Defect Detection Effectiveness and Product Quality in Global Software Development

Tihana Galinac Grbac[1] and Darko Huljenić[2]

[1] Faculty of Engineering, University of Rijeka, Vukovarska 58, HR-51000 Rijeka, Croatia
tihana.galinac@riteh.hr
[2] Ericsson Nikola Tesla, Krapinska 45, HR-10000 Zagreb, Croatia
darko.huljenic@ericsson.com

Abstract. Global software development (GSD) has become a common practice in the software development industry. The main challenge organizations have to overcome is to minimize the effect of organizational diversity on the effectiveness of their GSD collaboration. The objective of this study is to understand the differences in the defect detection effectiveness among different organizations involved into the same GSD project, and how these differences, if any, are reflected on the delivered product quality. The case study is undertaken in a GSD project at Ericsson corporation involving nine organizations that are commonly developing a software product for telecommunication exchanges. Comparing the effectiveness of defect detection on the sample of 216 software units developed by nine organizations, it turns out that there is statistically significant difference between defect detection effectiveness among organizations. Moreover, the defect density serves better as a measure of defect detection effectiveness than as a measure of the product quality.

Keywords: Global software development, defect detection effectiveness, defect density, software quality.

1 Introduction

Global Software Development (GSD) is becoming a common practice in the modern software industry, involving software development teams from the organizations that are distributed around the globe to develop a common software product. The main drivers for globalization of software development is in cost benefits, entrance into global market and access to a large multi-skilled resource pool. On the other hand, the main challenge the organizations working in the GSD environment have to overcome is to minimize the influence of its diversity onto GSD project success. The software development project success is highly dependent on the software project team effectiveness in executing the project processes to achieve the project goals. The main identified barriers for teams in GSD environment are coordination, communication and control [11].

The researchers and practitioners effort have been focused on defining general processes and providing guidelines to overcome these barriers. Besides having

D. Caivano et al. (Eds.): PROFES 2011, LNCS 6759, pp. 113–127, 2011.
© Springer-Verlag Berlin Heidelberg 2011

implemented the general processes that are highly supported with collaboration tools and following guidelines for distributed work, the main concern is whether the impact of the organizational distribution is significant or not. The objective of this study is to explore differences in the defect detection effectiveness among different organizations involved into the same GSD project and how these differences, if any, are reflected on the delivered product quality.

The rest of the paper is organized as follows. In Section 2 the metrics used for evaluating the defect detection process is introduced. In Section 3 the related work is reviewed. Section 4 describes the research context and methods used in the case study. The results of the study are presented in Section 5. Finally, in Section 6 the results are discussed, and we conclude the paper in Section 7.

2 Metrics

There exists a variety of metrics defined for measuring software quality [9], [10], [13]. Still, the most dominant metric that is used in the empirical studies for evaluation of software quality is the number of detected defects [3], [5]. The size of software, on which the number of defects in reported, is used for the comparison purposes. The defect density is defined in [12] as the ratio of the number of defects found and the software size involved into defect detection. The aim of defect detection activities is to deliver software product with zero remaining defect density. Therefore, the defect density is also considered as a measure for defect detection effectiveness [17].

The defect detection effectiveness may be used as a process control measure. Comparing this measure with the average from the history projects or with the project goal, one could bring decision about additional investment into defect detection process, as suggested for example in [8], [7] for the purpose of software inspection. The problem with defect density as a defect detection effectiveness measure lies in the fact that it is hard to distinguish if the number of defects identified by defect detection process is due to bad design, coding or good defect detection process. A number of defect injection and defect detection factors have been identified in [21]. Nevertheless, a number of studies have used defect density measure for the purpose of comparison of defect detection technique effectiveness in a given environment. The defect seeding technique is commonly used in these cases to increase the reliability of the study. Furthermore, the defect density considered "in time", that is as a function of invested testing effort, is used in reliability growth modelling [14]. The control of defect detection process is based on the trend of the defect density curve.

In this study we are neither interested into reasons for the amount of defects detected, nor to control the defect detection process. Instead we are analyzing the differences in the effectiveness of completed defect detection activities among distributed organizations that were supposed to apply the same general defect detection process for GSD project. Therefore, in this study the defect density was used as measure to determine differences in defect detection effectiveness among organizations involved into GSD and influence of the organizational diversity on the defect detection effectiveness and the product quality.

3 Related Work

One of the main motivator for the organizational movement into global software development is in its cost benefits. This implies access to larger and well–skilled resource pool at lower cost [4]. On the other hand, dislocation of development teams has been identified as the main challenge for effective communication, coordination and control that may have significant impact on the project performance and software quality.

A number of studies have questioned the performance ability of GSD along with the resulted software product quality. The empirical evaluation of GSD effect is hard to isolate from other effects, and requires well defined and consistent measurement system in all organizations involved into GSD. This is hard to achieve in practice resulting with limited related work in this area.

In [3] the post–release defects and software unit attributes are compared between software units developed by distributed versus collocated teams. Comparison of mean values for the number of post–release defects detected per software unit, has resulted with slightly more, but statistically significant, defects in GSD. Moreover, if the number of developers is included into analysis, the significance of the conclusion that larger number of post–release defects is affected by GSD becomes even smaller. Another analysis performed in [3], compared software unit attributes, such as code churn, complexity, dependency, and test coverage, measured per software unit. It turns out that there is no significant difference between software units developed in GSD and collocated environment.

A number of metrics have been evaluated as possible predictors of software unit defect–proneness. The aim of these is to control and predict software quality, and thus, better direct future verification efforts. In [15] the organizational complexity is proposed as a predictor of software quality, where the software quality is measured as the amount of post–release defects. The metrics scheme for organizational complexity qualification, as proposed in [15], consists of eight measures that are representing organizational distance of the developers, the number of developers involved into software unit development, the developers level of multi–tasking, and the amount of organizational involvement into software unit change. Using the proposed metrics scheme on the data obtained from the Windows Vista development resulted with the conclusion that such organizational complexity measure is a statistically significant predictor of the defect–prone software units. Moreover, the results showed that the suggested metrics scheme is better predictor for post–release defects than traditionally used metrics such as static code attributes, coverage, and pre–release defects.

Another approach to evaluating the impact of GSD on the software quality is by using the process maturity levels as defined in [6]. The study performed in [5] analyzed the impact of the process maturity level on the software quality, where the software quality is measured as the number of defects reported during the integration and system testing for each software unit. The analysis resulted with the conclusion that the process maturity level and the level of distribution have significant impact on the quality of software units. Moreover, the process

maturity impact on software quality becomes more significant as development becomes more distributed.

The code size, level of distribution and pre–release defect density are used to evaluate the experiences working within GSD in [2]. Comparing the pre–release defect densities for several completed projects working in GSD and the average defect density reported for US industry (2.6 defects per 1000 Lines of Code) the paper concludes that GSD has not increased the defect density.

4 A Case Study

In this section we define and explain the context of the research study, research questions, methods and strategy used for the data collection and analysis.

4.1 Context of the Study

The context of the study is the defect detection process used in globally distributed software development (GSD) within Ericsson corporation.

The software product is developed for telecommunication exchanges, in particular, the Mobile Switching Center (MSC) node that is one of the network nodes within next generation core telecommunication network. The software product is developed in the product line [1] fashion, in which the product releases are following the Core Network evolution strategy as prescribed by [19]. A product release is the outcome of a GSD project. The smallest self contained administrative unit of the software product is called a software unit. The size of a software unit is small enough so it can be understood and managed by humans. The software product we considered in this paper consisted of 216 software units that were impacted by some modifications. The total volume of impacted software units was $1.5\,M$ volume of code with included modification of over $400\,k$ volume of code. The volume of code is a measure obtained as a sum of the kilo programming statements, the number of kilo Assembler instructions multiplied by factor 0.3, and the amount of data storage. Each software unit is in responsibility of a software developer as suggested in [16], and according to its affiliation the software units are assigned to organizations involved into GSD project.

The software development process is an evolved version of the V–model with incremental delivery strategy and feature based development. Waterfall sequence of development phases such as analysis, design, coding, testing is kept on the feature level. Moreover, the classical project management model based on Tollgate concept [18] is used but with rather flexible tollgate criteria. For example, a requirement for passing a project Tollgate between two consecutive project phases could be that at least 60% of the features is already in the later phase. The process evolves from project to project, as best practices and lessons learned are incorporated into the new revision of software development process that is used in further software development projects. Since the process is used in GSD for years, it is very well supported by a standard toolbox, which is also updated with all process improvements. Collaboration tools are also incorporated into the standard toolbox, and the majority of existing software development tools are adapted for collaboration purposes.

The software development is a case of global software development where multiple partner organizations are involved into Software Design and Software Verification projects. The software design project is performed by nine Software Design (SD) organizations that are globally distributed and develops together the same software product (software for the MSC node). The software development work is divided among these SD organizations following the product ownership criteria, which follow the software product architecture, as suggested in [16]. All SD organizations participate in the same software development process phases, such as analysis, design, coding, and part of testing that are tailored for GSD.

All defect detection activities performed by SD organizations will be called early defect detection (EDD). The EDD includes software code inspections, code reviews, basic test, and those function tests that are performed in the simulated environment, simulating the rest of the software product. In the study, we compare the EDD effectiveness of different SD organizations that participate in the same software design project. They are supposed to follow the same early defect detection process prescribed by the GSD project.

The software verification project is performed by the Network Integration and Verification (NIV) organizations that are also globally distributed. However, unlike SD organizations, their work is divided mostly with respect to the specific market/customer. Since our study is concentrated only at two customer references, that are related to the basic node system test and basic integration and verification test, only one NIV organization is involved.

All defect detection activities performed by the NIV organization will be referred to as late defect detection (LDD). The LDD activities are performed on the software units delivered from the SD organizations. The whole software product is under test, without any simulations within the software product. The processes followed by the NIV organization are also standardized within their organization and they evolve with projects. In the study we compare the LDD effectiveness between software units developed by different SD organizations. The NIV organization is completely independent from SD organization, with separated resources. Nevertheless, as a best practice it is customary to borrow resources, but only as the support personnel during the hand–over processes. The results of the NIV organization, when analyzed between different SD organizations, could be a useful indicator of the impact of GSD organizations distribution on the delivered product quality. The LDD effectiveness could be also a measure of product quality.

4.2 Research Hypotheses

The main objective of our research study was to understand how the organizational diversity influences the early defect detection effectiveness, the late defect detection effectiveness and the software product quality. As already explained in Section 2, the defect density is used as the metric for this investigation. Therefore we selected the following research hypotheses:

HA_0 : The mean values of defect density in early defect detection for samples of different SD organizations within the same GSD project are equal.

HB_0 : The mean value of defect density in late defect detection for samples coming from different SD organizations within the same GSD project are equal.

The alternative hypotheses are that the mean values considered in the corresponding null hypothesis are not all equal.

4.3 Data Collection

The selected software design and software verification projects for which data collection is performed are one of the history projects where all software development activities are finalized. All data were collected per software unit as presented in Table 1 and are associated to the relevant SD organizations.

For each software unit, the main data collection questions, that are related to the research hypotheses, are as follows:

- What is the volume and modified volume of the software unit?
- What is the number of defects detected in the software unit during early defect detection process?
- What is the number of defects detected in the software unit during late defect detection process?
- Which SD organization is responsible for the software unit?

The random variables associated to these questions are listed in Table 1.

The measurements for all random variables in Table 1 except the one counting the defects in LDD were collected from the Quality Report documents that are stored in the project repository through the standard Ericsson Clear Case tool. This is a mandatory document required at TollGate3, at which the software product is handed–over from the software design project to the software verification project. All software units that were included into Quality Report documents, and that have reported modification volume, were included into this data collection procedure.

In order to verify the reliability of the collected data for random variable counting the defects in EDD, the data were additionally collected from the corporate Change Notification database, which is Ericsson's official database for reporting defects detected during early defect detection process. The software units in which data inconsistency is identified were removed from the sample.

Table 1. Measured random variables

Name	Description
SWU Size	Volume of respective SWU
SWU Modification Size	Modified volume of respective SWU
SWU Defects in EDD	Count of defects detected in EDD per SWU
SWU Defects in LDD	Count of defects detected in LDD per SWU

The measurements for the random variable that counts defects in LDD were collected from the corporate Modification Handling database. For all software units in which the modification is made in the considered software design project (according to the Quality Reports mentioned above), the number of defects reported in the database were counted. Note that we considered only defects that were analyzed and answered with correction, and ignore all enhancements and market adaptations. Moreover, our analysis was limited only on defects reported by the NIV project with two customer references, that are related to the node system test and basic configuration of the network test. Analyzing only the customer that is common for all SD organizations, we secure the same treatment in late defect detection process for all observed SD organizations. Note that, due to the product line development, the same software unit can be present in several product releases, as well as a number of node configurations intended to serve in several markets and customer sites. So, the number of different customers that can be involved into modification handling vary among the software units, and it is not necessarily that only one customer is involved into modification handling process of all SWUs that were treated by SD organizations within selected GSD project.

4.4 Threats to Validity

Since we perform an empirical case study, it suffers from a number of threats to validity. According to [20] there are four different aspects of validity: internal, external, conclusion, and construct validity.

In our study we identified the following threats to validity. The threat to external validity is the fact that the study is performed only within one GSD project and that the study was not performed on the random sample of software units. However, the company where the study is performed is ISO certified and six participating organizations are on the CMM level 2 and three on the CMM level 3. Hence, it is quite possible that the findings generalize to such organizations.

The stated conclusions are based on the data collection from the well defined tools that are used for a long time by the organization's personnel. Eventual bias caused by data collection in the considered critical case is eliminated by using two sources of data collection as already explained in Section 4.3. Moreover, large sample of the data collected increases the reliability of the study.

The construct validity examines whether the selected measures are representative for the considered research questions. Effectiveness is a common measure used for the evaluation of defect detection techniques. In our case, we are comparing the effectiveness of early defect detection, performed by different organizations that are supposed to follow the same process. We used them as a measure of organizational impact. On the other hand, the defect detection process is not so strict and may involve various defect detection techniques depending on the defined defect detection strategy. So, the differences in the early defect detection process may be caused by differences in the chosen detection strategy, and not by the organizational diversity. In the case of the study, the defect detection strategy in early phases is defined independently for each SWU by the

development team. Thus, taking a large sample of software units solves this issue. Also, the effect of the software unit difficulty is assumed to be minimized by takeing a sample of software units large enough per organization.

5 Results

In this section, the results of the statistical analysis of the collected data regarding the defect detection process are explained. In the first subsection the descriptive statistics is presented for all collected and derived variables used in the study. In the rest of the section we deal only with the defect density.

5.1 Descriptive Statistics

The main concern of the statistical analysis is the random variable measuring the defect density as defined in Section 2. We consider separately the defect density of early and late defect detection denoted, respectively, by X_0 and X_1. For both random variables X_0 and X_1, as explained in Section 4.3, we have collected nine samples, one for each of nine SD organizations involved in the software design and maintenance projects. These SD organizations and associated samples are denoted by upper case letters A to I.

Table 2. Descriptive statistics for random variable X_0

SD org.	N	Mean	Std. dev.	95% C.I.	Min.	Median	Max.
A	13	11.740	12.893	[3.949, 19.531]	1.177	7.485	45.281
B	9	10.441	7.616	[4.587, 16.295]	3.333	6.536	25.779
C	33	17.279	19.793	[10.261, 24.298]	0.799	8.972	82.192
D	17	16.211	14.125	[8.949, 23.473]	2.515	12.176	47.393
E	34	8.088	6.512	[5.816, 10.360]	0.528	5.855	28.221
F	5	10.275	5.231	[3.780, 16.771]	5.199	8.523	18.692
G	45	10.478	12.485	[6.727, 14.229]	0.128	5.583	62.500
H	11	18.083	24.104	[1.890, 34.276]	1.916	9.332	86.752
I	49	22.957	27.955	[14.927, 30.987]	1.268	12.976	151.163

The descriptive statistics of the defect density random variables X_0 and X_1 measured per SWUs in each of nine SD organizations are summarized in Table 2 for X_0 and in Table 3 for X_1. In the tables the column labels "N", "Mean", "Std. dev.", "95% C.I.", "Max.", "Median", and "Min" stand for the number of observations, mean value, standard deviation, 95% confidence interval, maximum, median, and minimum of the sample, respectively.

Observe that the sample measured in SD organization F consists of only five SWU. Since this sample size is insufficient for a significant statistical analysis, we removed it from further analysis. In the remaining samples we removed the outliers. The descriptive statistics given in the tables is for the samples in which the outliers are removed.

Table 3. Descriptive statistics for random variable X_1

SD org.	N	Mean	Std. dev.	95% C.I.	Min.	Median	Max.
A	13	14.547	13.838	[6.185, 22.909]	0.657	9.543	41.147
B	9	16.218	17.496	[2.769, 29.666]	1.669	9.757	57.143
C	33	8.855	8.377	[5.885, 11.825]	1.268	6.247	37.500
D	17	10.228	12.960	[3.564, 16.891]	0.846	6.827	47.847
E	34	18.241	22.179	[10.502, 25.979]	2.253	11.389	114.755
F	5	8.186	4.065	[3.138, 13.233]	4.422	6.490	14.521
G	45	12.960	19.269	[7.171, 18.748]	0.385	4.785	85.386
H	11	17.017	18.275	[4.740, 29.294]	2.874	12.273	65.476
I	49	17.399	21.857	[11.121, 23.677]	0.856	12.622	142.857

5.2 Normality Tests

The parametric hypothesis tests are more robust and reliable than non–parametric tests. However, a general assumption for all parametric tests is that the analyzed data samples are normally distributed. Hence, in order to justify the use of parametric tests for our research hypotheses, we need to check whether the samples follow the normal distribution.

Table 4. Normality test for transformed random variable X_0

SD org.	Kolmogorov–Smirnov		Shapiro–Wilk	
	d statistic	p–value	W statistic	p–value
A	0.165	> 0.20	0.961	0.768
B	0.178	> 0.20	0.915	0.355
C	0.129	> 0.20	0.954	0.173
D	0.096	> 0.20	0.958	0.600
E	0.096	> 0.20	0.963	0.302
F	0.165	> 0.20	0.983	0.950
G	0.078	> 0.20	0.974	0.390
H	0.154	> 0.20	0.968	0.869
I	0.091	> 0.20	0.982	0.632

The standard tests for normality are the Kolmogorov–Smirnov and Shapiro–Wilk test. We apply both tests to the nine samples for each of the defect density random variables X_0 and X_1. However, it turns out that none of the samples follows the normal distribution. This result was expected, since several authors investigating the defect density in software defect detection process already reported that the defect detection random variable follows the log–normal distribution. Hence, we transformed all the samples by applying the natural logarithm, and applied the Kolmogorov–Smirnov and Shapiro–Wilk test to the transformed

samples. The results of the normality tests on the transformed samples of random variables X_0 and X_1 are given in Table 4 and Table 5, respectively. From the tables we read that all the tests are significant (p–value > 0.05), which means that all the transformed samples follow the normal distribution.

Table 5. Normality test for transformed random variable X_1

SD org.	Kolmogorov–Smirnov		Shapiro–Wilk	
	d statistic	p–value	W statistic	p–value
A	0.153	> 0.20	0.919	0.240
B	0.145	> 0.20	0.969	0.886
C	0.085	> 0.20	0.964	0.331
D	0.115	> 0.20	0.973	0.870
E	0.112	> 0.20	0.946	0.094
F	0.215	> 0.20	0.962	0.819
G	0.098	> 0.20	0.968	0.253
H	0.182	> 0.20	0.941	0.532
I	0.096	> 0.20	0.964	0.144

As a consequence of these conclusions, we are free to apply parametric statistics for the further analysis.

5.3 Hypothesis Testing

The research hypotheses are stated in Section 4.2. In order to test these research hypothesis we applied one–way ANOVA on transformed random variables X_0 and X_1. The assumptions for applying ANOVA is that the dependent variables are normally distributed and that the groups have approximately equal variance on the dependent variable. The normality of the transformed samples was confirmed in Section 5.2. To verify the homogeneity of variances of the transformed samples we use Levene's and the Brown–Forsythe test. The results of these tests are given in Table 6. It turns out that for X_0 both tests show that the assumption of homogeneity of variances between samples should be rejected (p–value < 0.05). On the other hand, for X_1 both test confirm the homogeneity of variances. Thus, the assumptions for applying ANOVA are satisfied only for random variable X_1.

Consider first the random variable X_0. Since the assumptions of ANOVA are not satisfied, we performed the non–parametric tests for equality of means to test the research hypothesis HA_0. These tests are the Kruskal–Wallis ANOVA and median test. The results are given in the first row of Table 7. The conclusions of the two tests are not consistent. The Kruskal–Wallis ANOVA would imply that HA_0 should be rejected (p–value < 0.05), while the median test implies that it should not (p–value 0.05). Since the p–value for the median test equals $p = 0.055$, which is very close to the critical value 0.05, we conclude that we should reject the hypothesis HA_0.

Table 6. Homogeneity of variance tests

Transformed Var.	Levene		Brown–Forsythe	
	F statistic	p–value	F statistic	p–value
X_0	3.158	0.003	2.829	0.008
X_1	1.861	0.078	1.592	0.140

Afterwards, we also performed ANOVA, and the results are given in the first row of Table 8. The column labels "SS", "df", "MS", "F", and "p" stand for the sum of squares, degrees of freedom, mean squares ($\frac{SS}{df}$), F statistic, and p–value, respectively. The results show that the equality of means hypothesis should be rejected (p–value < 0.05). Although this conclusion should not be taken seriously, since the assumptions for ANOVA are violated, it provides more evidence in favor of our conclusion that the hypothesis HA_0 should be rejected.

Table 7. Non–parametric tests

Transformed var.	Kruskal–Wallis ANOVA		Median Test	
	H statistic	p–value	χ^2 statistic	p–value
X_0	20.170	0.005	13.803	0.055
X_1	12.786	0.078	10.309	0.172

Consider now the random variable X_1. In that case the assumptions of ANOVA for the eight considered samples are satisfied. Hence, we apply ANOVA, and the results are given in the second row of Table 8. The conclusion is that the equality of means hypothesis HB_0 could not be rejected (p–value > 0.05). Thus, we may assume that the means of the eight samples for the late defect detection density are equal.

Table 8. ANOVA tests

Transformed var.	SS	df	MS	F	p
X_0	25.157	7	3.594	3.069	0.004
X_1	16.337	7	2.334	1.921	0.067

In order to confirm this finding, we additionally performed non–parametric tests. The results are presented in the second row of Table 7. They also show that the hypothesis HB_0 could not be rejected (p–value > 0.05). This confirms our conclusion that the means of X_1 samples are equal.

6 Discussion

The study is performed on samples of software units that are grouped, by organizational responsibility, into nine software design organizations that were responsible for the software unit design. For software units, the effectiveness of defect detection process has been measured in two consecutive phases, early and late defect detection. The early defect detection of a particular software unit is performed by the same organization that performed the software design. The late defect detection is performed by the verification organization for the complete software system, which is composed of all the analyzed software units. The main outcome of the defect detection process is a number of defects that are identified and reported per each software unit. So, the effectiveness of defect detection process is measured per software unit as defect density, that is the number of identified defects per software unit's volume of code.

In the analysis, one of the research goals was exploring the difference in the early defect detection effectiveness among the software design organizations involved into the same global software development project. The results of the analysis indicate that there is a significant difference in the effectiveness of early defect detection among the software design organizations. Note that the same organization was responsible for design and early defect detection on software unit. One may conclude that software design organizations are not equally effective in early defect detection process, but this conclusion might be misleading. The differences in early defect detection effectiveness may be a result of the differences in organizational defect injection process as result of less experience, more complex part of functionality for implementation or other factor identified in [21], although that is also performed by the same organizations.

The other research goal was to understand the influence of the software design organizational distribution on the late defect detection effectiveness and the product quality. The research hypothesis was testing the significance of differences in late defect detection effectiveness for the groups of software units with aforementioned software design responsibilities. Note that late defect detection is performed for all software units by a single verification organization, that is different from the software design organizations. Surprisingly, the result of the analysis was that there is no significant difference in the effectiveness of late defect detection between the software units that were grouped according to the software design responsibility. One may conclude that late defect detection was equally effective for all groups of software units, that were developed and early verified by different software design organizations. In other words, the diversity in early defect detection effectiveness is not statistically significantly reflected in diversity in late defect detection effectiveness.

From these results, we may argue that the defect density is a good effectiveness measure in this context. The finding that late defect detection was equally effective for all software units no matter of effectiveness achieved in early defect detection may be followed by the conclusion that there is no relationship between early and late defect detection effectiveness. Moreover, the diversity of

early defect detection performed within software design organization does not influence the later defect detection effectiveness.

These conclusions should be taken with caution and should be analyzed in the light of the previous work. For example, in a number of studies the early defect detection is identified as a good predictor of late defect detection and remaining defect density in the system. On the other hand, in [9] and [10], it is empirically identified that units with many defects identified in the early defect detection phases have small number of defects in late defect detection phases, and units with majority of defects in late defect detection phases have been the units with small number of defects in early defect detection phases. From these conclusions we may expect that variation in the early defect density is repeated for late defect density that does not happen in our case.

Note that in this study the organizational influence participating in the GSD on the effectiveness of defect detection process and product quality is evaluated. For a proper valuation of the quality produced by development organization the evaluation of outliers is also an interesting indication. Generally, when evaluating organizational influence on the product quality the analysis of the number of faults should not be excluded although one should be aware that the number of faults depends on software size and the relation is not linear.

7 Conclusion and Future Work

The importance of the study presented in this paper lies in its empirically based evaluation of the influence of software development distribution on the global software development. Many challenges have been identified by researchers and practitioners in the global software development, but still a limited empirical evidence is presented. This study is a step in that direction.

In the context of this study, the diversity of early defect detection effectiveness has not influenced the late defect detection effectiveness. On the other hand, the early defect detection was performed by different organizations in the GSD project, while the late defect detection is performed by the same verification organization. This suggests that the defect density is better as the measure of the organizational effectiveness than as the measure of product quality. However, the presented study should be replicated in different contexts so that such conclusion could be generalized.

Another direction for future work is to explore defect detection effectiveness diversity among different GSD projects in the same context of the study. The results of these analysis could bring conclusions on the impact of GSD process evolution on diversity of defect detection effectiveness.

Finally, to get clear picture of the defect detection effectiveness impact on product quality, the defect density over the software unit life–cycle and failure density on the customer site should be also analyzed. Furthermore, the influence of organizational diversity on the severity of defects is also an issue to explore.

References

1. Ajila, S., Dumitrescu, R.: Experimental Use of Code Delta, Code Churn, and Rate of Change to Understand Software Product Line Evolution. J. Syst. Softw. 80(1), 74–91 (2007)
2. Battin, R.D., Crocker, R., Kreidler, J., Subramanian, K.: Leveraging Resources in Global Software Development. IEEE Softw. 18(2), 70–77 (2001)
3. Bird, C., Nagappan, N., Devanbu, P., Gall, H., Murphy, B.: Does Distributed Development Affect Software Quality? An Empirical Case Study of Windows Vista. In: 31st International Conference on Software Engineering ICSE 2009, pp. 518–528. IEEE Computer Society, Washington DC (2009)
4. Carmel, E., Agarwal, R.: Tactical Approaches for Alleviating Distance in Global Software Development. IEEE Softw. 18(2), 22–29 (2001)
5. Cataldo, M., Nambiar, S.: On the Relationship between Process Maturity and Geographic Distribution: an Empirical Analysis of their Impact on Software Quality. In: 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering ESEC/FSE 2009, pp. 101–110. ACM, New York (2009)
6. Chrissis, M., Konrad, M., Shrum, S.: CMMI: Guide for Process Integration and Product Improvement. Addison-Wesley, Boston (2004)
7. Ebenau, R.G., Strauss, S.H.: Software Inspection Process. McGraw Hill, Workingham (1994)
8. Fagan, M.E.: Design and Code Inspections to Reduce Errors in Program Development. IBM Syst. J. 15(3), 575–607 (1976)
9. Fenton, N.E., Ohlsson, N.: Quantitative Analysis of Faults and Failures in a Complex Software System. IEEE Trans. Softw. Eng. 26(8), 797–814 (2000)
10. Fenton, N., Neil, M.: A Critique of Software Defect Prediction Models. IEEE Trans. Softw. Eng. 25(5), 675–689 (1999)
11. Herbsleb, J.D.: Global Software Engineering: the Future of Socio–technical Coordination. In: 2007 Future of Software Engineering FOSE 2007, pp. 188–198. IEEE Computer Society, Washington DC (2007)
12. Institute of Electrical and Electronics Engineers (IEEE): Software Verification and Validation. IEEE Standard 1012–2004, Software Engineering Standards Committee of the IEEE Computer Society (2005)
13. International Organization for Standardization and International Electrotechnical Commission (ISO/IEC): Software Engineering – Product Quality – Part 1: Quality model. ISO/IEC Standard 9126–1, Geneva (1997)
14. Musa, J.D., Iannino, A., Okumoto, K.: Software Reliability Measurement, Prediction, Application. McGraw-Hill, New York (1987)
15. Nagappan, N., Murphy, B., Basili, V.: The Influence of Organizational Structure on Software Quality: an Empirical Case Study. In: 30th International Conference on Software Engineering ICSE 2008, pp. 521–530. ACM, New York (2008)
16. Parnas, D.L.: On the Criteria to be Used in Decomposing Systems into Modules. Commun. ACM 15(12), 1053–1058 (1972)
17. Pfleeger, S.L.: Software Engineering, Theory and Practice. Prentice-Hall, New York (2001)
18. Project Management Institute (PMI): A Guide to the Project Management Body of Knowledge (PMBOK Guide). PMI, Newtown Square (2004)

19. Third Generation Partnership Project (3GPP): Technical Performance Objectives. 3GPP, Technical Specification Group Core Network (2005)
20. Wohlin, C., Höst, M., Henningson, K.: Empirical Research Methods in Software Engineering. In: Conradi, R., Wang, A.I. (eds.) ESERNET 2003. LNCS, vol. 2765, pp. 7–23. Springer, Heidelberg (2003)
21. Jacobs, J., van Moll, J., Kusters, R., Trienekens, J., Brombacher, A.: Identification of factors that influence defect injection and detection in development of software intensive products. Inf. Softw. Technol. 49(7), 774–789 (2007)