquille O'Neal (nicknamed Shaq, a famous NBA basketball player). If we take *doesn't have same height* as the (binary) empirical relation to compare the heights of these two people, any measure aiming to measure the height attribute of a person entity must preserve the empirical relation as Tyrion does not have same height as Shaq. This empirical relation can be mapped to the numerical relation $\neq$. In other words, $M(Tyrion) \neq M(Shaq)$ must hold whenever a measure maps these people into numbers. If we use *metre* as a base measure from the International System of Units (SI), we observe that the empirical relation is preserved: $M(Tyrion) = 1.35 \neq M(Shaq) = 2.16$.

To evaluate and compare EC measures from the empirical relation system point of view (which

Table 3: Empirical Relations and Attributes applied on

Empirical Relation	**Attributes applied**	**Corresponding Mathematical Operation**	**Nominal**	**Ordinal**	**Interval**	**Ratio**	**Absolute**
is not equal in strength	Coupling Strength	$\neq$	✓	✓	✓	✓	✓

is addressed by our third research question, RQ3), we defined a basic empirical relation and its corresponding mathematical operations in Table 3. The *not equal in strength* empirical relation covers all scale types as shown in Table 3. So all existing EC measures can be covered by using this empirical relation. Users of our criteria can decide using stronger relationships such as greater than based on their specific measurement objectives and context. A condition holding for defect prediction may not be true for aspect detection. Or temporal aspect - recency can be interpreted quiet differently depending on the measurement objectives. Therefore we chose to use the *not equal in strength* to cover all EC possible measures and measurement objectives. In the context of EC measurement, **software artefact** is the *entity* and **coupling strength** is the *attribute* of the entity, which is to be measured. EC measures are indirect measures as first **co-change** entity is measured and aggregated onto **software artefact** entities. Therefore we need to focus on the direct measurement first: **co-change measurement**. As we detail previously, there are different characteristics of the **co-change** entity. A very important theoretical flaw of current EC measures is treating these different characteristics as equivalent during aggregating co-change measurement results and associating these with a software artefact. Any valid EC measure to be used should preserve the empirical relation. Based on the characteristics we previously describe for co-change, we similarly define the following empirical system evaluation criteria as provided in Table 2:

- **Locality Aspect:** cross-system evolutionary coupling *is not equal in strength* to within-system evolutionary coupling (**C3** in Table 2). As discussed previously, different types of coupling may contribute differently to the overall coupling measurement result and so EC measurement should consider cross-module, cross-package or cross-application couplings.
- **Change Scope:** evolutionary coupling resulting from a Bug-fix *is not equal in strength*

to evolutionary coupling resulting from an Enhancement (**C4** in Table 2). As discussed previously, the relationships among files coupled through a Bug-fix can be quite different from the relationships among files coupled through an Enhancement.

- **Change Type:** evolutionary coupling resulting from additions/deletions *is not equal in strength* to evolutionary coupling resulting from modifications (**C5** in Table 2). Treating addition, deletion, and modification as equivalent can be problematic in the same way that the CBO measure has also been said to be problematic (CBO incorrectly treats forward and backward links as equivalent [36, 28]). The relationships amongst files coupled through additions can be quite different from the relationships among files coupled through modifications.

- **Distinct Committers:** evolutionary coupling resulting from multiple distinct developers *is not equal in strength* to evolutionary coupling resulting from a single developer (**C6** in Table 2). Co-changed artefacts can be changed by different committers. There are various previous studies suggesting that the number of committers impacts on the results of analysing software. For example, Madeyski et al. [35] found out that the defect prediction models trained on a data containing Number of Distinct Committers (NDC) were significantly better than the simple models without NDC. Distinct committers are also likely to affect EC measures especially when they are used in defect prediction.

- **Temporal Aspect-Regularity:** evolutionary coupling resulting from regular changes *is not equal in strength* to evolutionary coupling resulting from non-regular changes (**C7** in Table 2). The patterns of co-changes can be diverse such as short-lived, long-lived regular, and long-lived non-regular. Mondal et al. [37] showed that couplings based on regular co-changes can be more useful in some contexts such as impact analysis or change prediction.

- **Temporal Aspect-Recency:** evolutionary coupling resulting from recent changes *is not equal in strength* to evolutionary coupling resulting from non-recent changes (**C8** in Table 2). For projects that are frequently restructured, assigning a higher weight to recent changes can increase precision and recall for change prediction [16]. Recent changes are more likely to be relevant than older changes as they reflect the current relationships between software artefacts.

- **Change Size:** evolutionary coupling resulting from larger change sizes *is not equal in strength* to evolutionary coupling resulting from smaller change sizes (**C9** in Table 2). As discussed previously, the size of changes can be very different. For example, it can be a few LOC, hundreds of LOC or thousands of LOC for a file. The size of the change constitutes another concern that can affect EC measurement, as the size of changes on co-changed software artefacts can contribute to the strength of the coupling between software artefacts.

Which of these criteria are are most important to consider will depend on the particular measurement objectives and the users of those measures. However it is important that each criteria and possibly its weighting is explicitly considered.

Mapping from Empirical Relation System to Numerical

Measurement is formally defined as a mapping from the empirical world to the numerical world. Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterise an attribute. Thus, the real world is the domain of the mapping, and the mathematical world is the range. A measure must specify the domain and range as well as the rule for performing the mapping.

Representation Condition: The behaviour of the measures in the numerical world should be the same as the corresponding elements in the real world, so that by studying the numbers we learn about the real world. Thus, we want the mapping to preserve the relation. This rule is called the *representation condition* [23]. A measurement mapping M must map entities into numbers and empirical relations into numerical relations.

Table 5 provides several examples for the *not equal in strength* empirical relation. We define at least one case for each criteria listed in Table 2 to be able to check that numerical relations preserve empirical relations. One counter-example is enough to show that the representation condition is not satisfied for a given measure. Let us explain the first example in Table 5. There are two commit transactions: Commit1 and Commit2. Two software artefacts (File1 and File2) have been changed in Commit1. File2 and File3 have been changed in Commit2. If we consider only this information to measure the evolutionary coupling between files, the EC between File1 and File2 (expressed as M(File1-File2)) will be same as the EC between File2 and File3 (expressed as M(File2-File3)). However, if we consider that File1 and File2 reside in System 1 while File3 resides in System 2, the EC between File2 and File3 as a cross-system coupling will be intuitively stronger compared to the EC between File1 and File2.

No	***Cases for not equal in strength Empirical Relation (L)***	***Validation Sample (Example Scenario)***	***Must Hold in Measurement Objective scope***
1	*Coupling through co-changes of two software artefacts residing on the same system /package/subsystem is not equal in strength to the one through co-changes from different system /package/subsystem* ***(Locality Aspect, Criterion 3)***	*Commit1: File1, File2* *Commit2: File2, File3* *(File1 and File2 reside in System 1)* *(File3 resides in System 2)*	$M(File1 - File2) \neq M(File2 - File3)$
2	*Being changed by multiple developers constitutes not equal coupling in strength compared to being changed by a single developer* ***(Distinct Committers, Criterion 6)***	*Commit1: File1, File2 (Developer 1)* *Commit2: File1, File2 (Developer 1)* *Commit3: File4, File5 (Developer 1)* *Commit4: File4, File5 (Developer 2)*	$M(File1 - File2) \neq M(File4 - File5)$
3	*Being changed regularly yields not equal coupling in strength to being changed non-regularly* ***(Temporal Aspect-Regularity, Criterion 7)***	*Commit1: File1, File2 (07/2014)* *Commit2: File1, File2 (07/2014)* *Commit3: File1, File2 (07/2014)* *Commit4: File4, File5 (01/2014)* *Commit5: File4, File5 (04/2014)* *Commit6: File4, File5 (07/2014)*	$M(File1 - File2) \neq M(File4 - File5)$

Table 5: Examples for Empirical Relation System Evaluation

4.4. Research Question 4: Do existing studies define measurement method and procedures?

In software metrology, the measurement method is generally defined as a logical organisation of operations used in a measurement [20]. A specific measurement method has to be designed to obtain a base measure for a specific attribute. We summarise the EC measurement process based on the ISO 15939 measurement information model [38]. The measurement method links the attribute of an entity to a base measure. Function Point (FP) and COSMIC-FFP are two examples of measurement methods in software engineering used to measure the functional size of software. EC is not a base measure and not measured directly. **Co-change measurement** is the direct measurement and co-change measures are base measures. Then, the values of two or more base measures are used to derive EC measures by means of a measurement function. Therefore EC is a derived measure. EC as a derived measure is then used in the context of an analysis model to explain the relationship between EC and the information needed [38].

Measurement methods should be complemented with detailed measurement procedures, which are sets of operations, described specifically, used in the performance of particular measurements according to a given method [20]. When measurement method and procedures are not sufficiently well defined and standardised to ensure the accuracy, repeatability, and repetitiveness of measurement results, then, when the same entity (software) is measured by different measurers, the results can potentially be significantly different. By using the adapted ISO 15939 measurement information model and our own experience in measuring EC [24, 25] on large software we identified the following evaluation criteria as provided in Table 6:

- **Normalisation:** EC measures should be normalised for comparison with different studies (**C10** in Table 6). Normalisation has a considerable impact on software measures [39, 10]. In the scope of EC measurement, product size or total number of couplings are potential candidates for normalisation of EC measurement results. A measure which is not normalised has little value out of the measurement context. EC measures hold this risk as well. Furthermore, size of the application can implicitly and unintentionally be included in EC measure if normalisation is not considered.

- **Grouping Approach - Temporal (Entity Level: Software Artefact Pair):** The period of time from which version history is used for measurement should be considered (whole life of an application or part of it such as only the maintenance phase, specific release, etc.)

(**C11** in Table 6). The commit history from version control systems (VCS) are essential for calculating EC measures. It should be specified which part of the commit history is or should be considered. Some EC measures need the version control history from a long period to yield meaningful results [16]. This type of measure will not be applicable especially in the initial development phases.

- **Large Merge Transactions (Measurement Method):** Large merge transactions should be ignored in EC measurement as large merge transactions generally contain co-changes which aren't relevant. We may end up with EC values which do not accurately reflect the underlying coupling among software artefacts (**C12** in Table 6). This is because the evolution of a product sometimes branches into different evolution strands, which may later be merged again. The merge from a branch to trunk becomes a large transaction which includes all the changes made in the branch. Large merge transactions do not reflect the fine-grained relations among files. To detect evolutionary coupling within transactions, large merge transactions should be avoided. For example, some new features implemented on a branch can be then moved (merged) all together to the trunk and all the files changed in the scope of multiple features will be incorrectly interpreted as coupled in this merge transaction. On the other hand, large transactions can be useful in some other contexts such as mining cross-cutting concerns (aspect mining) [6].

- **Branches - Aggregation of Base Measures (Measurement Function):** Commit transactions for the same bug-fix/enhancement on different branches should not be considered multiple times (**C13** in Table 6). If different versions of a product are alive and used by its customers, bug-fixes/enhancements should generally be addressed in multiple branches. Especially bug-fixes should be done on all live branches. The same set of files changed in the scope of the same bug-fix/enhancement should not be counted multiple times for different branches when calculating an EC measure. Otherwise the strength of EC among these files may be misleadingly measured higher than the actual strength.

- **Aggregation of Base Measures (Measurement Function):** Being changed in the scope of multiple MRs / Transactions should be considered (**C14** in Table 6). Some software artefacts can be coupled through multiple MRs / Transactions. The number of different MRs / Transactions can be involved in the measurement of EC. In some measurement contexts, the number of different MRs / Transactions involved can contribute to EC measurement.

- **Single Artefact Commits - Aggregation of Base Measures (Measurement Function):** Are commit transactions involving only single software artefact considered? (**C15** in Table 6). In the scope of some commits, only one software artefact is committed. When calculating EC measures, it should be indicated whether this is considered or not. Previous studies show that about one third of all commits have only one file involved. With increased granularity, this ratio will increase even more. Involvement of single artefact commits is critical for the calculation of some EC measures such as Confidence [16].

- **Refactorings - Software Artefacts (Measurement Method):** Refactorings should be considered in the measurement (**C16** in Table 6). Refactorings such as name changes, moving artefacts to different folders, packages, modules, etc. should be considered in EC measurement. It is very important especially for projects that are frequently restructured [16].

- **Prerequisite: Adequateness of Version Control (VC) History:** The version history should be good and rich enough to have meaningful measures. All software artefacts should be involved in at least one commit transaction / MR / Task (**C17** in Table 6). EC is an indirect measure and representation of the coupling attribute of a software artefact. In most of its uses (analysis model [38]) it is used to represent the coupling and complexity of a software artefact. However, EC being a process measure we need to have enough version control (VC) activity in order to represent coupling, which is a product measure. If we have no VC activity for a particular software artefact, it might be wrong to conclude that it is coupled with no other software artefacts. Therefore checking that all software artefacts are involved in at least one commit transaction, MR or Task as this will mean that each software artefact will reveal its couplings.

- **Relative Strength (Measurement Function):** All the transactions in the system should be considered in the EC measurement of each software artefact (**C18** in Table 6). Commit transaction density and co-change sizes can be different for different systems. It can be hard to interpret the measurement results. Considering all the transactions in the calculation of individual EC calculations of each software artefact can provide measurement results which can be comparable across studies and projects. Considering all transactions in a system

Table 6: Criteria related with RQ4

ID	Criteria	Category	Weight	RQ
C10	Normalisation	Measurement Method and Procedure	1	RQ4
C11	Grouping Approach - Temporal	Measurement Method and Procedure	1	RQ4
C12	Large Merge Transactions	Measurement Method and Procedure	1	RQ4
C13	Branches	Measurement Method and Procedure	1	RQ4
C14	Aggregation of Base Measures	Measurement Method and Procedure	1	RQ4
C15	Single Artefact Commits - Aggregation of Base Measures	Measurement Method and Procedure	1	RQ4
C16	Refactorings of Software Artefacts	Measurement Method and Procedure	1	RQ4
C17	Adequateness of Version Control (VC) History	Measurement Method and Procedure	3	RQ4
C18	Relative Strength (Measurement Function)	Measurement Method and Procedure	1	RQ4

can provide a better relative coupling strength as Beyer et al. [40] [5] proposed in their visualisation work. It can also be used for normalisation purposes.

These evaluation criteria in Table 6 are part of our overall EC evaluation criteria and are used to answer our fourth research question (RQ4). The impact of these criteria on EC is not high as the criteria identified under other categories earlier. So we assign a lower weight (1) to these criteria. The only exception is C17. This is a prerequisite and it is assigned a high weight (3).

4.5. Research Question 5: Do existing studies do scale type and mathematical validation?

As stated by Zuse [18] and Fenton [23], scale types are very important to determine the limitations on the kind of mathematical manipulations that can be performed on measurement results. For example, it is invalid to compute multiplication, division and ratios with nominal, ordinal and interval scales. The analysis is constrained by the scale type. Only those calculations that are permissible for the given scale should be performed, reflecting the type of attribute and mapping that generated the data. The table in the online appendix [41] provides a complete list of the scale types and permissible operations for each. In this study, we evaluate the scale types of EC and its sub-components. Furthermore, we evaluate the design of the measurement methods in terms of its scale types. When the measurement method includes a design that incorrectly mixes the scale types of its sub-components, numbers are produced, but such numbers are meaningless and their use in analysis models can only be meaningless. That is, they lead to meaningless models, which in turn provide users of the measurement with a false sense of well-being that is, they increase risks, rather than reducing them. For example, Zuse [18] has identified three concurrent scale types for Cyclomatic Complexity: ordinal, ratio and absolute. Typically, a measurement method should involve a single scale type.

We add a last evaluation criterion (**Criterion 19**) to consider the mathematical validation as part of research question five (RQ5). As this principle is very important for the soundness of a EC measure, we have assigned the highest weight which is 3.

ID	Criteria	Category	Weight	RQ
C19	Scale Type and Mathematical Validation	Scale Type and Mathematical Validation	3	RQ5

4.6. Applying the Evaluation Criteria

Four researchers were involved in the assessment process. One researcher (the first author) evaluated all papers against our evaluation criteria. Then three senior researchers (two of which were co-authors of this paper) also evaluated a randomly selected set of three papers independently to validate the assessments made by the first author. Disagreements among assessors were resolved with a meeting and the evaluation criteria clarified so that such disagreements could be avoided in the future. We have followed the following steps to randomize the papers. Papers have been assigned numbers according to the their order in the table. Then these numbers (from 1 to 15) have been written on small pieces of papers. Three numbers have been randomly chosen from these papers.

5. Results

We evaluated existing EC measures according to the criteria provided in the previous section. The following subsections provide evaluation results for each RQ separately.

Table 7: Objectives and Users of EC Measurement (RQ1)

Measurement Objective	**Intended Users**	**Studies**
Defect Prediction	Managers/Team Leaders	Gall et al. [2] [3], Beyer and Hassan [5], D'Ambros et al. [12], Cataldo et al. [21], Steff and Russo [9], Kouroshfar [11], Kirbas et al. [24, 25]
Impact Analysis	Analysts/Testers	Pirklbauer [42]
Change Prediction	Developers	Ying et al. [15], Zimmermann et al. [16], Kagdi et al. [31] [43], Zou et al. [30]
Bug Localisation	Developers	Tantithamthavorn et al. [10]
Visualisation	Architects/Managers/ Team Leaders	Ball et al. [1], Pinzger et al. [4], Beyer and Hassan [5], Steff and Russo [9]
Cross-cutting concern (Aspect) Detection	Developers	Breu and Zimmermann [6], Eaddy et al. [7], Adams et al. [8]

5.1. RQ1: What are the objectives of the EC measurement?

Table 7 summarises our findings on the objectives and users of EC measures of published studies. We have identified six different objectives in total. The most common objectives are defect prediction and change prediction. The less common objective is impact analysis. The intended users of EC measures are diverse covering managers, team leaders, developers, analysts, testers and architects.

5.2. RQ2: Do existing EC studies identify entities and attributes to be measured?

Table 9 summarises the evaluation results of criteria one and two, which are related to RQ2. All EC measures satisfy both criteria. So the answer to this research question is a clear Yes.

5.3. RQ3: Do existing EC studies use sound empirical relation systems?

Table 11 summarises the evaluation results of criteria from three to nine, which are related to RQ3. The last column of the table shows the total number of studies satisfying a criterion at each row. Each criterion is considered only by few studies (Max: 6, Min: 0 out of 15 studies). EC studies are relatively successful in addressing the criterion C8 (Temporal Aspect - Recency) which have been covered by 6 studies. No study satisfies all seven criteria related to the RQ3. Furthermore, no study provides an implicit definition of the empirical relation system and empirical relations used in their EC measure design. These results suggest that EC measures have problems with establishing a sound empirical relation system.

Criteria	**Ball et al. [1]**	**Gall et al. [2]**	**Gall et al. [3]**	**Ying et al. [15]**	**Zimmermann et al. [16]**	**Pinzger et al. [4]**	**Breu et al. [6]**	**Beyer et al. [5, 40]**	**Kagdi et al. [31]**	**Kagdi et al. [43]**	**Zou et al. [30]**	**D'Ambros et al. [12]**	**Cataldo et al. [21]**	**Kouroshfar [11]**	**Mondal et al. [37]**	**SUM**
C1.Granularity of the Entity	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**15/15**
C2.Grouping Approach - Entity Level	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	**15/15**
SCORE	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	**6/6**	

Table 9: EC Measurement Evaluation Results for Research Question 2 (RQ2)

Criteria	**Ball et al. [1]**	**Gall et al. [2]**	**Gall et al. [3]**	**Ying et al. [15]**	**Zimmermann et al. [16]**	**Pinzger et al. [4]**	**Breu et al. [6]**	**Beyer et al. [5, 40]**	**Kagdi et al. [31]**	**Kagdi et al. [43]**	**Zou et al. [30]**	**D'Ambros et al. [12]**	**Cataldo et al. [21]**	**Kouroshfar [11]**	**Mondal et al. [37]**	**SUM**
C3. Locality Aspects	X	✓	✓	X	X	✓	X	X	X	X	X	X	X	✓	X	**4/15**
C4. Change Scope	X	✓	✓	X	X	✓	X	X	X	X	X	X	X	X	X	**3/15**
C5. Change Type	X	X	X	X	✓	X	✓	X	X	X	✓	X	X	X	X	**3/15**

C6. Distinct Committers	X	X	X	X	X	X	✓	X	X	X	✓	X	X	X	X	**2/15**
C7. Temporal Aspect - Regularity	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0/15**
C8. Temporal Aspect - Recency	X	X	X	X	✓	X	✓	X	✓	✓	✓	✓	X	X	X	**6/15**
C9. Change Size	X	X	X	X	X	X	✓	X	X	X	X	X	X	X	X	**1/15**
SCORE	**0/21**	**6/21**	**6/21**	**0/21**	**6/21**	**6/21**	**12/21**	**0/21**	**3/21**	**3/21**	**9/21**	**3/21**	**0/21**	**3/21**	**0/21**	

Table 11: EC Measurement Evaluation Results for Research Question 3 (RQ3)

5.4. RQ4: Do existing EC studies define measurement method and procedures?

Table 13 summarises the evaluation results of criteria from C3 to C9, which are related to RQ4. Each criterion is considered only by few studies (Max: 9, Min: 0 out of 15 studies). EC studies are relatively successful in addressing the criterion C11 (Grouping Approach - Temporal) which have been covered by 9 studies. No study satisfies all nine criteria related to the RQ4. Furthermore, no study provides a detailed measurement procedure for EC measurement. These evaluation results suggest that measurement method and procedures for EC measurement are not sufficiently well defined and standardized.

Criteria	**Ball et al. [1]**	**Gall et al. [2]**	**Gall et al. [3]**	**Ying et al. [15]**	**Zimmermann et al. [16]**	**Pinzger et al. [4]**	**Breu et al. [6]**	**Beyer et al. [5, 40]**	**Kagdi et al. [31]**	**Kagdi et al. [43]**	**Zou et al. [30]**	**D'Ambros et al. [12]**	**Cataldo et al. [21]**	**Kouroshfar [11]**	**Mondal et al. [37]**	**SUM**
C10. Normalisation	✓	X	X	X	X	X	X	✓	X	X	✓	X	✓	X	✓	**5/15**
C11. Grouping Approach - Temporal	X	✓	✓	✓	✓	✓	✓	X	X	X	✓	X	✓	✓	X	**9/15**
C12. Large Merge Transactions	X	X	X	✓	✓	X	X	X	✓	✓	X	✓	X	X	X	**5/15**
C13. Branches	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	X	**1/15**

C14. Aggregation of Base Measures	X	X	X	X	X	X	✓	✓	X	X	X	✓	✓	X	X	**4/15**
C15. Single Artefact Commits	X	X	X	X	X	X	X	✓	X	X	✓	X	X	X	X	**2/15**
C16. Refactorings of Software Artefacts	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0/15**
C17. Adequateness of VC History	X	X	X	✓	X	X	X	X	X	X	X	✓	X	X	X	**2/15**
C18. Relative Strength	X	X	X	X	X	X	X	✓	X	X	X	X	X	X	X	**1/15**
SCORE	**1/11**	**1/11**	**1/11**	**5/11**	**3/11**	**1/11**	**2/11**	**4/11**	**1/11**	**1/11**	**3/11**	**5/11**	**3/11**	**1/11**	**1/11**	

Table 13: EC Measurement Evaluation Results for Research Question 4 (RQ4)

5.5. RQ5: Do existing EC studies do scale type and mathematical validation?

Another problematic area is mathematical validation (criterion 19). No EC measure published performed an explicit mathematical validation (Table 15). Furthermore, all existing EC measures applying summation to different potential base units in their EC calculations such as cross-system and within-system co-changes.

Criteria	**Ball et al. [1]**	**Gall et al. [2]**	**Gall et al. [3]**	**Ying et al. [15]**	**Zimmermann et al. [16]**	**Pinzger et al. [4]**	**Breu et al. [6]**	**Beyer et al. [5, 40]**	**Kagdi et al. [31]**	**Kagdi et al. [43]**	**Zou et al. [30]**	**D'Ambros et al. [12]**	**Cataldo et al. [21]**	**Kouroshfar [11]**	**Mondal et al. [37]**	**SUM**
C19. Scale Type and Math. Validation	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	**0/15**
SCORE	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	**0/3**	

Table 15: EC Measurement Evaluation Results for Research Question 5 (RQ5)

6. Limitations of the Study

The validity of our work is mainly related with the appropriateness of inferences made on the basis of our assessment, specifically whether our assessment can measure how well EC measures are currently measured. The main threats to validity arise from the subjective nature of the criteria identification process. The criteria used are based on our own experiences and those reported in the literature. To address this threat, we used some SLR practices [29] (a cut-down SLR approach) to identify a comprehensive set of studies proposing evolutionary coupling measures, which helps to identify a comprehensive set of criteria. Nevertheless, there might be some other criteria which could not be identified by this study. We hope that this will provide the foundations for further study of the criteria expected from EC measurement.

Another threat to the validity of this study is that the studies used for formulating the Criterion 18 have been themselves assessed against this criteria. Although this is not ideal, all other criteria are driven by measurement theory and metrology principles. The general measurement principles guiding the formulation of the criteria are general. There can be also some overlap between these broad categories defined in the paper such as numerical assignment rules and measurement method as observed. However, most of the criteria like grouping approach (C2), locality aspects (C3), temporal aspect-regularity (C7), large merge transactions (C12), adequateness of VC history (C17), etc. are reflecting the EC measurement specifics. We used these general categories to guide our criteria formulation, brain storming and to help covering all aspects of measurement. In our criteria formulation we use measurement method & procedure category to capture the practicalities of EC measurement and whether EC measurement is well defined and standardised to ensure the accuracy, repeatability, and repetitiveness of measurement results. On the other hand, numerical assignment rules are utilised in the scope of ”sound empirical relation system” category to create concrete cases for the criteria to check whether numerical relations preserve empirical relations. Although there is overlap, the focus is different and they uncover different aspects of EC measurement.

Another threat is potential bias in the evaluation process. However we employed a validation exercise with three senior researchers to mitigate any potential bias.

External validity relates to the generalisation of our study results. We evaluated a number of EC measures. These may not be exhaustive and new EC measures might be proposed in the future.

7. Discussion

Our EC measure evaluations suggest that EC is currently not measured well. The particular weakness revealed by our research questions provide important information about the quality of existing EC measures and potential areas to improve. The EC measures of Breu & Zimmermann [6] and Zou et al. [30] are particularly worth to highlight as they have the highest overall scores. Our evaluations suggest that there is a need for new EC measures, which more clearly address the criteria we present. Furthermore, the criteria we presented can be used in the future to benchmark any newly developed measures of EC. We now discuss the answers to each of our research questions:

7.1. RQ1: What are the objectives of the EC measurement?

Our results show that EC measures are used for various purposes in software engineering research. We found six different objectives for EC measurement: Defect prediction, impact analysis, change prediction, bug localisation, visualisation and aspect detection. All of these areas of research could be improved in the future if EC measures were developed that satisfy our evaluation criteria.

7.2. RQ2: Do existing EC studies identify entities and attributes to be measured?

Our results reveal that all existing EC studies identify entities and attributes to be measured. Both criteria C1 and C2 of RQ2 are the only criteria satisfied by all existing studies. Existing studies do very well in this area. The level of granularity (C1) used by the studies vary and include files, classes and methods. The main grouping approach (C2) is VCS commits. This result provides a promising basis for improving other aspects of EC measurement.

7.3. RQ3: Do existing EC studies use sound empirical relation systems?

Breu & Zimmermann [6] and Zou et al. [30] satisfy the highest score, 12 and 9 respectively, in relation to this research question. The measurement objectives of these studies are aspect detection and change prediction respectively. Unfortunately some RQ3 criteria were satisfied by very few studies. For example the criterion 6 (distinct committers) is satisfied by only Breu & Zimmermann [6] and Zou et al. [30] and the criterion 9 (co-change size) is satisfied by only Breu & Zimmermann [6].

Overall our results suggest that existing EC measures need improvement. Evaluation criteria related to the empirical relation system (i.e. change locality, change scope, temporal, change size and change type) potentially lead to studies using different base units and so are important to the reliable measurement of EC. We suggest that practitioners and researchers measure each aspect identified in our evaluation criteria separately and use each in their analysis models as separate parameters. None of the studies used satisfied criterion C7. This suggests that the regularity of co-changes is not addressed in any of the EC measures that we assessed. Again, this may mean that the measures currently used are undermined.

7.4. RQ4: Do existing EC studies define measurement method and procedures?

D'Ambros et al. [12] and Ying et al. [15] satisifed the most criteria for their methods and procedures. In addition some criteria are satisfied by very few studies. For example, the criterion 13 (branches) is only covered by Zimmermann et al. [16]; the criterion 15 (single artefact commits) is only covered by Beyer & Hassan [5] and Zou et al. [30]; the criterion 17 (adequateness of version history history) is only covered by Ying et al. [15] and D'Ambros et al. [12]; the criterion 18 (relative strength) is only covered by Beyer et al. [40] [5]. However none of the studies satisfied the criterion 16 in EC measurement, which means that no study considered refactorings within or across projects is a potential candidate for EC measurement.

The studies we evaluate highlight many areas of improvement are needed. Overall the methods and procedures used to measure EC tend not to be defined well. Consequently measurement results of the same entity by different measurers can potentially be significantly different. This means that the usability and repeatability of existing EC measures is not strong which will make it difficult to replicate EC studies. It is essential that detailed measurement procedures should be described for EC measurement.

7.5. RQ5: Do existing EC studies do scale type and mathematical validation?

Our results suggest that this area is not addressed well by current EC measures. None of the existing studies declare the scale type of their EC measures or attempt to validate the measures mathematically. EC data is generally not normally distributed as reported by several studies [2] [12] [24, 25]. This limits the statistical methods that can be used on EC measures. For example, Pearson correlation will not apply to EC measures as well as other parametric methods like ANOVA (Analysis of Variance). Nonparametric methods such as Spearman and Kendall correlations should be preferred over parametric methods such as Pearson correlation.

Furthermore, aggregation of different types of co-changes, especially RQ3 related criteria such as C3 (within vs. cross system) and C5 (deletion/creation vs. modification), may be also problematic as these can be potentially interpreted as different base units in their EC calculations. We suggest measuring different type of co-changes separately and aggregating them within these different types when calculating EC.

Overall our results suggest that the quality of measurement used in EC needs significant improvement. Even the studies who score highest in our evaluations miss important criteria. On the other hand, the criteria related to RQ1 (providing the objectives of measurement) and RQ2 (identifying entities and attributes) are satisfied by all studies. So the results we present do show that there is a base of good measurement practices from which EC measurement could be strengthened.

8. Conclusions

In this paper, we evaluate EC measurement from a measurement theory perspective. We combine the software measurement ideas of Fenton, Zuse and Abran to develop 19 EC measurement evaluation criteria. We use these criteria to evaluate the current published studies on EC measurement. Despite the importance of EC and the published studies of EC, our results suggest that many basic principles of good software measurement have not been used in the measurement of EC. Scale type and mathematical validation stands out an area not addressed by EC studies. None of the existing EC studies declare their scale type nor attempt to validate their measures mathematically. Our evaluation results suggest that establishing a sound empirical relation system is also a problematic area. Existing studies fail to address the different types of co-changes such as locality (within vs. cross system) and change type (deletion/creation/modification). We also found that EC measurement methods and procedures are not well reported by existing studies. In addition no EC study presents measures that account for the temporal aspects of EC in terms of regularity or the impact of refactorings on EC. Overall our results suggest that there is much work to be done to put EC measurement on a firm footing that will enable the reliable measurement of EC and the accurate replication of EC measurement.

To the best of our knowledge, this is the first work that applies measurement theory and metrology principles to EC measurement. Although our results suggest significant weakness in the way that EC has previously been measured, it is highly likely that measurement in other areas of software engineering also requires significant improvement. Indeed Cheikhi et al. [28] show such measurement weakness in their analysis of OO metrics. We believe that the results and the criteria developed in this study will guide future research in EC to improve how EC is measured. Indeed the criteria presented in this paper can be adapted and used in any EC related study to ensure measurement quality.

References

[1] T. Ball, J.-M. Kim, A. A. Porter, H. P. Siy, If your version control system could talk, in: ICSE Workshop on Process Modelling and Empirical Studies of Software Engineering, 1997.

[2] H. Gall, K. Hajek, M. Jazayeri, Detection of logical coupling based on product release history, in: Software Maintenance, 1998. Proceedings., International Conference on, IEEE, Bethesda, MD, USA, 1998, pp. 190–198.

[3] H. Gall, M. Jazayeri, J. Krajewski, Cvs release history data for detecting logical couplings, in: Software Evolution, 2003. Proceedings. Sixth International Workshop on Principles of, IEEE, 2003, pp. 13–23.

[4] M. Pinzger, H. Gall, M. Fischer, M. Lanza, Visualizing multiple evolution metrics, in: Proceedings of the 2005 ACM symposium on Software visualization, ACM, New York, NY, USA, 2005, pp. 67–75.

[5] D. Beyer, A. E. Hassan, Animated visualization of software history using evolution storyboards, in: Reverse Engineering, 2006. WCRE'06. 13th Working Conference on, IEEE, Benevento, Italy, 2006, pp. 199–210.

[6] S. Breu, T. Zimmermann, Mining aspects from version history, in: Automated Software Engineering, 2006. ASE'06. 21st IEEE/ACM International Conference on, IEEE, Tokyo, Japan, 2006, pp. 221–230.

[7] M. Eaddy, T. Zimmermann, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, A. V. Aho, Do crosscutting concerns cause defects?, Software Engineering, IEEE Transactions on 34 (4) (2008) 497–515.

[8] B. Adams, Z. M. Jiang, A. E. Hassan, Identifying crosscutting concerns using historical code changes, in: Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE '10, ACM, 2010, pp. 305–314.

[9] M. Steff, B. Russo, Co-evolution of logical couplings and commits for defect estimation, in: Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on, IEEE, Zurich, Switzerland, 2012, pp. 213–216.

[10] C. Tantithamthavorn, A. Ihara, K.-I. Matsumoto, Using co-change histories to improve bug localization performance, in: Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2013 14th ACIS International Conference on, IEEE, Honolulu, HI, USA, 2013, pp. 543–548.

[11] E. Kouroshfar, Studying the effect of co-change dispersion on software quality, in: Software Engineering (ICSE), 2013 35th International Conference on, IEEE, San Francisco, CA, USA, 2013, pp. 1450–1452.

[12] M. D'Ambros, M. Lanza, R. Robbes, On the relationship between change coupling and software defects, in: Reverse Engineering, 2009. WCRE'09. 16th Working Conference on, IEEE, Lille, France, 2009, pp. 135–144.

[13] T. L. Graves, A. F. Karr, J. S. Marron, H. Siy, Predicting fault incidence using software change history, Software Engineering, IEEE Transactions on 26 (7) (2000) 653–661.

[14] P. Knab, M. Pinzger, A. Bernstein, Predicting defect densities in source code files with decision tree learners, in: Proceedings of the 2006 international workshop on Mining software repositories, ACM, Shanghai, China, 2006, pp. 119–125.

[15] A. T. Ying, G. C. Murphy, R. Ng, M. C. Chu-Carroll, Predicting source code changes by mining change history, Software Engineering, IEEE Transactions on 30 (9) (2004) 574–586.

[16] T. Zimmermann, A. Zeller, P. Weissgerber, S. Diehl, Mining version histories to guide software changes, Software Engineering, IEEE Transactions on 31 (6) (2005) 429–445.

[17] N. Fenton, B. Kitchenham, Validating software measures, Software Testing, Verification and Reliability 1 (2) (1991) 27–42.

[18] H. Zuse, A Framework of Software Measurement, Walter de Gruyter & Co., Hawthorne, NJ, USA, 1997.

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

[20] A. Abran, Software Metrics and Software Metrology, Wiley-IEEE Computer Society Pr, 2010.

[21] M. Cataldo, A. Mockus, J. Roberts, J. Herbsleb, Software dependencies, work dependencies, and their impact on failures, Software Engineering, IEEE Transactions on 35 (6) (2009) 864–878.

[22] V. R. Basili, D. M. Weiss, A methodology for collecting valid software engineering data, IEEE Trans. Softw. Eng. 10 (6) (1984) 728–738.

[23] N. Fenton, J. Bieman, Software metrics: a rigorous and practical approach, CRC Press, 2014.

[24] S. Kirbas, A. Sen, B. Caglayan, A. Bener, R. Mahmutogullari, The effect of evolutionary coupling on software defects: An industrial case study on a legacy system, in: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, ACM, Torino, Italy, 2014, pp. 6:1–6:7.

[25] S. Kirbas, A. Sen, B. Caglayan, A. Bener, Değişiklik bağlaşımı ve yazılım hataları ilişkisinin incelenmesi, in: Proceedings of the 8th Turkish National Software Engineering Symposium, Güzelyurt, KKTC, Turkey, September 8-10, 2014., 2014.

[26] F. S. Roberts, Measurement theory.

[27] I. B. of Weights, M. (BIPM), What is metrology? (August 2016).
URL http://www.bipm.org/en/worldwide-metrology/

[28] L. Cheikhi, R. E. Al-Qutaish, A. Idri, A. Sellami, Chidamber and kemerer object-oriented measures: Analysis of their design from the metrology perspective, International Journal of Software Engineering & Its Applications 8 (2).

[29] B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering, Tech. rep., Technical report, EBSE Technical Report EBSE-2007-01 (2007).

[30] Y. Zhou, M. Wursch, E. Giger, H. Gall, J. Lu, A bayesian network based approach for change coupling prediction, in: Reverse Engineering, 2008. WCRE'08. 15th Working Conference on, IEEE, 2008, pp. 27–36.

[31] H. Kagdi, S. Yusuf, J. I. Maletic, Mining sequences of changed-files from version histories, in: Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR '06, 2006, pp. 47–53.

[32] M. Colaço, M. Mendonça, F. Rodrigues, Mining software change history in an industrial environment, in: Software Engineering, 2009. SBES'09. XXIII Brazilian Symposium on, IEEE, 2009, pp. 54–61.

[33] A. Alali, B. Bartman, C. D. Newman, J. I. Maletic, A preliminary investigation of using age and distance measures in the detection of evolutionary couplings, in: Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, 2013, pp. 169–172.

[34] International Standards Organisation (ISO), ISO/IEC 14764: Software Engineering - Software Life Cycle Processes - Maintenance, 2006.

[35] L. Madeyski, M. Jureczko, Which process metrics can significantly improve defect prediction models? an empirical study, Software Quality Journal 23 (3) (2015) 393–422.

[36] B. Kitchenham, Whats up with software metrics?–a preliminary mapping study, Journal of systems and software 83 (1) (2010) 37–51.

[37] M. Mondal, C. Roy, K. Schneider, Improving the detection accuracy of evolutionary coupling by measuring change correspondence, in: Software Maintenance, Reengineering and Reverse Engineering (CSMR-WCRE), 2014 Software Evolution Week - IEEE Conference on, 2014, pp. 358–362.

[38] International Standards Organisation (ISO), ISO/IEC 15939: Software Engineering - Software Measurement Process, International standard (2007).

[39] M. Ericsson, W. Lowe, T. Olsson, D. Toll, A. Wingkvist, A study of the effect of data normalization on software and information quality assessment, in: Software Engineering Conference (APSEC), 2013 20th Asia-Pacific, Vol. 2, 2013, pp. 55–60.

[40] D. Beyer, A. Noack, Mining Co-Change Clusters from Version Repositories, Tech. rep. (2005).

[41] O. Appendix, Mathematical operations supported by scale types (August 2016).
URL `http://depend.cmpe.boun.edu.tr/scp/MathematicalOperationsSupportedbyScaleTypes.pdf`

[42] G. Pirklbauer, C. Fasching, W. Kurschl, Improving change impact analysis with a tight integrated process and tool, in: Information Technology: New Generations (ITNG), 2010 Seventh International Conference on, 2010, pp. 956–961. doi:10.1109/ITNG.2010.100.

[43] H. Kagdi, J. I. Maletic, Combining single-version and evolutionary dependencies for software-change prediction, in: Proceedings of the Fourth International Workshop on Mining Software Repositories, MSR '07, 2007, pp. 17–.