Empirical Software Engineering, 10, 81–103, 2005.

An Empirical Exploration of the Distributions of the Chidamber and Kemerer Object-Oriented Metrics Suite

GIANCARLO SUCCI[1] giancarlo.succi@unibz.it
Center for Applied Software Engineering, Free University of Bolzano-Bozen, Bolzano-Bozen, Italy

WITOLD PEDRYCZ pedrycz@ee.ualberta.ca
Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada

SNEZANA DJOKIC snezana@ee.ualberta.ca
Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada

PAOLO ZULIANI paolo.zuliani@unibz.it
Center for Applied Software Engineering, Free University of Bolzano-Bozen, Bolzano-Bozen, Italy

BARBARA RUSSO barbara.russo@unibz.it
Center for Applied Software Engineering, Free University of Bolzano-Bozen, Bolzano-Bozen, Italy

Editor: Audris Mockus

Abstract. The object-oriented metrics suite proposed by Chidamber and Kemerer (CK) is a measurement approach towards improved object-oriented design and development practices. However, existing studies evidence traces of collinearity between some of the metrics and low ranges of other metrics, two facts which may endanger the validity of models based on the CK suite. As high correlation may be an indicator of collinearity, in this paper, we empirically determine to what extent high correlations and low ranges might be expected among CK metrics.

To draw as much general conclusions as possible, we extract the CK metrics from a large data set (200 public domain projects) and we apply statistical meta-analysis techniques to strengthen the validity of our results. Homogeneously through the projects, we found a moderate (~0.50) to high correlation (>0.80) between some of the metrics and low ranges of other metrics.

Results of this empirical analysis supply researchers and practitioners with three main advises: a) to avoid the use in prediction systems of CK metrics that have correlation more than 0.80 b) to test for collinearity those metrics that present moderate correlations (between 0.50 and 0.60) c) to avoid the use as response in continuous parametric regression analysis of the metrics presenting low variance. This might therefore suggest that a prediction system may not be based on the whole CK metrics suite, but only on a subset consisting of those metrics that do not present either high correlation or low ranges.

Keywords: CK metrics, collinearity, object-orientation, software metrics, meta-analysis.

1. Introduction

The Chidamber and Kemerer (CK) object-oriented metrics suite (Chidamber and Kemerer, 1991; Chidamber and Kemerer, 1994) is a widely accepted standard for

measuring object-oriented software systems. It has been theoretically discussed and validated (Hitz and Montazeri, 1996; Churcher and Shepperd, 1995; Henderson-Sellers et al., 1996). Recent empirical studies have used multivariate regression models based on the CK suite to predict defect levels of software systems (Basili et al., 1996; El Emam et al., 1999) and to predict development costs (Briand and Wüst, 2001) and maintainability (Li and Henry, 1993).

In multivariate statistics it may happen that some of the variables of a regression model are collinear, thereby diminishing their effectiveness as predictors. A recent study evidences the presence of collinearity among three of the metrics in the CK suite: Coupling Between Objects (CBO), Number Of Methods (NOM), and Response For a Class (RFC) (Chidamber et al., 1998). Therefore, further analysis in general settings is needed to understand whether those metrics are reliable predictors in a prediction system based on regression techniques.

Other studies report that two metrics in the suite, Depth of the Inheritance Tree (DIT) and Number of Children (NOC), often assume very low values (Ronchetti and Succi, 1999; Briand and Wüst, 2001; Cartwright and Shepperd, 2000; Tang et al., 1998; Basili et al., 1996); this may affect the performance of statistical analysis techniques, as continuous distributions may not be suitable to properly capture the data.

In software engineering, companies often rely on their own internal development models or development guidelines—companies like Cisco, IBM and Nortel do it, for example. Given their simplicity and intuitiveness, the CK metrics are often included in such internal models. These models are likely to be developed by technical people who may not be aware of the high risks caused by collinearity and low ranges. In this work we study whether such two dangerous situations are the ''norm'', that is, whether CBO, NOM and RFC are in general highly correlated and DIT and NOC assume in general low or narrow range of values.

To draw as much general conclusions as possible, we first extract the CK metrics from a large data set (100 public domain C++ and 100 public domain Java projects). Then we analyze the collected data using statistical meta-analytical techniques, to generalize findings across projects. In particular, the correlations computed from the measures are combined using the meta-analytical technique of the Weighted Estimators of a Common Correlation (see Section 4.4, Hedges and Olkin, 1985). The distribution of the maximum values of DIT and NOC in each project is extracted. As in previous findings (Chidamber et al., 1998; Cartwright and Shepperd, 2000; and Chidamber and Kemerer, 1994), we may set 10 as the upper bound for data range of DIT and NOC. This threshold indicate also the impossibility of having ''quasi-normal'' distributions on natural numbers—low variability caused by low ranges prevents the approximation with continuous distributions and hampers the use of these metrics as dependent variables in a regression model. In Section 5.2 we also perform a sign test to establish whether the maximum value of each metrics for each project is below an even lower fixed threshold in a significant number of cases (see Section 4.4).

The paper is organized as follows. In Section 2, we present an overview of the CK metrics suite and some related literature. Then we give a brief account of existing uses of statistical meta-analysis in software engineering. In Section 3, we discuss the nature and the relevance of the problem. The detailed description of the experimental data and the

research method applied in this study are given in Section 4. Analysis of results and the discussion of the applied methods are provided in Section 5. Limitations of the meta-analytical approach are discussed in Section 6. Finally, conclusions of the investigation are drawn in Section 7.

2. Background

In this Section we shall present the CK metrics suite and review a few existing studies on the validation of the suite. We shall also briefly describe the use of some statistical meta-analysis techniques in software engineering.

2.1. The Chidamber and Kemerer Metrics Suite

Various suites of software measures have been proposed to determine whether a system is of a desired quality. The Chidamber and Kemerer metrics suite is a set of six object-oriented design metrics. A summary of their definitions follows; their complete definitions are in (Chidamber and Kemerer, 1994).

- *Weighted Method Count (WMC)*: summation of the complexity of each method in a class. If we consider complexity of each method unitary (as we assume here), WMC is equal to NOM (Li and Henry, 1993).

- *Depth of Inheritance Tree (DIT)*: maximum length of the path from a class to the root of the hierarchical tree.

- *Number of Children (NOC)*: number of immediate subclasses of a class.

- *Coupling Between Object classes (CBO)*: count of other classes to which a class is coupled. Two classes are coupled if one invokes a method or uses an attribute of the other.

- *Response For a Class (RFC)*: number of methods that can be executed in response to a message received by an object of a class.

- *Lack of Cohesion in Methods (LCOM)*: defined as the number of disjoint sets of local methods.

Several studies have been undertaken to validate CK metrics both theoretically and empirically.

Li and Henry (1993) examine the usefulness of five out of the six ''CK metrics'' in predicting effort in software maintenance. The study does not include CBO in the investigation, because it is not related to inheritance. The empirical validation is conducted on two commercial systems using multivariate linear regression. The initial

multivariate model contains ten predictors, including five CK metrics, two size measures, Number of Methods, Data Abstraction Coupling and Message-passing Coupling. Two size measures are discarded later because of the confirmed collinearity with other measures, therefore the final model includes eight regressors. The results show that these measures can be employed to predict maintenance effort in an object-oriented system, measuring number of lines changed in each class.

Basili et al. (1996) investigate whether the CK measures could provide relevant insight into fault-proneness of classes. The study is based on the data collected in a university setting from eight medium-sized information management systems developed with the same requirements. Univariate regression models are used with each CK metrics. The results indicate that in general, DIT, RFC, NOC and CBO are very significant, NOM is somewhat significant, and LCOM was shown to be insignificant in all cases.

Chidamber et al. (1998) explore the applicability of CK metrics for managerial purposes on three commercial object-oriented systems. The study is aimed at examining the relationships between these metrics and cost, quality, and productivity. The results show that high values of CBO and LCOM are associated with lower productivity, greater rework and greater design effort. Also, it is shown that the suite provides additional information compared only to size measures.

Harrison et al. (1998) investigate if CBO is a good measure of coupling and whether it can be a good indicator of class understandability and fault density. The study analyzes two coupling measures: CBO and NAS (Number of Associations) across five software systems from various sources. The results indicate that there is a strong relationship between these two measures, implying that only one is needed for the assessment of a system. No relationship is found between class understandability and coupling, and only limited evidence links increased coupling and fault density.

Daly et al. (1996) investigate the effects of DIT on the maintainability of object-oriented software, by means of formal experiments on student populations. The results suggest that software systems with three levels of inheritance require less time for maintenance, with respect to systems with no inheritance. However, systems with five levels of inheritance require more effort for the same purpose.

The correctness and the usefulness of this metrics suite have also been questioned. NOM is criticized for being language and assumption dependent (Churcher and Shepperd, 1995); CBO for naively treating all kinds of coupling as equal (Hitz and Montazeri, 1996); LCOM for being counter-intuitive (Hitz and Montazeri, 1996). There are several redefinitions of LCOM which take into account the cohesion stemming from a method invoking another method, and other aspects. El Emam et al. (1999) claim that the empirical usability of most of the metrics in the suite comes from the strong relationship of these metrics with the size measure.

Some researchers also suggest that these metrics can be supplemented with some other object-oriented measures, to provide a more complete picture (Li and Henry, 1993; Tang et al., 1998; Briand et al., 1999).

Despite the criticism, the ‘‘CK metrics’’ have still been widely cited and adopted since they are simple and intuitive to use and, as mentioned in a few experiments, they have shown their usefulness in constructing prediction systems for size and number of defects. The main advantage of the CK metrics is their availability in the design stage

of the software development process, when the classical size measure is not available, yet. This clearly helps developers in taking a ''healthy'' development road right from the beginning. The additional convenience of the CK metrics lies in the fact that several automatic-modeling tools, such as Rational Rose, GDPro, and Together are able to compute them.

2.2. Statistical Meta-Analysis

Statistical meta-analysis is the branch of statistics that studies how it is possible to combine statistical indexes across different experiments (Hedges and Olkin, 1985; Rosenthal, 1991; Cooper and Hedges, 1994; Hunter and Schmidt, 1990).

Statistical meta-analysis has already been used in software engineering. Hu is probably the first researcher attempting to combine results from different studies (1997). His study evaluates four alternative production models that have not been applied widely in software engineering: linear, quadratic, Cobb-Douglas and translog. The performances are checked using the P-test. The comparison of the four alternative models suggests that quadratic software production model could be applied to a wide variety of projects, while the other three have limited practical applicability.

The comprehensive analysis by Pickard et al. (1998) targets the relationship between project effort and product size in various software companies. The investigation shows that there is a high correlation between those variables, and that heterogeneity of data does not affect the result too much. The analysis includes a test of sensitivity for meta-analysis, and an attempt to achieve more consistent results through exclusion of a small portion of data points.

Hayes (2000) focuses on investigating the efficacy of different software requirements inspection techniques. The meta-analysis includes five experiments, the original and four replications. The paper does not give a clear opinion whether a unique result can be achieved; rather, it provides a discussion of the heterogeneity found in the data sets.

Miller (2000) examines the effectiveness of several defect detection techniques. The analysis includes five datasets. The study concludes that the common estimate cannot be reached due to the diversity of the experiments and the data sets.

The experimental study by Briand et al. (2002) attempts to devise a general prediction model to use across different software systems. Two mid-size Java projects developed in the same company are used. The results indicate that it is possible to devise a statistical model on one software system and to apply it successfully to the other.

In summary, some studies found that common conclusions can be drawn from several experiments, while others found that it is not possible.

3. Nature and Relevance of the Problem

In this Section we shall present and discuss the problems which stem from collinearity and low ranges. We shall also briefly review some previous work on the subject.

3.1. Relevance of Collinearity

Collinearity is the presence of a linear relationship between two or more independent variables in a multivariate model. Its presence usually prevents a precise determination of the statistical parameters under study (Belsley, 1991). In this subsection we explain *intuitively* why collinearity is dangerous when building multivariate linear models.

Suppose that we want to estimate a dependant variable Y using two independent variables, X_1 and X_2. It is common in this case to use Ordinary Least Squares (OLS) multiple regression and to determine the coefficients a, b, and c so that we can express the expected values of Y as:

$$Y = aX_1 + bX_2 + c \tag{1}$$

If X_1 and X_2 are linearly related, we can estimate X_1 with X_2 using the following linear equation, where d and f are the mean square coefficients:

$$X_1 = dX_2 + f \tag{2}$$

We can restructure (1) as follows (t is a real number):

$$Y = aX_1 + bX_2 + tX_1 - tX_1 + c = (a + t)X_1 + bX_2 - tX_1 + c \tag{3}$$

We can then substitute the term tX_1 with $t(dX_2 + f)$:

$$Y = (a + t)X_1 + (b - td)X_2 + c - tf \tag{4}$$

We see that there are infinitely many real values of t for which $(a + t) \neq a$ or $(b - td) \neq b$ or $(c - tf) \neq c$. Therefore we cannot build a unique model with the OLS multiple regression, since the effects of the independent variables are unclear. We could even change the direction of the effect, as evidenced by Fenton and Neil (1999).

Statistics textbooks warn to check for the presence of collinearity and suggest that, if collinearity appears, one of the independent variables should be discharged (Aron and Aron, 1997).

Belsley (1991) offers a comprehensive analysis of collinearity, and provides a review of and suggestions for possible diagnostic and improvement techniques. Among them there are:

- Examining the eigenvalues and eigenvectors of the cross-product matrix. The square root of ratio of the largest and the smallest eigenvalues of the cross product of the design matrix is called condition index and is a commonly used measure of collinearity.

- Studying the correlation matrix of the least square estimator based on partial correlations between any two columns in this matrix.

- Using the correlation matrix and its inverse, under the assumption of a normal data distribution.

- Investigating correlation matrix of explanatory variables since a high correlation may imply collinearity.

This research aims at determining if *in general* there are linear relationships between the three variables, CBO, NOM, and RFC. Since we focus on only three variables, we may take the simpler approach of studying the correlation matrix of the least square estimators. Had we dealt with more variables, we should have introduced other techniques and considered the problem of mutual dependence among three or more variables at the same time.

It is well known that high correlation may imply collinearity, but collinearity does not imply correlation (Belsley, 1991). However, we are interested in verifying the *presence* of collinearity, and not its *absence*. In view of this, a correlation estimation looks like a reasonable methodology.

Correlation measures usually have one weakness as collinearity diagnostic; there is no obvious cutoff for how large a correlation must be to indicate collinearity. In many statistical studies, correlations of the magnitude 0.45 are considered large (Barnston, 1994). These are usually studies interested in testing the hypothesis that no correlation exists. Conversely, in studies presuming the existence of a relation and interested in knowing if it is strong, correlations like 0.45 are considered small. We consider as a flag of possible collinearity a presence of the correlation between two variables higher than 0.5 (Cohen, 1977). In this paper high correlation would mean correlation >0.80 and moderate correlation, correlation >0.50.

Using correlations on a large amount of data to determine the presence of collinearity has two major advantages in this study:

(a) It works also with non-parametric correlations, more suited for situations when no assumption can be made on the distribution of data

(b) It is amenable for the application of statistical meta-analysis and it increases the probability of detecting true collinearity between variables.

3.2. Analyzing Integer-Valued Measures Assuming Low Values on an Absolute Scale

When integer-valued measures are on an absolute scale, they assume only positive integers values. If the range of such integers is limited, extra care should be paid in the analysis, as the continuous parametric statistical analysis techniques for continuous data may not work properly.

We provide an intuitive explanation of this problem with an example. Suppose we are measuring an integer-valued attribute X on an absolute scale and we obtain the results presented in Table 1.

Using the ''usual'' parametric methods we would say that the data have a central tendency (represented by its mean) of 0.925 and a spread (represented by its standard deviation) of 1.186. Moreover, the 95% confidence interval computed over a normal curve is [−1.40, 3.25].

Clearly such numbers make little sense. A measure on an absolute scale would never assume a value of 0.925, nor would its confidence interval include a negative number. In

Table 1. Example of collected measures.

Value of the measure of attribute X	Frequency of the measure
0	20
1	10
2	5
3	3
4	2

addition, the range [−1.40, 3.25] has size 4.65, while we know that all data are in the interval [0, 4] of size 4 and 95% of it is in the interval [0, 3] of size 3.

In cases like this, it is more appropriate either to apply other statistical measures such as median, mode and range or to assume different, more suitable distributions (e.g. the Poisson distribution (Lloyd, 1999)).

In the present paper we consider the threshold value of 10 for the data ranges of the metrics DIT and NOC. Our choice has been motivated by previous results (e.g. Chidamber et al., 1998; Cartwright and Shepperd, 2000; and Chidamber and Kemerer, 1994) and by our box plot analysis of their maximum values across the projects (see Figure 1). In Section 5.2 we also perform a sign test to investigate whether the maximum value of each metrics for each project is below an even lower fixed threshold in a significant number of cases (see Section 4.4).

3.3. Previous Work in the Area

In Chidamber et al. (1998), the authors suggest the presence of collinearity between CBO, NOM, and RFC studying three industrial projects written in C++ and Objective C. The collinearity is diagnosed on the basis of significant correlations calculated between these measures. The study includes three industrial software systems and the results

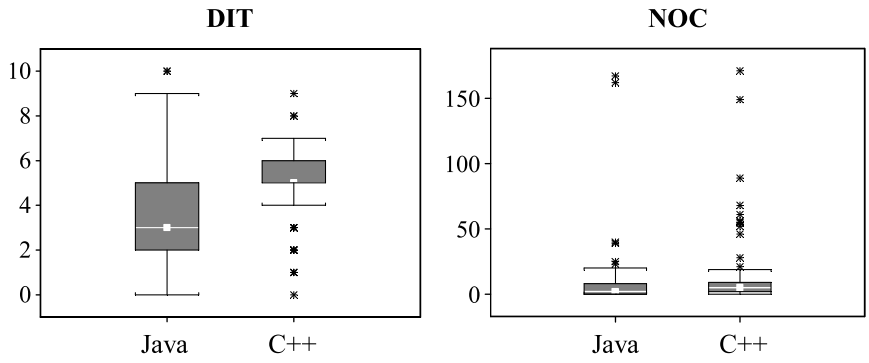

Figure 1. Distributions of the maximum values of DIT and NOC.

indicate that such high correlation exists in general. The study also notes that DIT and NOC assume low values and attributes this fact to the chosen design preferences.

Such collinearity is not analyzed in the earlier study on two object-oriented systems by Li and Henry (1993), although their experimental results show that DIT and NOC usually assume very low values.

The work of Basili et al. (1996) reports of an analysis based on eight medium-sized systems developed on identical requirements in C++. They find that the linear Pearson's correlations between these three measures are quite weak. The authors conclude that these measures are mostly independent in statistical terms, and do not capture redundant information excessively. Descriptive statistics of the CK metrics obtained from the experiments show that in general DIT and NOC assume low values compared to other metrics in the suite.

The results in Tang et al. (1998) show that DIT and NOC have narrow ranges across three observed systems, while the analysis on collinearity is not provided.

Ronchetti and Succi (1999) do not report on the cross-correlations between CBO, NOM, and RFC; however, they confirm that DIT and NOC assume low values.

4. Research Method

In this Section we describe the data used in the study and the analysis techniques adopted.

4.1. Description of the Data

In this study we use public domain projects downloaded from the web. We have selected projects available through the web without any prescribed rule or algorithm of randomness. Size and type vary casually. The language is either Java or C++. The values for the CK metrics have been extracted from the code using the WebMetrics tool (Succi et al., 1998). A summary of the values of the number of classes for the 100 Java and the 100 C++ projects is in Table 2.

Our goal is to determine if there are high correlations among some of the object-oriented metrics belonging to the CK suite and whether the ranges for DIT and NOC are usually narrow.

Since we deal with two different programming languages, we use the programming language as a moderator variable.

Table 2. Descriptive statistics for the number of classes in the 100 C++ and 100 Java projects.

	Min	Max	Median	St. Dev.
Java	28	936	83.5	169.58
C++	30	2520	59	268.29

The drawback of these public domain data sets is the lack of sufficient information on the projects, since they do not come from controlled environments.

This is a common problem in studies requiring generalization of findings, such as epidemiology or education (Cohen, 1977; Cook et al., 1994; Liao, 1998; Belanger, 1997). For this reason, meta-analysis contains tests of homogeneity, meant to determine if the overall data have the same nature and meta-analytic results can be considered valid. We anticipate that the homogeneity tests were satisfied for all the three correlations studied.

4.2. Our Approach to the Detection of Collinearity

The goal of this paper is to warn practitioners and researchers on the possibility of the existence of variables that may endanger the efficiency of a prediction system based on CK metrics. We focused on two specific behaviors: a) the presence of high correlation between variables or b) narrow range of their values.

The former may be indicator of collinearity, although it is not yet clear to what extent, since it depends on the square root of ratio of the largest and the smallest eigenvalues of the correlation matrix—see Section 3.1. Then we propose the following approach based on meta-analysis techniques to detect collinearity.

1. We approximate Spearman's correlation with the Pearson's correlation

2. We use the Weighted Estimators of a Common Correlation technique to determine the 95% confidence interval of the approximated Pearson's correlation coefficient for each project.

3. We check the homogeneity of the Pearson's approximated coefficient across the different projects.

The latter implies low variability of the corresponding distribution. This prevents the approximation of the original distribution by a normal continuous distribution, which is the main ingredient for using the meta-analysis techniques of Section 4.3. Therefore we may use different techniques—for example based on binomial distribution (Lloyd, 1999)—which may be a matter of future research.

4.3. Analysis of High Correlation

Making general conclusions about the presence of high correlation is difficult and needs to satisfy three conditions:

1. The collection of a **large amount of data** from many sources to satisfy meta-analysis assumptions.

2. The application of **suitable statistics** to summarize the data into a single, coherent framework.

3. The elimination of **confounding effects** from other independent variables.

To approach the **first** issue, we have collected from the web and analyzed 100 public domain Java projects and 100 public domain C++ projects. We use the programming language as moderator variable. In this way we hope to obtain **(a)** results that can be generalized to the wider population of Java and C++ projects and **(b)** indication for general rules, applicable across different programming languages and, possibly, phases of development.

Framed in terms of research questions, we want to test two hypotheses:

- $H_1^{1\text{-}C++}$: There is no high correlation between CBO, RFC, and NOM in the analyzed C++ Projects

- $H_1^{1\text{-}Java}$: There is no high correlation between CBO, RFC, and NOM in the analyzed Java Projects

The target significance level is 0.05, as common in software engineering (Mišić and Tešić, 1997; Nesi and Querci, 1998; El Emam et al., 2001; Wood et al., 1999).

The **second** challenge is that we cannot simply average correlations of samples to get the correlation (Hedges and Olkin, 1985). Studies have shown that the correlations of samples are biased estimators of the correlation of the original population. In addition, the range of a correlation is $[-1, 1]$. Therefore, the distribution of correlations is not normal. To address these issues, we employ a meta-analytical technique.

The **third** issue deals with the potential confounding effect of size, so that measures that appear to be related, in reality are just different expressions of the same size attribute. El Emam et al. (1999) warn that this is usually overlooked because most of the proposed models use only one independent variable. They show that high correlation between specific measures and fault proneness is usually caused by high correlation between size and fault proneness and size and the independent variable.

In this study, the most commonly used size metrics, LOC, is not considered, as we aim at studying collinearity among metrics which are accessible in the early part of the software life cycle, when LOC is not available. RFC and NOM are size measures in the sense of (Basili et al., 1996). Also, CBO could be considered a size measure as it counts how many links exist between a class and other classes.

Therefore, in a sense, the research of El Emam et al. is a precursor of this work, as several defect prediction models for object-oriented systems use CBO, RFC, and NOM (Basili et al., 1996; Chidamber et al., 1998; Li and Henry, 1993; Ronchetti and Succi, 1999).

4.4. Statistical Meta-Analysis and Weighted Estimators of a Common Correlation

The method usually employed for estimating a common correlation from several studies is the meta-analytical technique called Weighted Estimators of a Common Correlation (Hedges and Olkin, 1985). The technique assumes that:

- sample data come all from normally-distributed populations with the same correlation;
- sample correlations are computed according to Pearson's definition of correlation.

Since we cannot assume any particular distribution of the samples, we compute the non parametric Spearman's correlation coefficients of the samples. Such correlation can be computed for any distribution of the population.

From Spearman correlation coefficient r_s one can approximate Pearson correlation coefficient by using the coefficient r_c computed by the following formula (Pearson, 1907):

$$r_c = 2\sin\left(\frac{\pi}{6} r_s\right) \tag{5}$$

Studies show that r_c generally underestimates Pearson correlation coefficient, though the error is usually very small: in simulation studies the largest error found between the converted correlation coefficient and the actual correlation was -0.005. The error decreases when the sample size and the actual population correlation increase (Kendall, 1949; Kendall and Gibbons, 1990; Rupinski and Dunlap, 1996). This turns out to be useful in our case, as we deal with large data sets and we clearly do not want to raise ''false alarms'' by overestimating the correlation coefficients.

The steps of the Weighted Estimators of a Common Correlation technique are:

1. Normalization of the sample correlations with the Fisher z transformation.
2. Computation of the required confidence interval for the transformed correlations, in our case the 95% confidence interval.
3. Application of the inverse Fisher z transformation on the resulting range.

At the end, a check is performed on whether the results disprove the original assumption on homogeneity of the data. For this purpose, an extension of the chi-square test is used.

Given below, there is the summary of the three steps as described in (Hedges and Olkin, 1985).

Suppose that samples of size $n_1,\ldots,n_k$ are taken from k studies; we compute the sample correlations $r_1,\ldots,r_k$ (in our case we compute the Spearman correlations and then we transform them with the formula (5)). The sample correlations are then transformed by the Fisher z transform:

$$z(r_i) = \frac{1}{2}\log\frac{(1+r_i)}{(1-r_i)} \equiv z_i \tag{6}$$

Given a set of transformed correlation coefficients $z_1,\ldots,z_n$, we can now compute the mean transformed value $\bar{z}$ as a weighted average of the z_is:

$$\bar{z} = \sum_{i=1}^{k} w_i z_i \tag{7}$$

where the weight for the i-th experiment is computed over the size n_i of sample i as:

$$w_i = \frac{(n_i - 3)}{\sum_{j=1}^{k}(n_j - 3)} \tag{8}$$

The 95% confidence interval in the transformed space, $[z^-, z^+]$, is then determined using the ''usual'' rule for normal distributions:

$$z^- = \bar{z} - \frac{1.96}{\sqrt{\sum_{i=1}^{k}(n_i - 3)}} \tag{9}$$

$$z^+ = \bar{z} + \frac{1.96}{\sqrt{\sum_{i=1}^{k}(n_i - 3)}} \tag{10}$$

To determine the 95% confidence interval on the original correlation coefficient, $[r^-, r^+]$, we have to compute the inverse Fisher z transform, z^{-1},on z^- and z^+:

$$z^{-1}(x) = \frac{(e^{2x} - 1)}{(e^{2x} + 1)} \tag{11}$$

At the end, we have to check if the original data is indeed homogeneous. To do so, a chi-square test against the null hypothesis of homogeneity is performed. Given the sample size and the power of the test, a non-rejection of the null hypothesis amounts to its acceptance.

The Q statistics used in this case is:

$$Q = \sum_{i=1}^{k}(n_i - 3)(z_i - \bar{z})^2 \tag{12}$$

We compare it with the chi-square threshold value for k-1 degrees of freedom. If it is lower, then the dataset is considered homogeneous, otherwise it is not (Hedges and Olkin, 1985).

4.5. Analysis of the Ranges

To determine the size of the ranges of the metrics, we first define what a ''small range'' is. In our paper we assume that a range of size 10 is small, as suggested by Figure 1. This

choice is rather arbitrary and based on previous works (see Chidamber et al., 1998; Cartwright and Shepperd, 2000; and Chidamber and Kemerer, 1994), but we believe that 10 should be sufficiently large to capture the distribution trend (see also Briand and Wüst, 2001). Small ranges hamper the use of techniques based on (quasi-)normal continuous distribution, as in such a range a distribution has low variability and hardly may approximate a normal one.

In Section 5.2 we also perform a sensitivity analysis of our choice by setting lower thresholds.

We proceed applying a sign test on whether the maximum value of the target metrics in the project is below our threshold in a significant number of cases.

Formally, the hypotheses to test are as follows:

- $H_0^{2\text{-C++}}$: The maximum values of NOC and DIT are not below 10 in the analyzed C++ Projects

- $H_0^{2\text{-Java}}$: The maximum values of NOC and DIT are not below 10 in the analyzed Java Projects.

It is our goal to test this hypothesis at 0.05 significance level.

A description of the applied sign test procedure follows.

For each project the maximum value of DIT (or NOC) is compared with the threshold. If it is below the threshold a ''$-$'' is assigned to the project, otherwise a ''+''. Then the total number of ''+'' symbols is computed and the probability of getting such number or any lower number by chance is determined.

The probability C of obtaining by chance exactly k projects out of n with maximum value of DIT (or NOC) lower than the threshold, is given by the binomial distribution:

$$C(k) = \frac{\mathrm{n!}}{\mathrm{k!(n-k)!}} p^k (1-p)^{n-k} \tag{13}$$

where:

- n is the total number of projects (100)

- p is the probability of the maximum value for a project being lower than the threshold by chance (0.5)

The total probability P of obtaining by chance exactly k projects with maximum value of DIT (or NOC) lower than the threshold, is given by:

$$P(k) = \sum_{i=0}^{k} C(i) \tag{14}$$

Also in this case, we set the significance level to 0.05, that is, if the total probability is above 0.05 the null hypothesis cannot be rejected.

5. Analysis of the Results

5.1. Correlation

The three object-oriented metrics of interest show a high correlation (Table 3). This holds for both values of the moderator variable. All the computed coefficients lie below the 0.05 significance level.

The correlations between NOM and LCOM are also very high, and higher in Java projects than in C++. The higher correlation in Java projects could be explained with the presence of accessor methods, more common in Java than in C++. In fact, for every ''relevant'' attribute in a Java class, there should be *get* and *set* methods for accessing such attribute. In case of *n* attributes in a class, there will be typically *2n* accessor methods. Clearly, this results in an increased number of methods for a class. As each couple {*get*, *set*} does not have any common variable with any other couple {*get*, *set*}, LCOM tends to increase dramatically with the number of {*get*, *set*} methods. At the moment the WebMetrics tool cannot exclude *get* and *set* methods from the metric calculation. This is indeed part of future work.

The high values of the correlations between RFC and LCOM are considered a byproduct of the high correlations between NOM and LCOM.

Table 4 contains the results of the homogeneity test. We compare Q values with $Q_{0.05}$, and if the calculated Q is lower than $Q_{0.05}$ we can conclude that the data are homogeneous for that particular relationship. In that case, we can draw general conclusions on the collinearity investigated.

In our datasets the homogeneity test is satisfied for all three cases under consideration. Considering that the data are not coming from controlled experiments, but rather from easily available public projects, the results of the homogeneity tests are very good, strongly confirming our hypotheses. The tests do not confirm the relationship between NOM and LCOM, while they confirm the relationship between RFC and LCOM.

Table 3. Summary of the weighted correlations.

		NOM	DIT	NOC	CBO	RFC	LCOM
Java projects	NOM	1	0.03	0.18	0.57	0.82	0.99
	DIT		1	−0.03	0.26	0.21	0.04
	NOC			1	0.03	0.08	0.19
	CBO				1	0.87	0.56
	RFC					1	0.81
	LCOM						1
C++ projects	NOM	1	0.46	0.24	0.59	0.97	0.92
	DIT		1	0.12	0.38	0.50	0.47
	NOC			1	0.04	0.21	0.24
	CBO				1	0.72	0.49
	RFC					1	0.85
	LCOM						1

Table 4. Summary of homogeneity test.

		DIT	NOC	CBO	RFC	LCOM
Java projects $Q_{0.05} = 6291.61$	NOM	3188.16	401.82	1404.50	5236.62	6553.93
	DIT		2095.26	1153.95	2726.33	2276.49
	NOC			254.41	398.97	395.09
	CBO				2064.25	1016.83
	RFC					2801.56
C++ projects $Q_{0.05} = 6171.21$	NOM	1629.97	431.64	1187.20	1417.07	12932.93
	DIT		326.33	947.39	935.21	1168.64
	NOC			339.61	321.52	429.93
	CBO				698.89	683.62
	RFC					1366.98

From the two tables above we can reasonably infer that:

- RFC is generally highly correlated with any of the NOM, CBO and LCOM (homogeneity test passed in all the cases and for both Java and C++ languages). Therefore RFC has shown a high correlation with NOM, CBO and LCOM, homogeneously through the projects. RFC then is a variable that is likely to be collinear with NOM, CBO and LCOM. This strongly advises that prediction models should never use RFC as independent variable in conjunction with any of the other three mentioned metrics;

- CBO is moderately correlated with NOM (homogeneity test passed for both languages). In this case we suggest that any prediction model employing both CBO and NOM should be subject of thorough tests for collinearity, such as the Belsley-Kuh-Welsh (BKW) test for multicollinearity. It would have to be done case by case as the empirical evidence we gathered is not as strong as for the RFC case above.

- Even though it appears that correlations between NOM and LCOM came from heterogeneous populations (Table 4), the correlations are so high (Table 3) it is worth advising the caution of using these variables together.

5.2. Ranges

The boxplots in Figure 1 evidence the ranges of maximum values of DIT and NOC in the analyzed projects.

The results of the sign tests are that the probabilities of obtaining maximum values below 10 by chance are below our 0.05 significance threshold, as we can see from Table 5.

From the analyses presented above, it can be concluded that the maximum values vary more in Java projects. The range of maximum values for NOC is quite narrow and similar for both programming languages.

Table 5. Summary of the results of the sign test (threshold = 10).

	DIT	NOC
Java	0.00	1.35E-10
C++	3.98E-27	9.05E-08

It is also evident that the maximum values are slightly higher for C++ than for Java projects, and the presence of outliers is more significant in C++ projects.

Although the threshold 10 is valid for all the cases, we have further investigated whether we can set a lower threshold (Table 6).

The detailed analysis shows that for different programming languages and DIT and NOC, different thresholds can be set. The lowest threshold is 4 (NOC–Java) and the highest is 6 (DIT–Java and NOC–C++). The conclusion can be made that both measures (DIT and NOC) assume values in narrow ranges for both programming languages, Java and C++.

Such a low variability implies that:

- continuous parametric statistics (confidence intervals over normal distribution, etc.) should not be used when analyzing DIT and NOC data;

- DIT and NOC cannot be usefully employed in regression models, as there would be little chance to explain variance of the dependent variable. Chidamber et al. (1998) defined variables HINOC and HIDIT, which take value 1 or 0 depending on whether the corresponding metric exceeds a threshold value specific to the data set. They found out that over three data sets only in one (small) data set those Boolean variables were significantly predictive.

In the end, this low variability should not be seen as a shortcoming of the two metrics, it might rather reveal a precise design policy that makes a little use of inheritance (Chidamber et al. 1998).

Table 6. Summary of the results of the sign test for different values of the threshold.

Threshold	DIT–Java	DIT–C++	NOC–Java	NOC–C++
9	7.97E-29	3.22E-24	9.05E-08	2.35E-06
8	1.31E-25	6.26E-23	8.34E-07	1.76E-03
7	9.56E-16	1.27E-16	6.29E-06	6.66E-02
6	6.29E-06	1.04E-10	9.16E-05	1.35E-10
5	1	2.04E-04	8.95E-04	0.90
4	1	0.24	3.32E-03	0.99
3	1	1	0.31	1

6. Limitations

Meta-analysis is not free from controversy. Several researchers warn about the risks associated with its application in software engineering (Miller, 2000 and Pickard et al., 1998).

1. Combining results from different experiments poses a high risk of comparing apples and oranges.

2. There is limited or no control on the quality of the data, since meta-analysis deals with results provided by other scientists.

3. Effect sizes are not always checked for significance.

4. The randomization of the data sets, the prerequisite for correct meta-analysis, is sometimes not checked.

5. Scientists are not always provided with the raw data.

6. Homogeneity test is not always performed.

7. Using high correlation to detect collinearity may create bias in the results. It is not clear at what level correlations make the regression problem unstable, since it depends on condition index—the square root of ratio of the largest and the smallest eigenvalues of the correlation matrix.

The first problem is related to the fact that in our study little is known about the personnel capability and the level of object-oriented programming experience of the project teams. This is partially overcome by the size of the sample and by the fact that we still deal with code. Although the code originates from a very wide range of sources (public domain software), it is still Java and C++ code. However, future work on the subject should better address this problem.

The second problem does not arise, since the authors have full control over all the experiments. The data collection has been performed using a solid metrics extraction tool, WebMetrics (Succi et al., 1998). In this way the measurement errors have been avoided and all the data have the same consistent quality.

As for the magnitude of effect size, in this investigation we report large effect sizes for both moderator variables, which cast away any doubt on the statistical significance of the results.

We also try to eliminate the bias and maintain randomization of the studies by including all the individual studies and providing numbers of classes for each project that was available. Nonetheless we need to remark that all available data may not be a perfect representation of all projects, including the ones for which data are not available.

The advantage of dealing with the raw data is also on our side, since we have access to the source code for all the individual studies in question.

The always-critical test of homogeneity is satisfied for the correlations of interest in this study.

This paper deals with high correlations and evaluates collinearity through meta-analysis techniques on a large number of projects. The reliability of our results depends on the accuracy used in the adoption of meta-analysis. Conclusions are guaranteed by the analysis of a large number of available projects preventing bias and distortions. An accurate use of meta-analysis techniques—such as the Weighted Estimators for Common Correlation (Hedges and Olkin, 1985) that we have used in this paper—on a large number of projects usually produce solid results that may be very different form the ones using standard statistical techniques on a more limited number of projects—sometimes even opposite, see Basili et al. (1996). Namely, peculiarities of a limited number of projects may affect results, addressing different conclusions.

Our contribution warns practitioners and researchers of situations of danger in the construction of predictive systems and support results of previous existing studies. Different approaches and future research may further confirm our results.

7. Conclusions

This paper presents an investigation on two aspects of CK object-oriented metrics: the presence of collinearity among three measures in the suite, CBO, NOM and RFC, and the narrow ranges of NOC and DIT. We consider a large data set made of public domain Java and C++ projects. To generalize conclusions across experiments, we analyze the data by means of statistical meta-analysis techniques.

The meta-analytical study of the cross-correlations shows that for both Java and C++ code, one should generally expect a moderate to high correlation. The homogeneity test confirms that the results obtained from the meta-analytic study are valid.

For the second issue, we consider a range ''small'' if it contains at most 10 points. A sign test on whether the maximum value of DIT and NOC is higher than 10 confirms that this is not the case. A further analysis shows that the value of 6 still satisfies the sign test for all the considered measures.

The implications for practitioners and researchers are:

- prediction models should generally never use RFC in conjunction with any of the NOM, CBO and LCOM metrics;

- prediction models employing CBO and NOM should be thoroughly tested for collinearity. In the positive case, suitable solution strategies should be developed, such as dropping one of the collinear independent variables (Aron and Aron, 1997) or using Poisson distributions (Lloyd, 1999);

- NOC and DIT, due to their low variance cannot be analyzed with continuous parametric statistical methods. So with our approach we cannot test them for high correlation and consequent possible collinearity.

Future work on this subject should aim at confirming (or not) our results with a more controlled data set. In particular, it would be useful to carry out an empirical investigation of similar size on commercial software packages, for which there are usually available more information about the project, revision history, faults, and personnel capability.

Acknowledgments

The authors acknowledge the support of the Natural Science and Engineering Research Council of Canada, the Government of Alberta and University of Alberta. Special thanks also go to Eric Liu and Raymond Wong for their significant contribution to this work. Thanks also to Luigi Benedicenti and Arrigo L. Frisiani for their valuable feedback on this work.

A special thank is due to the referees for their accurate revision of the paper and their essential advises.

Note

1. Corresponding author: Prof. Giancarlo Succi, Center for Applied Software Engineering, Free University of Bolzano-Bozen, Piazza Domenicani, 3, 39100 Bolzano-Bozen, Italy. Phone: +39-0471-315640; Fax:+39-0471-315649.

References

Aron, A., and Aron, E. N. 1997. *Statistics for the Behavioral and Social Sciences*. Prentice Hall.

Barnston, A. G. 1994. Linear statistical short-term climate predictive skill in the northern hemisphere. *J. Climate* 7: 1513–1564.

Basili, V. R., Briand, L. C., and Melo, W. L. 1996. A validation of object-oriented design metrics as quality indicators. *IEEE Trans. Softw. Eng.* 22(10): October, 751–761.

Belanger, S. E. 1997. Literature review and analysis of biological complexity in model stream ecosystems: Influence of size and experimental design. *Eco. Environ. Saf.* 36(1): 1–16.

Belsley, D. A. 1991. *Conditioning Diagnostics: Collinearity and Weak Data in Regression*. New York: J. Wiley.

Briand, L., and Wüst, J. 2001. Modeling development effort in object-oriented systems using design properties. *IEEE Trans. Softw. Eng.* 27(11): November, 963–986.

Briand, L., Ikonomovski, S. V., and Lounis, H. 1999. Investigating quality factors in object-oriented designs: An industrial case study. *Proc. 21st Int. Conf. on Softw. Eng*. Los Angeles, May 16–22, 345–354.

Briand, L., Melo, W., and Wüst, J. 2002. Assessing the applicability of fault-proneness models across object-oriented software projects. *IEEE Trans. Softw. Eng.* 28(7): July, 706–720.

Cartwright, M., and Shepperd, M. 2000. An empirical investigation of an object-oriented software system. *IEEE Trans. Softw. Eng.* 26(8): August, 786–796.

Chidamber, S. R., and Kemerer, C. F. 1994. A metrics suite for object-oriented design. *IEEE Trans. Softw. Eng.* 20(6): June, 476–493.

Chidamber, S. R., and Kemerer, C. F. 1991. Towards a metrics suite for object oriented design, *Proceedings of the 6th ACM Conference on Object-Oriented Programming, Systems Languages, and Applications (OOPSLA'91)*, Phoenix, AZ, pp. 197–211.

Chidamber, S. R., Darcy, D. P., and Kemerer, C. F. 1998. Managerial use of metrics for object-oriented software: An exploratory analysis. *IEEE Trans. Softw. Eng.* 24(8): August, 629–639.

Churcher, N., and Shepperd, M. J. 1995. Comments on ''A metrics suite for object oriented design''. *IEEE Trans. Softw. Eng.* 21(3): March, 263–265.

Cohen, J. 1977. *Statistical Power Analysis for the Behavioral Sciences*. New York: Academic Press.

Cook, T. D., Cooper H., Cordray D. S., Hartmann H., Hedges L. V., Light R. J., Louis T. A., and Mosteller F. 1994. *Meta-Analysis for explanation—A casebook*.

Cooper, H. M., and Hedges, L. V. (eds.) 1994. *The Handbook of Research Synthesis*. New York: The Russell Sage Foundation.

Daly, J., Brooks, A., Miller, J., Roper, M., and Wood, M. 1996. Evaluating inheritance depth on the maintainability of object-oriented software. *Empir. Soft. Eng.* 1(2): 109–132.

El Emam, K., Benlarbi, S., and Goel, N. 1999. The confounding effect of class size on the validity of object-oriented metrics. *Technical Report, NRC/ERB-1062*, September.

El Emam, K., Melo, W., and Machado, J. C. 2001. The prediction of faulty classes using object-oriented metrics. *J. Syst. Softw.* 56(1): 63–75.

Fenton, N. E., and Neil, M. 1999. A critique of software defect prediction models. *IEEE Trans. Softw. Eng.* 25(5): September/October, 675–689.

Harrison, R., Counsell, S., and Nithi, R. 1998. Coupling metrics for object-oriented design. *Proc. 5th Int. Symp. Softw. Metr*. Bethesda Maryland, November.

Hayes, W. 2000. Research synthesis in software engineering: A case for meta-analysis. *Proc. 6th IEEE Int. Symp. Softw. Metr*. Boca-Raton, Florida, USA, November.

Hedges, L. V., and Olkin, I. 1985. *Statistical Methods for Meta-Analysis*. Orlando: Academic Press.

Henderson-Sellers, B., Constantine, L. L., and Graham, I. M. 1996. Coupling and cohesion (towards a valid metrics suite for object-oriented analysis and design). *Object Oriented Syst.* 3(3): 143–158.

Hitz, M., and Montazeri, B. 1996. Chidamber and Kemerer's metrics suite: A measurement theory perspective. *IEEE Trans. Softw. Eng.* 22(4): 267–271.

Hu, Q. 1997. Evaluating alternative software production functions. *IEEE Trans. Softw. Eng.* 23(6): June, 379–387.

Hunter, J. E., and Schmidt, F. L. 1990. *Methods for Meta-Analysis: Correcting Error and Bias in Research Findings*. Newbury Park, CA: Sage.

Kendall, M. G. 1949. Rank and product-moment correlation. *Biometrika* 36: 177–193.

Kendall, M. G., and Gibbons, J. D. 1990. *Rank Correlation Methods*, 5th edition. New York: Oxford University Press.

Li, W., and Henry, S. 1993. Object-oriented metrics that predict maintainability. *J. Syst. Softw.* 23: 111–122.

Liao, Yuen-Kuang Cliff. 1998. Effects on hypermedia versus traditional instruction on students' achievement: A meta-analysis. *J. Res. Comput. Educ*. 30(4): 341–359.

Lloyd, C. J. 1999. *Statistical Analysis of Categorical Data*. Wiley-Interscience.

Miller, J. 2000. Can results from software engineering experiments be safely combined? *Proc. 6th IEEE Int. Symp. Softw. Metr*. Boca-Raton, Florida, USA, November.

Mišić, V. B., and Tešić, D. N. 1997. Estimation of effort and complexity: An object-oriented case study. *J. Syst. Softw.* 41(2): 133–143.

Nesi, P., and Querci, T. 1998. Effort estimation and prediction of object-oriented systems. *J. Syst. Softw.* 42(1): 89–102.

Pearson, K. 1907. Mathematical contributions to the theory of evolution. XVI. On further methods of determining correlation. *Drapers' Company Research Memoirs* (Biometric Series 4). Cambridge University Press.

Pickard, L. M., Kitchenham, B. A., and Jones, P. W. 1998. Combining empirical results in software engineering. *Inf. Softw. Technol.* 40(14): 811–821.

Ronchetti, M., and Succi, G. 1999. Early estimation of software size in object-oriented environments a case study in a CMM level 3 software firm. Submitted to *IEEE Trans. Softw. Eng.*

Rosenthal, R. 1991. *Meta-Analytical Procedures for Social Research*, Revised edition. Newbury Park, CA: Sage.

Rupinski, M. T., and Dunlap, W. P. 1996. Approximating Pearson product-moment correlations from Kendall's tau and Spearman's rho. *Educ. Psychol. Meas.* 56(3): 419–429.

Succi, G., Benedicenti, L., Bonamico, C., and Vernazza, T. 1998. The Webmetrics project—exploiting software tools on demand. *World Multiconference on Systemics, Cybernetics, and Informatics*. Orlando, FL.

Tang, M. H., Kao, M. H., and Chen, M. H. 1998. An empirical study on object-oriented metrics. *Proc. 6th IEEE Int. Symp. Softw. Metr*. Boca-Raton, Florida, USA, November.

Wood, M., Daly, J., Miller, J., and Roper, M. 1999. Multi-method research: An empirical investigation of object-oriented technology. *J. Syst. Softw.* 48(1): 13–26.

Giancarlo Succi received his Laurea Electrical Engineering and Ph.D. in Computer and Electrical Engineering from the University of Genova, in 1988 and in 1993 respectively, and his MSc in Computer Science from SUNY Buffalo in 1991. He is now Professor and Director of the Center for Applied Software Engineering at the Free University of Bolzano-Bozen.

His research interests include experimental software engineering, agile methodologies, and software product lines.

Witold Pedrycz received his M.Sc, Ph.D., and D.Sci. in Computer Science from the Silesian Technical University, Gliwice, Poland, in 1978, 1980, and 1986 respectively. From 1998 to 2001 he served as Professor and Director of Computer Engineering at the Department of Electrical & Computer Engineering of the University of Alberta. He is now chair of the Department of Electrical & Computer Engineering of the University of Alberta.

Prof. Pedrycz holds a Canada Research Chair since 2001 and is IEEE Fellow since 1999.

His research interests embrace various research areas of Computer Engineering and Computer Science including Computational Intelligence (granular computing including fuzzy set technology, neural networks and evolutionary computing), pattern recognition, data mining, and emerging behaviour and adaptive systems.

Snezana Djokic received her master of Science graduate from the University of Alberta specializing in Software Engineering with outstanding GPA and academic credentials. In the same university she was Research Associate, Lab Instructor and Teaching Assistant till 2002.

Paolo Zuliani after receiving a Laurea Degree in computer science in 1997 from Università degli Studi di Milano, Italy, began graduate studies at Oxford University. He completed his work toward a Ph.D. degree as a member of the Programming Research Group in the Computing Laboratory under the supervision of Dr. Jeff Sanders.

He is currently Assistant Professor in the Faculty of Computer Science at the Free University of Bolzano-Bozen, Italy.

Barbara Russo received her Laurea degree and her PhD degree from the University of Trento, Italy, respectively in 1991 and 1996, all in Mathematics. Dr. Russo was post-doc fellow and research assistant in Mathematics at the University of Trento, Italy. She is currently a assistant professor at the Center for Applied Software Engineering of the Free University of Bolzano-Bozen.

Her current research interests focus on Defects Detection Process, Statistical Analysis and Software Measure.