The Repeatability of Code Defect Classifications

Khaled El Emam and Isabella Wieczorek
Fraunhofer Institute for Experimental Software Engineering
Sauerwiesen 6
D-67661 Kaiserslautern
Germany
{elemam , wieczo}@iese.fhg.de

International Software Engineering Research Network Technical Report ISERN-98-09

The Repeatability of Code Defect Classifications

Khaled El Emam and 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 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 [29]. The relationship between defect types and other variables such as whether a module was new or modified [4], and the cost of correction [26][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 [32], 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 [24]) 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 misleading results [17].

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 [30], 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 [20], and an IEEE Standard for defect classification that presents a number of different classification schemes [23]. 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

[1] We use the terms "reliability" and "repeatability" interchangeably in this paper.

the ODC scheme [11][13], which has also been incoporated in the SEI's Personal Software Process [22]. 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 [9], Kusumoto et. al. [25] 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 to remove 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 the 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 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 perform 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 [25] 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.

Steps	Roles	Person
Planning	Moderator	A
Preparation	Inspector	A and B
Meeting	Moderator Inspector Author	A A and B B
Correction	Author	B
Follow-Up	Moderator Author	A B

Table 1: Steps of the inspection process.

All defects considered for this study were found by two different pairs of inspectors (total of three different inspectors), giving two different data sets, one for each pair.

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 [22]. 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. Both of these were also removed from the defect classification suggested in [22]. A number of defect classes were also added. Namely, these were “Data” (also included in [22]), “Environment”, “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.

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.

2.3 Data Analysis

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

		Inspector A				
		$Class_1$	**$Class_2$**		**$Class_k$**	
Inspector B	**$Class_1$**	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 k x k table for representing *proportions* of defect classifications made by two inspectors.

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.

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:

$$P_e = \frac{1}{k} \qquad \textbf{Eqn 1}$$

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

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

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 assessors).

A general form for agreement coefficients[2] is [33]:

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

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.

[2] It should be noted that "agreement" is different from "association". For the ratings from two inspectors to agree, the ratings must fall in the same defect class. For the ratings from two teams 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.

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 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) [15].

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 [18]. Umesh et al. [31] 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][21].

Extensive use in various disciplines means that guidelines have been developed for interpreting a particular statistic. In [16] 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 0.4 indicate inadequate agreement. Values above 0.6 indicate good agreement, and values above 0.75 indicate excellent agreement.

We can also test the null hypothesis that the observed amount of agreement (or greater) could have occurred by chance (i.e., the inspectors classifying at random according to their marginal proportions). The standard error of Kappa has been derived by Fleiss et al. [19] and can be used for hypothesis testing. Since we have two data sets in our study, all statistical tests are conducted at a Bonferonni adjusted alpha level (see [28]). 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 both inspectors find the same defects that they both classify as "Function", and that "Function" type defects constitute the majority of defects found by both inspectors. This means that there is high agreement on classifying "Function" defects. Because Kappa can be considered as a weighted average of the agreement on each class [17], 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 would be highly inflated compared to the value that would be obtained if we used all logged defects.

In general, if the calculated Kappa is inflated and has a low value, this means that the actual repeatability of defect classifications for inspections is not going to be good enough, and we can conclude that the classification scheme is not reliable. If calculated Kappa is deflated and has a high value then it sets a lower bound on the reliability of defect classification during inspections, and we can conclude that the defect classification scheme is reliable.

It is therefore prudent to perform a sensitivity analysis to determine whether 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* all the defects that were not actually found by inspector A were classified by A. We do this through a Monte Carlo simulation [27]. To construct this simulation we have to define how an inspector would classify defects that s/he did not actually find.

To explain the simulation, we take inspector B as an example. We refer to the hypothetical data in Table 4 for this discussion. This hypothetical table assumes that our defect classification scheme has only three classes: "X", "Y", and "Z". It shows each inspector's classification and the final logged classification.

During the inspection meeting, inspector B is presented with the defects that s/he did not find, and these are classified by both inspectors as type "X" (defect number

1). By looking at the classification of defects that B did find, we can determine the 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”, and s/he classifies the remaining 50% as “X”. 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. This is a worst 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), with one exception below.

	Inspector A	**Inspector B**	**Logged**	**Simulated Distribution for Inspector B**
1.	X		X	Discrete((“Y”,0.50),(“X”,0.50))
2.	X	Y	X	N/A
3.	Y	X	X	
4.	Y		Y	Discrete((“Y”,0.33),(“X”,0.66))
5.		Y	Y	N/A
6.		Y	X	N/A
7.		X	X	N/A
8.		X	Y	N/A
9.		X	Y	N/A
10.	Z		Z	Discrete((“Y”,0.43),(“X”,0.57))

Table 4: Hypothetical data collected and simulated distribution for inspector B.

If B never found a defect that was logged during the meeting, for example defects of type “Z”, then we make a worst case 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 (see defect number 10).

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 0.6 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 the inspectors specialize in finding different types of defects.

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

Table 5: Summary of the two data sets.

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 reflects the specialization of the inspectors.

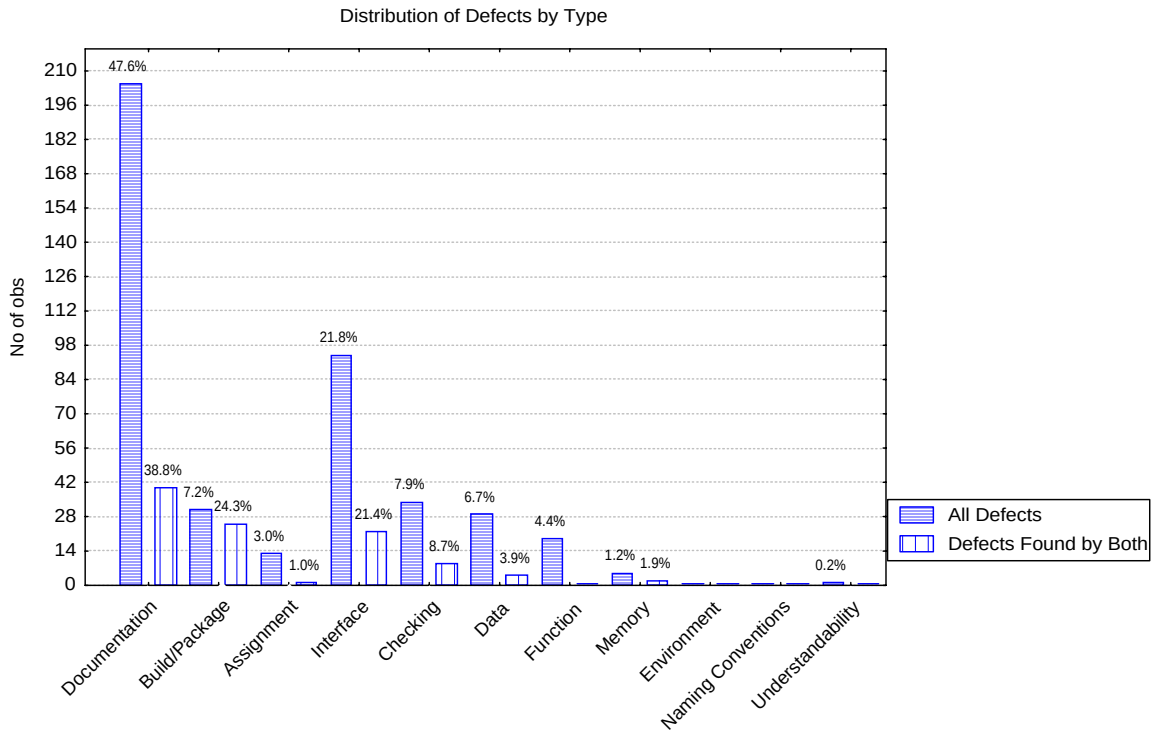

Figure 1: Distribution of defects for data set 1.

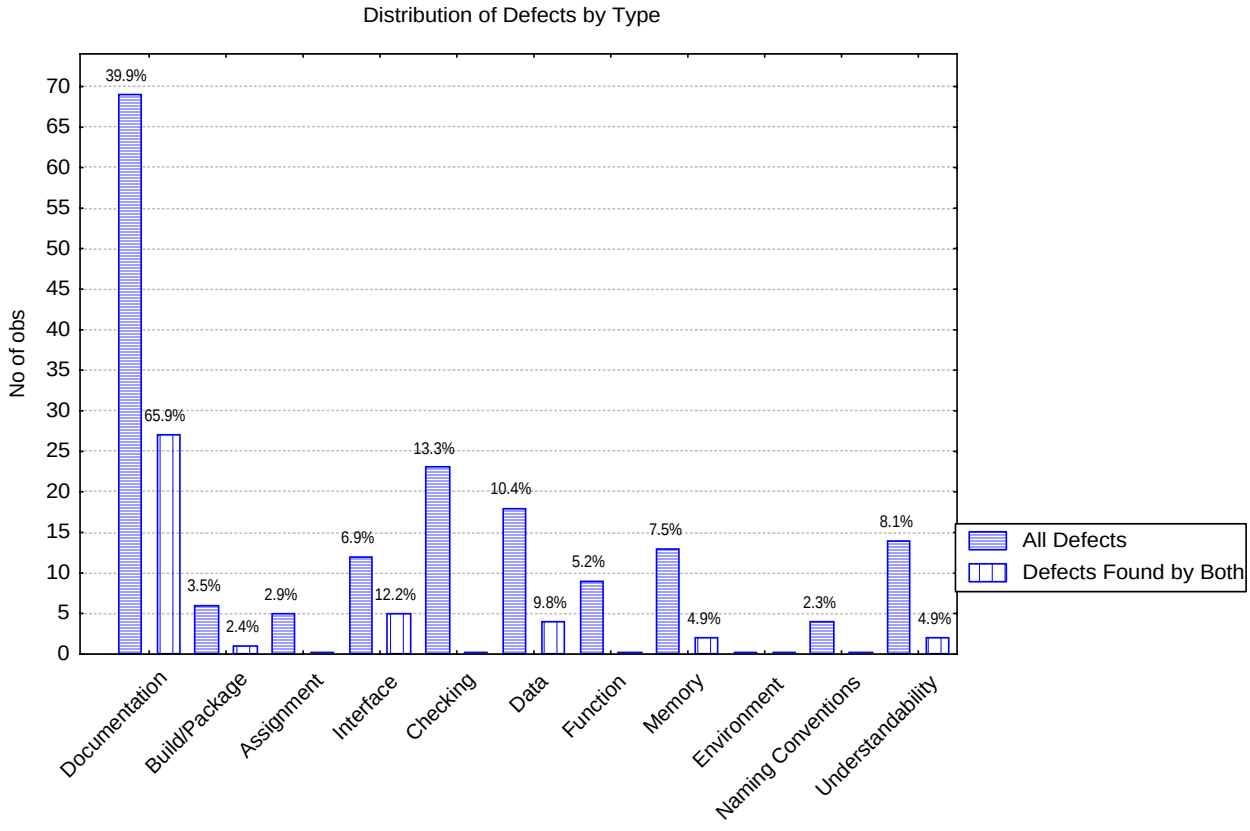

Figure 2: Distribution of defects for data set 2.

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 0.6 indicating good agreement, with the second data set above 0.75 indicating excellent agreement. Both values are statistically significant.

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).

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 a slight inflation of calculated Kappa since most values are greater than the calculated 0.66, and the mean is 0.80. The cummulative density is shown in Figure 4. As can be seen, the proportion of times that the values of Kappa are at 0.6 or lower is close to zero. This indicates that under the worst case assumptions made during the sensitivity analysis, the extent of agreement will still be good almost all of the time.

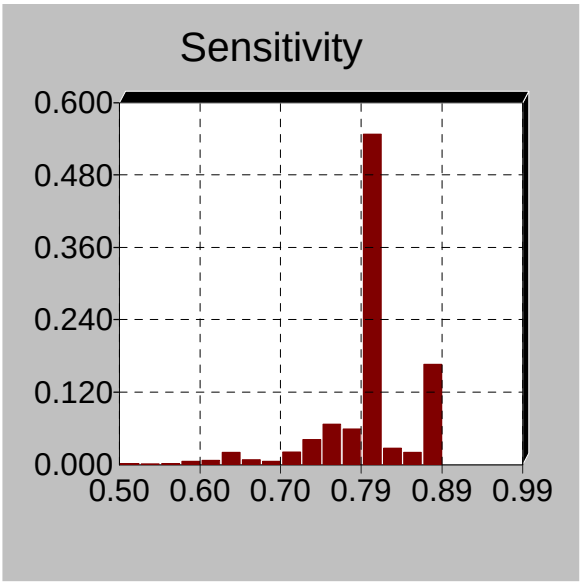

Figure 3: Sensitivity of Kappa for data set 1 (mean 0.80). The y-axis is the frequency from 1000 iterations, and the x-axis is the value of Kappa.

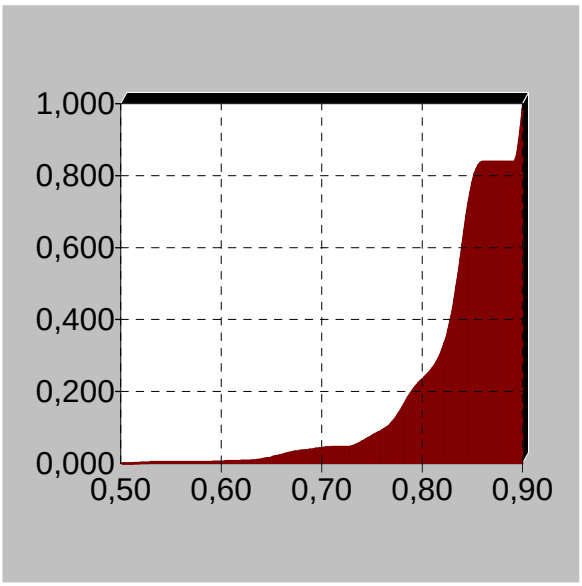

Figure 4: Kappa Cumulative Density Function for data set 1. The y-axis represents the probability of obtaining an x value equal to or smaller than the value on the x-axis, and the x-axis is Kappa.

The frequency distribution of simulated Kappa for the second data set in Figure 5, and the cummulative density in Figure 6. For data set 2 there is a slight deflation of calculated 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 0.75, but very rarely if ever does it go below 0.6. This is evident in Figure 6 where the proportion of times the simulated Kappa is at 0.6 or less is almost zero.

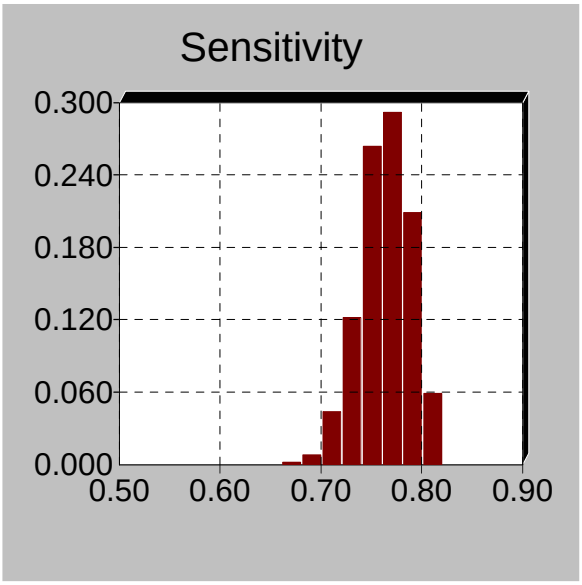

Figure 5: Sensitiviy of Kappa for data set 2 (mean 0.76). The y-axis is the frequency from 1000 iterations, and the x-axis is the value of Kappa.

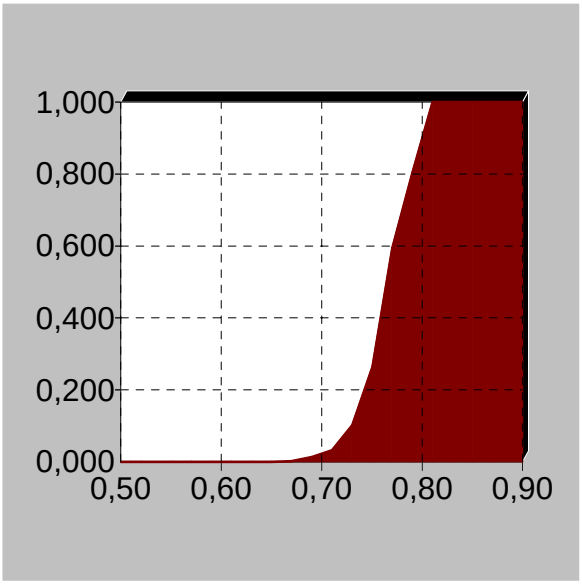

Figure 6: Kappa Cumulative Density Function for data set 2. The y-axis represents the probability of obtaining an x value equal to or smaller than the value on the x-axis, and the x-axis is Kappa.

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.

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).

Based on this result, it is suggested that the "Data" defect class be either refined further to clarify its distingushing 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.

Our study was conducted for one type of document, code, for 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.

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.

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 in this approaches are based on the premise that defect classification is repetable. However, this assumption has not been systematically investigated thus far. Furthermore, while perforimg 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.

5. 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 for reviewing an earlier version of this paper.

6. 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*, vol. 33, no. 1, pp. 182--214, 1994.

[9] D. Bisant and J. Lyle: "A Two-Person Inspection Method to Improve Programming Productivity". In *IEEE Transactions on Software Engineering*, Vol.15, No. 10, Oct. 1989, pp. 1294-1304.

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

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

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

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

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

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

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

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

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

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

[25] 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.

[26] 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.

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

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

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

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

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

[32] S. Vander Wiel and L. Votta: “Assesing Software Designs Using Capture-Recapture Methods”. In *IEEE Transactions on Software Engineering*, vol. 19: 1045--1054, 1993.

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