The Repeatability of Code Defect Classifications

Khaled El Emam
Isabella Wieczorek

Fraunhofer Institute for Experimental Software Engineering,
Sauerwiesen 6
D-67661, Kaiserslautern
Germany
{elemam,wieczo}@iese.fhg.de

Abstract

Counts of defects found during the various defect detection activities in software projects and their classification provide a basis for product quality evaluation and process improvement. However, since defect classifications are subjective, it is necessary to ensure that they are repeatable (i.e., that the classification is not dependent on the individual). In this paper we evaluate a slight adaptation of a commonly used defect classification scheme that has been applied in IBM's Orthogonal Defect Classification work, and in the SEI's Personal Software Process. The evaluation utilizes the Kappa statistic. We use defect data from code inspections conducted during a development project. Our results indicate that the classification scheme is in general repeatable. We further evaluate classes of defects to find out if confusion between some categories is more common, and suggest a potential improvement to the scheme.

Keywords: *defect classification, software inspections, measurement reliability, agreement.*

1 Introduction

The classification of software defects plays an important role in measurement-based process and product improvement. This is evidenced in, for example, the Orthogonal Defect Classification (ODC) work [11][13], whereby Defect Types are matched with Defect Triggers to identify potential problems during a software project. The distribution of defects by type can be used to identify product and process problems [8][12], and can be used as a project planning and monitoring tool [35]. The relationship between defect types and other variables such as whether a module was new or modified [4], and the cost of correction [32][6], can provide insight into development activities. Defect causal analysis methods utilize defect classes for clustering defects and focusing the causal analysis meeting [10]. Incorporation of defect types is considered to improve the accuracy of capture-recapture models for defect content estimation [39], and for improving the applicability of reliability growth models [14].

A basic premise of all of these approaches is that the defect classification is repeatable[1]. For example, the semantic classification proposed in ODC is "likely to be accurate" and is believed to be less error-prone than opinion-based classifications [13]. However, since the classification of defects is a subjective exercise, it is plausible that different individuals would classify the same defect in a different category. If such disagreement in classifications is prevalent, then there is justifiable doubt in basing improvement decisions and investments on analyses that utilize defect class information. Furthermore, a commonly suggested data analysis approach for defect data is a chi-square test to determine if all defect classes are equally likely (e.g., see [29]) and for investigating the relationship between defect class and other variables (e.g., see [6]). However, chi-square tests of contingency tables whereby one of the variables has low reliability (e.g., say one of the variables is the defect class) are known to produce quite

1 We use the terms “reliability” and “repeatability” interchangeably in this paper.

misleading results [21].

There are different ways in which a defect can be classified. For example, a semantic classification that characterizes the fix [13], by the phase where the defect is injected [36], and by characterizing it as an omission, or as a commission [5]. There are a number of different additional classification schemes that have been suggested in the literature, for example at the SEI [24], and an IEEE Standard for defect classification that presents a number of different classification schemes [28]. While special care has been taken in defining classification schemes that are believed to be repeatable, this has not been, to our knowledge, empirically demonstrated through systematic investigation.

In this paper we report on a study that evaluated the repeatability of a defect classification scheme using real code inspection data. It is advantageous to use real inspection data since artificial or seeded defects may be harder/easier to classify than real defects, hence questioning the applicability of results using non-real defects. The defect classification scheme that we evaluate is a slight adaptation of the ODC scheme [11][13], which has also been incorporated in the SEI's Personal Software Process [27]. We use two data sets totalling 605 inspection defects.

Briefly, our results indicate that this classification scheme has sufficiently high repeatability. Furthermore, the method that we follow can be applied to evaluate the repeatability of other defect classification schemes using data that is commonly collected during software inspections.

In the next section we present our research method, including the environment of study, the defect classification scheme, the data analysis method, and a sensitivity analysis approach. In Section 3 we present the results and discuss their limitations. We conclude the paper in Section 4 with a summary and directions for future research.

2 Research Method

2.1 Environment of Study

The data we use in our study comes from a development project conducted within a company in Germany. The system consists of approximately 30 KSLOC and has a peak staff load of 5 persons. The application is a data analysis program that implements a proprietary data mining technique.

The goal was to perform effective code inspections under tight resource constraints. Thus, the code inspection process was restricted to two persons. A two-person inspection was found to be a promising and useful approach by Bisant and Lyle [9]. They report that this approach saves considerable manpower compared to conventional inspections with larger teams. Furthermore, it was found to be as effective as conventional, three to five person inspections. Based on this approach, Kusumoto et. al. [31] defined more precisely how the two persons are selected from a development team. Major achievements were a decrease of inspection effort, decrease of unreviewed documents, and an increase of completion rate.

All defect data was collected during code inspections. The inspection process consisted of the following steps: planning, preparation, meeting, correction, and follow-up. Each step may involve the following roles: moderator, author, and inspector. One person (A) assumes the moderator and the inspector role. The other person (B) has the role of the author and the inspector. Bisant and Lyle [9] recommend removing the role of the moderator, which is not the case in our process.

During the planning step, the moderator puts together documents that are handed in by the author. These are the code document, the test-cases, and the documentation. The moderator also sets up a date for the meeting, and ensures that both inspectors have the same version of the documents. After the planning step, during the preparation, the inspectors individually check the documents. They detect and classify defects using a

Step	Role	Person(s)
Planning	Moderator	A
Preparation	Inspector	A and B
Meeting	Moderator	A
	Inspector	A and B
	Author	B
Correction	Author	B
Follow-up	Moderator	A
	Author	B

Table 1: Steps of the inspection process.

Defect Type	Description and Examples of Questions
Documentation	*Comments, Messages* • Is the function described adequately at the top of the file ? • Are variables described when declared ? • Does the function documentation describe its behavior properly ?
Build/Package	*Change management, library, version control* • Is there a version number defined for the file ? • Are the correct versions of functions included in the build ?
Assignment	*Declaration, duplicate names, scope, limits* • Are variables initialized properly ? • Are all library variables that capture a characteristic or state of the object defined ? • Are all return values that are special cases (e.g., an error return) really invalid values (i.e., would never occur unless there was an error) ?
Interface	*Procedure calls and references, I/O, user formats, declarations* • Does the library interface correctly divide the functions into their different types ? • Do the functions follow the proper object access rules ? • Are the declared and expected interface signatures the same ?
Checking	*Error messages, inadequate checks* • Are all possible error conditions covered ? • Are appropriate error messages given to the user ? • Does the function return the <error> value in case of errors ? • Is there checking or debugging code that is left in the function that shouldn't be there ? • Does the function check for missing data before making a computation ? • Are all checks for entry conditions of the function correct and complete ?
Data	*Structure, content, declarations* • Are files opened with the right permissions ? • Are the correct data files accessed ? • Are there any missing variables for the object definition ? • Are variable definitions of the right size to hold the data ?
Function	*Logic, pointers, loops, recursion, computation* • Are all branches handled correctly ? • Are pointers declared and used as pointers ? • Are arithmetic expressions evaluated as specified ?
Memory	*Memory allocation, leaks* • Are objects instantiated before being used ? • Do all objects register their memory usage ?
Environment	*Design, compile, test, or other support system problems* • Are all test cases running properly? • Are compile options set properly (e.g., after changing compiler version) ?
Naming Conventions	*Naming of files, functions, and variables* • Do the function and file names follow the naming conventions for the project ? • Do the variable names follow the naming conventions for the project ?
Understandability	*Hinder understandability* • Are there enough explanations of functionality or design rationale ? • Are there any misleading variable names? • Are the comments clear and correctly reflect the code ?

Table 2: Defect classification scheme used in this study.

checklist. Each inspector fills out one defect report form while preparing. This form contains a defect's location, description, and classification. After preparation, the two inspectors hold an inspection meeting. For each defect found during preparation, a decision is made whether it is a "real" defect or a false positive. A false positive is an issue documented during preparation, but not considered as a "real" defect in the meeting. Furthermore, the participants reach an agreement/consensus on the classification of a "real" defect. Additional defects may also be found during the meeting, but this is not its major goal. All "real" defects are logged on a meeting form. In contrast to this process, Kusumoto et. al [31] merge the preparation and meeting step. In this case, both inspectors jointly check the product using a checklist.

After the meeting, the author corrects the logged defects. During the follow-up, the moderator checks whether all defects are corrected. Table 1 summarizes our inspection steps, the roles involved, and the participants in each step.

All defects considered for this study were found by two different pairs of inspectors with a total of three different inspectors, giving two different data sets, one for each pair (i.e., the first data set consists of all inspections conducted by inspectors A and B, and the second data set consists of all inspections conducted by inspectors B and C).

2.2 Defect Classification Scheme

The defect classification scheme that was used in this study is based on the original scheme developed in the Orthogonal Defect Classification work [13][11]. This scheme has been adopted in the Personal Software Process developed at the SEI [27]. Some of the defect classes were not included for the project under study because they were not directly relevant. In particular, the "timing/serialization" defect type was removed since it was not applicable to this type of application, and the "algorithm" type was removed since a commercial-off-the-shelf library was used that implemented most of the algorithms that were required for this application. A number of defect classes were also added. Namely, these were "Data" and "Environment" (both are also included in [27]), “Memory” (covered as “System” in [27]), "Naming Conventions", and "Understandability" since these were believed to require specific actions different from the other defect classes. The final defect classification scheme is presented in Table 2 with a number of illustrative questions that ought to be asked about the defect during the preparation step. This helps finding defects and classifying them.

Inspector B

Inspector A	$Class_1$	$Class_2$	...	$Class_k$	
$Class_1$	P_{11}	P_{12}		P_{1k}	P_{1+}
$Class_2$	P_{21}	P_{22}		P_{2k}	P_{2+}
...					
$Class_k$	P_{k1}	P_{k2}		P_{kk}	P_{k+}
	P_{+1}	P_{+2}		P_{+k}	

Table 3: Example kxk table for representing *proportions* of defect classifications made by 2 inspectors.

2.3 Data Analysis

The objective of this section is to discuss different coefficients that can be used for evaluating agreement in defect classification for two inspectors.

Data from a reliability study can be represented in a table such as Table 3 for a classification scheme with k defect classes. Here we have two inspectors that have independently classified the defects that they found. Inspectors independently classify defects during the preparation step of the inspection process. The table would include the proportion of ratings that fall in each one of the cells.

In this table P_{ij} is the proportion of ratings classified in cell (i,j), P_{i+} is the total proportion for row i, and P_{+j} is the total proportion for column j:

$$P_{i+} = \sum_{j=1}^{k} P_{ij}$$

$$P_{+j} = \sum_{i=1}^{k} P_{ij}$$

The most straightforward approach to evaluating agreement is to consider the proportion of ratings upon which the two inspectors agree:

$$P_O = \sum_{i=1}^{k} P_{ii}$$

However, this value includes agreement that could have occurred by chance. For example, if the two inspectors employed completely different criteria for classifying defects, then a considerable amount of observed agreement would still be expected by chance[2].

There are different ways for evaluating extent of agreement that is expected by chance. We will consider two alternatives here. The first assumes that chance agreement is due to the inspectors assigning classes to defects randomly at equal rates. In such a case chance agreement would be:

$$P_e = \frac{1}{k} \quad \textbf{Eqn. 1}$$

An alternative definition of chance agreement considers that the inspectors' proclivity to distribute their classifications in a certain way is a source of disagreement:

$$P_e = \sum_{i=1}^{k} P_{i+} P_{+i} \quad \textbf{Eqn. 2}$$

The marginal proportions in the above equation are maximum likelihood estimates of the population proportions under a multinomial sampling model [1]. If each of the inspectors makes classifications at random according to the marginal proportions, then the above is chance agreement (derived using the multiplication rule of probability and assuming independence between the two inspectors).

A general form for agreement coefficients is [40]:

$$Agreement = \frac{P_O - P_e}{1 - P_e}$$

When there is complete agreement between the two inspectors, P_O will take on the value of 1. The observed agreement that is in excess of chance agreement is given by P_O - P_e. The maximum possible excess over chance agreement is 1 - P_e. Therefore, this type of agreement coefficient is the ratio of observed excess over chance agreement to the maximum possible excess over chance agreement.

If there is complete agreement, then the agreement coefficient is 1. If observed agreement is greater than chance, then the agreement coefficient is greater than zero. If observed agreement is less than would be expected by chance, then the agreement coefficient is less than zero.

An agreement coefficient[3] that considers chance agreement as in Eqn. 1 is Bennett et al.'s S coefficient [7]. An agreement coefficient that considers chance agreement as in Eqn. 2 is Cohen's Kappa (κ)[4,5] [18].

A priori, in an inspection context, it seems a reasonable assumption that inspectors have a prior tendency to classify defects in a certain way, therefore suggesting that Cohen's Kappa is a more appropriate coefficient. Furthermore, there is considerable use in the social and medical sciences of the Kappa coefficient. For instance, Kappa has been used to evaluate the agreement in identifying mental disorders, such as depression, neurosis, and schizophrenia [22]. Umesh et al. [38] note that up to April 1988 Kappa had been cited more than 1100 times in social science research. This number is undoubtedly much larger by now. Furthermore, in medical methodology texts Kappa has been presented as a measure of agreement in diagnosis reliability studies [2][3][25].

2 Hartmann [26] notes that percentage (or proportion) agreement tends to produce higher values than other measures of agreement, and discourages its use since the tradition in science is to be conservative rather than liberal. A more detailed analysis of the behavioral literature where proportion agreement was used concluded that large fractions of these observations would be deemed unreliable if corrections for chance agreement were considered [37]. Therefore, in general, the use of percentage or proportion agreement is not recommended as an evaluative measure.

3 It should be noted that “agreement” is different from “association”. For the classifications of two inspectors to agree, the classifications must fall in the same defect class. For the classifications from the two inspectors to be associated, it is only necessary to be able to predict the defect class of one inspector from the defect class of the other inspector. Thus, strong agreement requires strong association, but strong association can exist without strong agreement.

4 A weighted version of Kappa has also been defined [19]. This allows weighting disagreements, and can be useful in the case of non-nominal scales, and when the relative costs of disagreements can be quantified. Neither of these apply to our current study, and therefore we do not consider weighted Kappa.

5 A priori one may expect that a defect classification scheme with 11 classes, such as ours, would exhibit higher repeatability than a scheme with much fewer classes. One study showed that repeatability increases only slightly when the number of classes increases above 7 [17]. However, the above study used an intraclass correlation coefficient as a measure of repeatability, which was shown to be equivalent to weighted Kappa using a particular weighting scheme [30]. In our study we do not assign weights.

Extensive use in various disciplines means that guidelines have been developed for interpreting a particular statistic. In [20] a review of the literature in various disciplines provides guidelines for interpreting Kappa, as well as interpretation guidelines for using Kappa in evaluating the reliability of software process assessments. In general, Kappa values less than 0.45 indicate inadequate agreement. Values above 0.62 indicate good agreement, and values above 0.78 indicate excellent agreement.

We can also test the null hypothesis that the observed amount of agreement could have occurred by chance (i.e., the inspectors independently classifying at random according to their marginal proportions). The standard error of Kappa has been derived by Fleiss et al. [23] and can be used for hypothesis testing. (the appendix includes further discussions of performing hypothesis testing in our context). Since we have two data sets in our study, all statistical tests are conducted at a Bonferonni adjusted alpha level (see [34]). The experimentwise alpha level that we used is 0.05.

2.4 Sensitivity Analysis

In our study we can only include defects that were found by both inspectors during preparation. However, not all defects logged during the inspection meeting are found by both inspectors. Therefore, it can be argued that there is bias in this kind of analysis because we do not include all defects found during inspections. If there is bias then the results of a reliability analysis are not applicable to inspections in general.

For example, let's say that for a subset of defects both inspectors find, they both classify this subset as "Function". Further, assume that "Function" type defects constitute the majority of defects found by both inspectors. This would mean that there is high agreement on classifying "Function" defects. Because Kappa can be considered as a weighted average of the agreement on each class [21], this will increase overall calculated Kappa. However, if "Function" defects are a small fraction of all the defects logged during the meeting then the calculated Kappa (using only defects found by both inspectors) would be highly inflated compared to the value that would be obtained if we used all logged defects.

It is therefore prudent to perform a sensitivity analysis to determine the extent to which the calculated Kappa value is inflated or deflated. We want to find out what would happen to Kappa if inspector B had classified the defects that were not found by B, and if all the defects that were not actually found by inspector A were classified by A. We do this through a Monte Carlo simulation [33]. To construct this simulation we have to define how an inspector would classify defects that s/he did not actually find.

	Defect Classifications				
	Insp. A	**Insp. B**	**Logged**	**Simulated Values for A**	**Simulated Values for B**
1	X		X	N/A	Discrete((“Y”,0.5),(“X”,0.5))
2	X	Y	X	N/A	N/A
3	Y	X	X	N/A	N/A
4	Y		Y	N/A	Discrete((“Y”,0.34),(“X”,0.66))
5		Y	Y	Discrete((”Y”,1),(“X”,0))	N/A
6		Y	X	Discrete((“Y”,0.34),(“X”,0.66))	N/A
7		X	X	Discrete((“Y”,0.34),(“X”,0.66))	N/A
8		X	Y	Discrete((“Y”,1),(“X”,0))	N/A
9		X	Y	Discrete((“Y”,1),(“X”,0))	N/A
10	Z		Z	N/A	Discrete((“Y”,0.43),(“X”,0.57))

Table 4: Hypothetical defect classifications for two inspectors, A and B, across *all* inspections where they both participate. The table also includes the specifications for the simulations using discrete distributions.

Two situations are possible:

- For a defect, one inspector did not find this defect, but overall s/he found defects of this type
- For a defect, one inspector did not find this defect, and overall s/he never found defects of this type

To explain the simulation, we refer to the hypothetical data in Table 4 for this discussion. This hypothetical table assumes that two inspectors A and B are involved in all inspections, our defect classification scheme has only three classes: "X", "Y", and "Z", and ten defects were found in all performed inspections. The table shows each inspector's classification and the final logged classification. If an inspector's column is empty, this means that the inspector did not find this defect. The last two columns in the table show the parameters of the distribution that are used to simulate the inspectors' classification. Since we use discrete distributions, for each value to be simulated we have the category ("X", "Y", or "Z") and its probability of occurrence. An “N/A” entry indicates that a simulated value is not necessary since the inspector found that defect.

Let us consider the simulation for inspector B. During the inspection meeting, inspector B is presented with the defects that s/he did not find, and these are classified by both inspectors (i.e., logged) as type "X" (this is the case for defect number 1). By looking at the classification of defects that B did find, we can determine how B classifies defects that are logged as of type "X". For example, for the defects that B found, s/he classifies as "Y" 50% of the defects that are subsequently logged during the meeting as "X" (these are defects number 2 and 6), and s/he classifies the remaining 50% as "X" (these are defects number 3 and 7). During the simulation we make B's classification for the defects that s/he did not find and that were logged as "X" to be a discrete distribution with probabilities 0.5 and 0.5 for their "X" and "Y" classifications respectively. This is illustrated in the last column of the table for defect number 1. Note that since inspector B did not find any defects of type "Z" this is not included in the simulation (or can be included with a probability of zero). This is an average case assumption because we assume that the inspector is guessing with these proportions. This is done for all defects that B did not find (e.g., defect number 4 which is a defect that was logged as "Y" but that was not found by inspector B), with one exception described below.

If B never found defects of a certain type that were logged during a meeting, for example defects of type "Z", then we make an assumption that B will guess according to his or her overall proclivity. This proclivity is calculated from all defects that B did find. For example, B classifies 43% of all the defects that s/he finds as "Y" and 57% as "X", then we construct a discrete distribution with a 0.43 and 0.57 probability (this is the case for defect number 10). Similar to the procedure for inspector B, we constructed the distributions for inspector A. These distributions are presented in the fourth column of Table 4.

All simulations we performed used 1000 iterations. By considering the simulated Kappa distribution we can determine the extent to which Kappa would be affected had both inspectors classified all defects. In particular, we wish to find out the extent to which the Kappa value would be equal to or fall under the 0.62 threshold.

3 Results

3.1 Description of Data

As noted earlier, we have two data sets. In total, these two data sets represent 605 defects found during inspections (see Table 5). In the first data set only 23% of the defects are found by both inspectors, and 24% in the second data set. This reflects the fact that in this environment these inspectors specialize in finding different types of defects.

The distribution of defects for the first data set is illustrated in the histogram of Figure 1. A number of points can be arrived at from this figure. First, most of the defects that are logged during the meeting (almost half) are "Documentation" type defects. This reflects that in this environment documentation standards are not enforced consistently. Second, that there is a substantial difference in the distribution of all defects that are logged against those that are found by both inspectors. Again, this

	Logged Defects	Found by Both
Data Set 1	432	99
Data Set 2	173	41

Table 5: Overall summary of the two data sets.

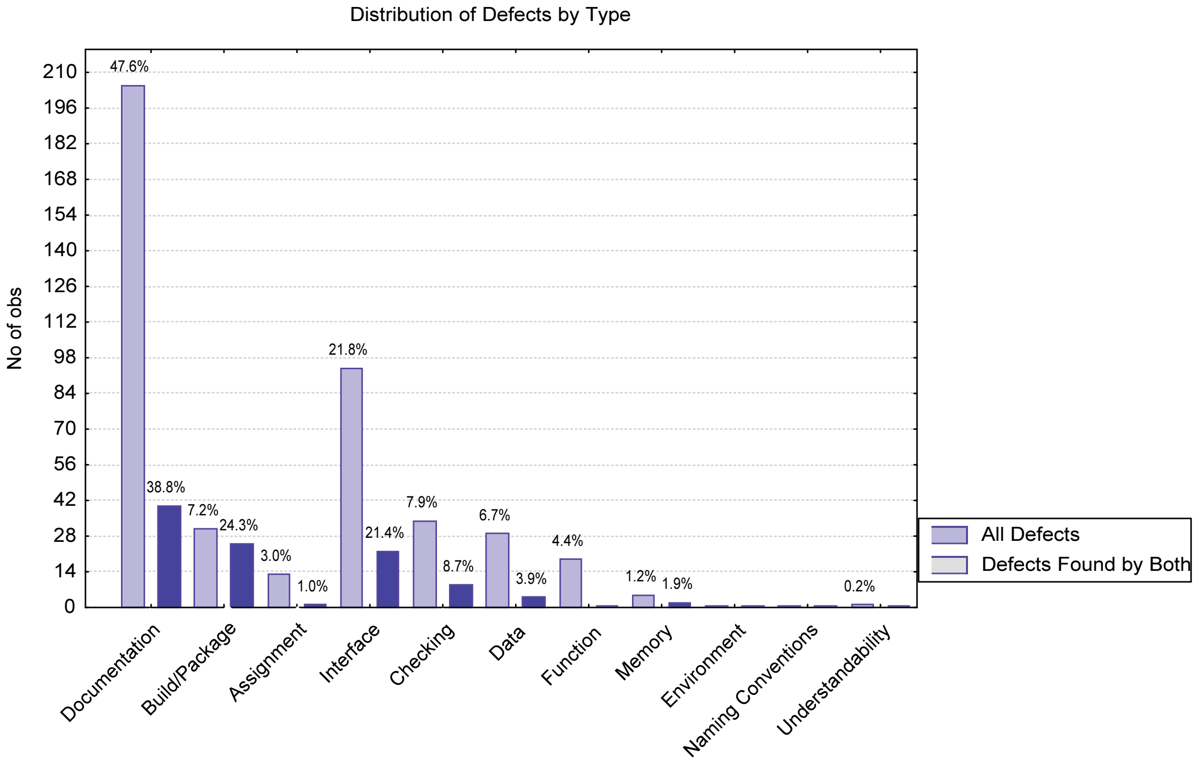

Figure 1: Distribution of defects for data set 1.

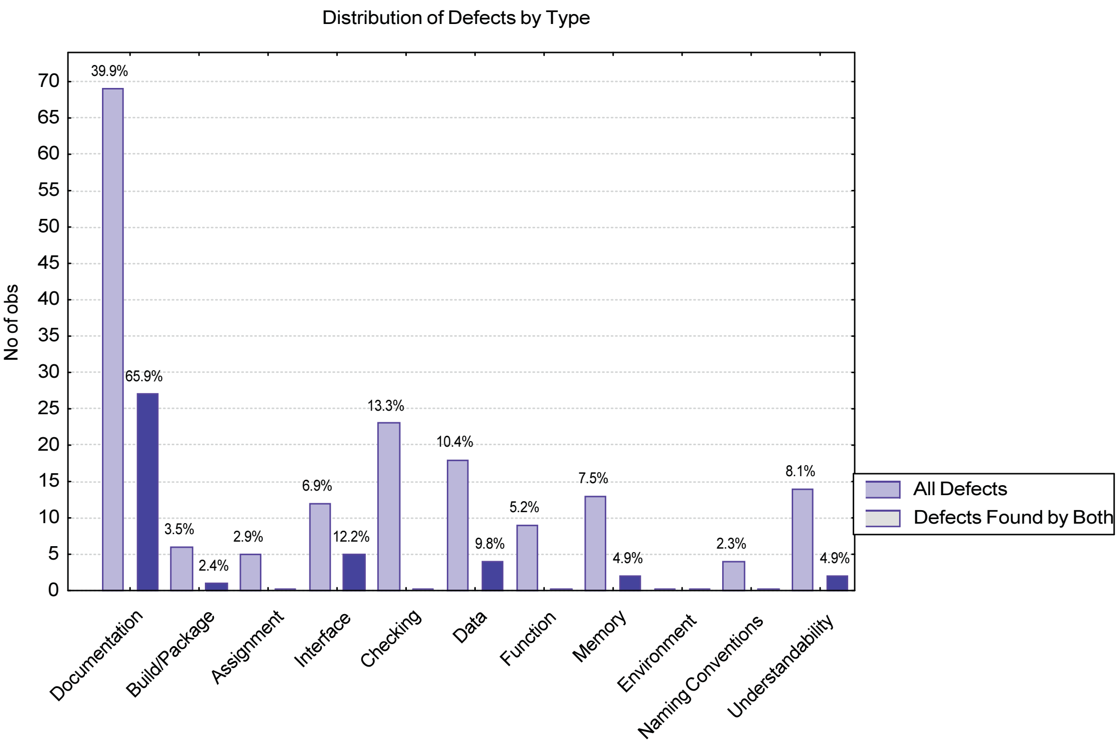

Figure 2: Distribution of defects for data set 2.

Data Set	Kappa Value
Data Set 1	0.66*
Data Set 2	0.82*

Table 6: Kappa values for the two data sets (the asterisk indicates statistical significance at an experimentwise alpha level of 0.05).

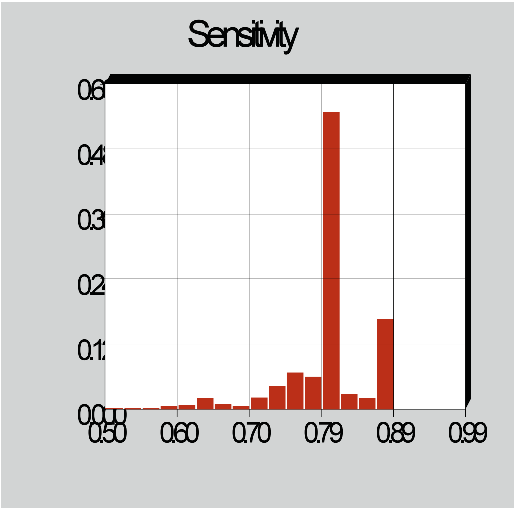

Figure 3: Sensitivity of Kappa for data set 1 (mean 0.80). The y-axis is the frequency from 1000 iterations, and the x-axis is the value of Kappa.

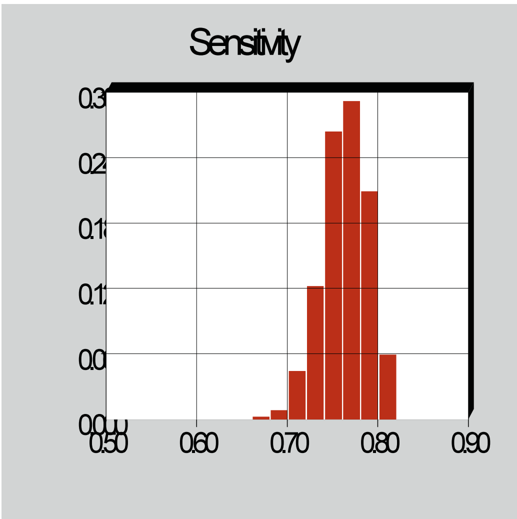

Figure 4: Sensitivity of Kappa for data set 2 (mean 0.76). The y-axis is the frequency from 1000 iterations, and the x-axis is the value of Kappa.

Data Set	Kappa Value
Data Set 1	0.73*
Data Set 2	0.82*

Table 7: Kappa values for the two data sets after combining the “Assignment” and “Data” defect classes (the asterisk indicates statistical significance at an experimentwise alpha level of 0.05).

reflects the specialization of these inspectors.

The distribution of defects for the second data set is shown in Figure 2. Again, most of the defects that are found are "Documentation" type, and the specialization effect is visible.

3.2 Evaluation of Classification Agreement

The Kappa coefficients for both data sets are shown in Table 6. They are both above 0.62 indicating good agreement, with the second data set above 0.78 indicating excellent agreement. Both values are statistically significant.

3.3 Results of Sensitivity Analysis

The results of the sensitivity analysis for the first data set are shown in Figure 3, where the frequency distribution is depicted. For data set 1 there is an inflation of Kappa since most values are greater than the calculated 0.66, and the mean is 0.80. As can be seen, the proportion of times that the values of Kappa are at 0.62 or lower is close to zero. This indicates that under the assumptions made during the sensitivity analysis, the extent of agreement will still be good almost all of the time.

The frequency distribution of simulated Kappa for the second data set is shown in Figure 4. For data set 2 there is a slight deflation of Kappa since most simulated values are slightly lower than the calculated 0.82, and the simulated Kappa distribution has a mean of 0.76. However, it can be seen that in quite a few cases the value of Kappa does go below 0.78, but very rarely if ever does it go below 0.62.

Based on this sensitivity analysis, we can conclude that the extent of agreement is sufficiently high, making the defect classification scheme reliability at least good and therefore is usable in practice as is.

3.4 Improving the Classification Scheme

To explore further why data set 1 has a lower Kappa value, we looked more closely at the raw data and found that there tends to be more disagreement between the "Data" and "Assignment" defect classes, for the second data set (this can be determined by looking at the cells in the contingency table). In addition, discussions with developers indicated that from their perspective there was sometimes difficulty in distinguishing between "Assignment" and "Data" defect classes. To investigate whether there was confusion amongst these two categories, we combined them and recalculated the Kappa value. If there is confusion amongst the categories then combining them is expected to improve the extent of agreement. The results for this are shown in Table 7. This indicates that for one data set there is substantial improvement in the extent of agreement after the combination.

Based on this result, it is suggested that the "Data" defect class be either refined further to clarify its distinguishing features from the "Assignment" class, or merged with the "Assignment" class.

3.5 Limitations

The important limitations of this study are concerned with its generalizability. There are two elements to generalizability: within the same organization, and to other organizations.

Our study was conducted for one type of document, code, for one defect classification scheme, and for one type of defect detection activity, inspections. Therefore, it is reasonable to make conclusions about the reliability of the defect classification scheme within this scope. However, it is early to make statements on the reliability of this defect classification scheme for other types of documents and for other activities. The nature of the defects found in other documents and using other defect detection activities can be quite different from code and inspections respectively, making the classification scheme harder/easier to use.

Our study was performed with a subset of the inspectors involved in the project. However, given the size of the project, this subset does represent a majority of those involved. Further studies need to be performed to determine whether the same results hold for other projects within the same organization.

It is expected that the distribution of defect types would be different for inspections in different organizations. The Kappa coefficient is affected by the marginal distributions. Therefore, in another organization with markedly different defect distributions, it is plausible that results dissimilar to ours would be obtained. However, this is an empirical question that would need to be answered for specific organizations. Our study can therefore be considered as initial evidence that this defect classification scheme is reliable.

For the inspection defect detection activity, the method that we have presented in this paper can be applied with data that is normally collected during software inspections. Therefore, it would be possible to perform reliability evaluation studies of the defect classification scheme for other document types in other organizations.

4 Conclusions

The classification of defects found during software development plays an important role in measurement-based process and product improvement. This is evidenced, for example, in the Orthogonal Defect Classification work and in the Personal Software Process. Many of the improvement decisions made when applying these approaches are based on the premise that the defect classification is repeatable. However, this assumption has not been empirically demonstrated in a systematic fashion thus far. Furthermore, while performing data analysis using defect class as a variable, it is known that low reliability of the variables can lead to quite erroneous results.

The objective of this paper was to evaluate the repeatability of a defect classification scheme in the context of software inspections. The study was performed using real inspection data and using a defect classification scheme similar to those in common use. Our results indicate that the defect classification scheme has high reliability by standards used in the social sciences, medical studies, and in other areas of software engineering. We further identified an improvement that can be made to the scheme to increase its reliability.

The method that we have presented can be applied for evaluating other defect classification schemes in the context of inspections. This method is particularly advantageous since it requires data that is usually collected during software inspections anyway.

Further research should also consider evaluating the repeatability of commonly used defect classification schemes for other defect detection activities, such as the various types of testing. Such research may converge on a classification scheme that is empirically demonstrated to be reliable for the whole of the defect detection life cycle.

Acknowledgements

We wish to thank Erik Dick, Steffen Gabel, Julien Fouth, and Igor DeCanck for their contributions to this study, and for helping to collect and organize the code inspections data. We also wish to thank Bernd Freimut, James Miller, and Audris Mockus for reviewing an earlier version of this paper.

References

[1] A. Agresti: *Categorical Data Analysis*. John Wiley and Sons, 1990.

[2] D. Altman: *Practical Statistics for Medical Research*. Chapman and Hall, 1991.

[3] P. Armitage and G. Berry: *Statistical Methods in Medical Research*. Blackwell Science, 1994.

[4] V. Basili and B. Perricone: "Software Errors and Complexity: An Empirical Investigation". In *Communications of the ACM*, 27(1):42-52, January 1984.

[5] V. Basili and H. D. Rombach: "Tailoring the Software Process to Project Goals and Environments". In *Proceedings of the 9th International Conference on Software Engineering*, pages 345-357, 1987.

[6] V. Basili, S. Condon, K. El Emam, B. Hendrick, and W. Melo: "Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components". In *Proceedings of the 19th International Conference on Software Engineering*, pages 282-291, 1997.

[7] E. Bennett, R. Alpert, and A. Goldstein: "Communications Through Limited Response Questioning". In *Public Opinion Quarterly*, 18:303-308, 1954.

[8] I. Bhandari, M. Halliday, J. Chaar, R. Chillarege, K. Jones, J. Atkinson, C. Lepori-Costello, P. Jasper, E. Tarver, C. Lewis, and M. Yonezawa: "In-process Improvement Through Defect Data Interpretation". In *IBM Systems Journal*, 33(1): 182-214, 1994.

[9] D. Bisant and J. Lyle: "A Two-Person Inspection Method to Improve Programming Productivity". In *IEEE Transactions on Software Engineering*, 15(10):1294-1304, 1989.

[10] D. Card: "Learning from our Mistakes with Defect Causal Analysis". In *IEEE Software*, pages 56-63, January-February 1998.

[11] J. Chaar, M. Halliday, I. Bhandari, and R. Chillarege: "In-Process Evaluation for Software Inspection and Test". In *IEEE Transactions on Software Engineering*, 19(11):1055-1070, November 1993.

[12] R. Chillarege, W-L Kao, and R. Condit: "Defect Type and its Impact on the Growth Curve". In *Proceedings of the 13th International Conference on Software Engineering*, pages 246-255, 1991.

[13] R. Chillarege, I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray, and M-Y. Wong: "Orthogonal Defect Classification - A Concept for In-Process Measurements". In *IEEE Transactions on Software Engineering*, 18(11):943-956, November 1992.

[14] R. Chillarege and S. Biyani: "Identifying Risk Using ODC Based Growth Models". In *Proceedings of the Fifth International Symposium on Software Reliability Engineering*, pages 282-288, 1994.

[15] D. Cicchetti: “Testing the Normal Approximation and Minimal Sample Size Requirements of Weighted Kappa When the Number of Categories is Large”. In *Applied Psychological Measurement*, 5(1):101-104, 1981.

[16] D. Cicchetti and J. Fleiss: “Comparison of the Null Distributions of Weighted Kappa and the C Ordinal Statistic”. In *Applied Psychological Measurement*, 1(2): 195-201, 1977.

[17] D. Cicchetti, D. Showalter, and P. Tyrer: “The Effect of Number of Rating Scale Categories on Levels of Interrater Reliability: A Monte Carlo Investigation”. In *Applied Psychological Measurement*, 9(1):31-36, 1985.

[18] J. Cohen: "A Coefficient of Agreement for Nominal Scales". In *Educational and Psychological Measurement*, 20:37-46, 1960.

[19] J. Cohen: “Weighted Kappa: Nominal Scale Agreement with Provision for Scaled Agreement or Partial Credit”. In *Psychological Bulletin*, 70:213-220, 1968.

[20] K. El Emam: *Benchmarking Kappa for Software Process Assessment Reliability Studies*. Technical Report ISERN-98-02, International Software Engineering Research Network, 1998.

[21] J. Fleiss: *Statistical Methods for Rates and Proportions*, John Wiley & Sons, 1981.

[22] J. Fleiss: "Measuring Nominal Scale Agreement Among Many Raters". In *Psychological Bulletin*, 76(5):378-382, 1971.

[23] J. Fleiss, J. Cohen, and B. Everitt: "Large Sample Standard Errors of Kappa and Weighted Kappa". In *Psychological Bulletin*, 72(5):323-327, 1969.

[24] W. Florac: *Software Quality Measurement: A Framework for Counting Problems and Defects*. Software Engineering Institute, Technical Report CMU/SEI-92-TR-22, 1992.

[25] L. Gordis: *Epidemiology*. W. B. Saunders, 1996.

[26] D. Hartmann: “Considerations in the Choice of Interobserver Reliability Estimates”. In *Journal of Applied behavior Analysis*, 10:103-116, 1977.

[27] W. Humphrey: *A Discipline for Software Engineering*. Addison-Wesley, 1995.

[28] IEEE Computer Society: *IEEE Standard Classification for Software Anomalies*, IEEE Standard 1044-1993, 1993.

[29] IEEE Computer Society: *IEEE Guide to Classification for Software Anomalies*, IEEE Standard 1044.1-1995, 1995.

[30] K. Krippendorff: "Bivariate Agreement Coefficients for Reliability of Data". In E. Borgatta (ed.): *Sociological Methodology*, pp. 139-150, Jossey-Bass, 1970.

[31] S. Kusumoto, A. Chimura, T. Kikuno, K. Matsumoto, and Y. Mohri: "A Promising Approach to Two-Person Software Review in Educational Environment". In *Journal of Systems and Software*, Vol. 40, No 2, pp. 115-123, 1998.

[32] Y. Mashiko and V. Basili: "Using the GQM Paradigm to Investigate Influential Factors for Software Process Improvement". In *Journal of Systems and Software*, 36:17-32, 1997.

[33] C. Mooney: *Monte Carlo Simulation*. Sage Publications, 1997.

[34] J. Rice: *Mathematical Statistics and Data Analysis*. Duxbury Press, 1987.

[35] Software Engineering Laboratory: *Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules*. NASA/GSFC Software Engineering Laboratory Technical Report SEL-91-001, 1991.

[36] Software Engineering Laboratory: *Software Measurement Guidebook*. NASA/GSFC Software Engineering Laboratory Technical Report SEL-94-002, 1994.

[37] H. Suen and P. Lee: "The Effects of the Use of Percentage Agreement on Behavioral Observation Reliabilities: A Reassessment". In *Journal of Psychopathology and Behavioral Assessment*, 7(3):221-234, 1985.

[38] U. Umesh, R. Peterson, and M. Sauber: "Interjudge Agreement and the Maximum Value of Kappa". In *Educational and Psychological Measurement*, 49:835-850, 1989.

[39] S. Vander Wiel and L. Votta: "Assessing Software Designs Using Capture-Recapture Methods". In *IEEE Transactions on Software Engineering*, 19: 1045-1054, 1993.

[40] R. Zwick: "Another Look at Interrater Agreement". In *Psychological Bulletin*, 103(3):374-378, 1988.

Appendix

In this appendix we present an evaluation of the Type I error rates (the rate at which the null hypothesis is incorrectly rejected) when determining the statistical significance of κ. The critical ratio for testing the null hypothesis H_0: $\kappa=0$, versus the alternative H_A: $\kappa \neq 0$ was derived by Fleiss et al. [23]. This critical ratio approximates the standard normal distribution. In the same article, a critical ratio for the weighted version of κ (see [19]) was derived.

Previous studies evaluated this approximation. Cicchetti and Fleiss [16] conducted a Monte Carlo simulation to compare the critical ratio to the standard normal distribution in terms of mean, variance, skewness, and kurtosis, and in terms of selected one and two tailed areas. They concluded that the approximation was valid when the number of observations was as low as $2k^2$. This study was done for $3 \leq k \leq 7$. A followup study was performed for $8 \leq k \leq 10$, and came up with the same conclusions [15].

Simulated Data Set	Type I Error rate		
	α = 0.10	α = 0.05	α = 0.025
Data Set 1	0.106	0.049	0.029
Data Set 2	0.094	0.044	0.023

Table 8: Results of evaluating the Type I error rates for one-tailed tests using parameters similar to those from our data sets for three different alpha levels.

These simulations were not extended to k=11, which is the case in our study (we have 11 defect classes, although for some classes no defects were detected). Furthermore, they used a weighted version of κ, and the marginal proportions they simulated were not necessarily similar to the ones in our study.

Therefore, we conducted a Monte Carlo simulation to determine the Type I error rates for our observed marginal proportions and our sample sizes using the critical ratio for unweighted Kappa. Two simulations were run, each one using the parameters of one of our data sets. Each simulation consisted of 1000 runs. The simulation generates tables with the expected proportions equal to our observed proportions, and the row and column variables are independent (equivalent to the null hypothesis above), with the total number of observations equal to our sample sizes.

The results are summarized in Table 8. For the second data set, one can conclude that our statistical results using the unweighted Kappa critical ratio are conservative. The Type I error rates tend to be slightly lower than the nominal alpha level, but the differences are not marked. For the first data set, there is no systematic conservatism, but the differences are not marked. Therefore, even though our sample sizes are lower than those recommended in [15][16], there is confidence that, for our data sets using unweighted Kappa, the critical ratio used provides valid results.