# FACTORS OF SOFTWARE QUALITY – ANALYSIS OF EXTENDED ISBSG DATASET

Łukasz RADLIŃSKI*

**Abstract.** In this paper, we analyze the extended ISBSG dataset, which contains data on a wide range of software projects developed in various companies worldwide. The main aim of this paper is to identify important factors that influence software quality and to investigate the nature of these relationships. This analysis involves using various statistical techniques, both analytical and graphical. We provide a rating for each variable to express the strength of its relationship with software quality. Unlike earlier analyses, we focus on the business perspective and its relationships on software quality. Obtained results may be used do support decision making in software projects, specifically by demonstrating the impact of selected software development practices.

**Keywords:** software quality, identification, process factors, project factors, empirical data, ISBSG, analysis, statistical techniques, rule induction

## 1. Introduction

Software quality is an important aspect of software projects. It may be expressed quantitatively by various measures that reflect software defectiveness. The most common of these measures are number of defects, defects rate, fault proneness, time between failures or time to the next failure.

Developing software of high quality requires effective processes of quality management and software engineering. Various authors proposed several approaches focused on achieving high quality of software [5], [10], [12], [20]. In managing software quality, it is essential to identify and analyze the factors that influence software quality. While experienced practitioners are typically able to intuitively determine such factors, for more general use it is necessary to provide an empirical confirmation of investigated relationships.

For many years Jones [9] has been analyzing software quality data from various companies worldwide. His work has been widely recognized and is often treated as a

* Department of Information Systems Engineering, University of Szczecin, ul. Mickiewicza 64, 71-101 Szczecin, Poland, lukrad@uoo.univ.szczecin.pl

baseline by other authors, e.g. [3], [6], [15], [19]. Although Jones analyzed a variety of factors affecting software quality, there are still other factors not investigated in his work. This paper focuses on these additional factors but, to provide complete results, considers also some of those mentioned by Jones.

In this paper, we performed our own analysis of empirical data for diverse software projects. The main goal of this paper was to identify the factors affecting software quality and to investigate the nature of these relationships. We analyzed three dependent variables describing software quality: *number of defects*, *defect rate* (i.e. number of defects in a software unit) and *zero defects* (i.e. an indication if the software has zero or more defects).

We achieved the goal by answering the following three research questions:

- **RQ1**: How are various project factors related with the number of defects?
- **RQ2**: How are various project factors related with the defect rate?
- **RQ3**: What are the features of projects delivered with zero defects?

To answer these questions we performed an analysis of ISBSG dataset [7] – a large set of software project data. We used previous editions of this dataset in earlier analyses [17], [18]. Other researchers also used it, mainly in the analyses of development effort [2], [11], [14]. We selected this dataset because it covers a variety of software projects. Furthermore, in contrast with previous studies, we analyzed an extended version of this dataset that contains more variables. It is important from the business perspective because these additional variables provide more details on various management initiatives and software development process. According to our knowledge, no results covering this extended dataset have already been published.

In this analysis, we used various non-parametric statistical measures and several types of graphs. Additionally, in the analysis focused on zero defects (RQ3), we analyzed the rules generated by the CN2 algorithm [4].

We believe that the results of this analysis might be a useful contribution for the software engineering research community. More importantly, managers of software projects may also use these results as guidelines in leading future software projects.

This paper is organized as follows: Section 2 describes the ISBSG dataset used in this study and explains the research procedure that we followed. Section 3 provides the details on obtained results. Section 4 summarizes these results and compares them with previous analysis. Section 5 considers the limitations and threats to validity of the obtained results. Section 6 draws the conclusions and discusses the future work.

## 2. Research approach

### 2.1. Dataset Used

In this study, we used the ISBSG dataset [7]. It contains data on 5204 software projects developed worldwide. The scope, quality, purpose, features, release date and development process of these projects are very diverse. The standard version of this dataset contains about 100 variables describing the projects. However, in this study we used an extended version that contains 205 variables.

Table 1 provides the details for the variables used in this analysis. To keep it concise we list only those variables that were kept in the analysis after stage 1 (see Section 2.2 for details).

**Table 1. Variables used in analysis**

| Variable | Number of cases* | | T# | Variable | Number of cases* | | T# |
|---|---|---|---|---|---|---|---|
| | TD | DR | | | TD | DR | |
| Total defects delivered | 935 | 686 | I | CASE tool used | 403 | 355 | B |
| Defect rate | 686 | 686 | D | Upper CASE used | 358 | 318 | B |
| Zero defects | 935 | 686 | B | Lower CASE (with code gen) used | 229 | 203 | B |
| Data quality rating | 935 | 686 | N | Lower CASE (no code gen) used | 283 | 258 | B |
| Year of Project | 935 | 686 | I | Integrated CASE used | 230 | 205 | B |
| Functional size | 686 | 686 | I | Project user involvement | 124 | 94 | B |
| Summary work effort | 935 | 686 | D | Project manager experience | 222 | 208 | I |
| Activity planning | 641 | 499 | B | Project manager changes | 242 | 228 | I |
| Activity specification | 630 | 500 | B | Personnel changes | 253 | 237 | I |
| Activity design | 644 | 503 | B | Metrics program | 362 | 355 | B |
| Activity build | 650 | 504 | B | User satisfaction survey | 196 | 104 | B |
| Activity test | 646 | 503 | B | Meet stated objectives | 95 | 49 | R |
| Activity implement | 639 | 501 | B | Meet business requirements | 99 | 52 | R |
| Development type | 934 | 686 | N | Quality of functionality | 93 | 47 | R |
| Application type | 822 | 581 | N | Quality of documentation | 100 | 55 | R |
| Package customization | 283 | 246 | B | Ease of use | 93 | 48 | R |
| Architecture | 609 | 520 | N | Training given | 93 | 46 | R |
| Client-server | 595 | 505 | B | Speed of defining solution | 105 | 60 | R |
| Development platform | 857 | 618 | N | Speed of providing solution | 105 | 60 | R |
| Language type | 881 | 638 | N | Process improvement program | 205 | 192 | R |
| Portability requirements | 306 | 267 | N | Project objective A: all functionality | 197 | 184 | R |
| Used methodology | 638 | 545 | B | Project objective B: minimum defects | 204 | 191 | R |
| Project management tools | 295 | 257 | B | Project objective C: minimum cost | 194 | 181 | R |
| Debugging tools | 345 | 306 | B | Project objective D: shortest time | 199 | 186 | R |
| Testing tools | 325 | 287 | B | Build products | 231 | 217 | N |
| Performance monitoring tools | 279 | 243 | B | | | | |

* Number of cases with non-missing values: TD – for analysis of *total defects delivered* and *zero defects*, DR – for analysis of *defect rate*

# Type: B – Boolean, D – Double, I – Integer, N – Nominal, R - Ranked

There are three dependent variables in this study, one for each research question stated in Section 1:

- *Total defects delivered* – the total number of defects reported in the first month of use of the software;
- *Defect rate* – the number of defects per function point, defined as a ratio: *total defects delivered* / *functional size*;
- *Zero defects* – Boolean variable indicating whether or not the *total defects delivered* for a project was equal to '0'.

The ISBSG dataset also contains other variables describing software quality – number of defects per project categorized by their severity, i.e. extreme, major and minor defects. However, they contain significantly more missing values than dependent variables selected for our experiment. Besides, we have already investigated factors affecting the number of categorized defects in [18].

Because the dataset provides data aggregated at the project level, we could not investigate the factors of fault proneness of detailed software parts (sub-systems, components, etc.). Further, the dataset does not contain any data on failures caused by software defects.

## 2.2. Research procedure

The research procedure for this analysis involved the following four stages:

**1. Data preparation**

The aim of this stage was to prepare the dataset for the analysis. In the first step we reviewed available data. We inspected the dataset and corrected obvious mistakes in the dataset, for example typos. We also removed all values 'don't know' because such values do not provide more useful information than if the the values were missing.

Based on the previous analyses, mainly involving the same dataset, we have pre-identified the variables that might be useful for this analysis. We removed only those variables from further analysis, which were clearly out of scope or for which no detailed and appropriate explanation was available. Specifically, we removed nominal variables with unclear definition of their states.

Several nominal variables have multiple responses. For example, respondents often provided several *application types* for a single project. In earlier study [17], we had grouped such projects by assigning a single value for such variables. However, that caused the loss of information because projects with several *application types* originally provided, had been assigned to a single aggregated *application type*. To avoid it in this analysis we created dummy Boolean variables for most common states of particular multiple-response variable, for example 19 dummy variables for *application type*.

As suggested in [8], in further analysis we used only projects with data quality assessed as 'A' or 'B' (meaning 'very high' or 'high' data quality). Thus, we had 935 cases with values provided for *total defects delivered* and *zero defects*, and 686 cases with values provided for *defect rate*. These numbers of cases are not equal because for some projects no data for *functional size* was provided, what made impossible to calculate the *defect rate*.

In the next step, we removed variables with very high number of missing values, i.e. where the number of non-missing values was lower than about few dozens.

Several statistical tests that we planned to use require the numeric variables to be normally distributed. Based on previous analysis [17], we suspected that the key numeric variables were not normally distributed. Thus, we transformed them using ln() function (natural logarithm) as we did in previous analysis and as suggested in [13]. During these transformations, we made adjustments to some variables. The natural logarithm cannot be calculated for value '0', however, for several projects the *total defects delivered*, and thus the *defect rate*, was equal to '0'. To avoid losing these cases, we transformed such values to a small positive value, as suggested in [7], for which the natural logarithm can be computed. We set these values to '0.1' and called the variables involved with this transformation with the suffix 'adjusted' (as can be seen on some further figures).

**2. Basic data analysis**

The aim of the basic analysis was to determine the most appropriate techniques, such as statistical measures, for detailed analysis. For this purpose, we mainly analysed the histograms but also basic statistics and frequency tables. Based on obtained results, we decided to use non-parametric tests – see Section 3.1 for detailed justification.

**3. Detailed analysis of correlations and associations**

The aim of this stage was to analyse the relationships that would help answering the three research questions stated in Section 1. In the analysis of relationships with *total defects delivered* and *defect rate*, we used the following techniques:

- non-parametric measures – Spearman's rank correlation coefficient (ρ), Mann-Whitney U, Kruskal-Wallis H,
- visual techniques – scatterplots, box-plots and categorized histograms.

For ranked variables with more than two states, we used both ρ and H. For some states of nominal variables, there were very few observations of *total defects delivered* or *defect rate*. To avoid bias of such states on H, we used only those states, for which there were at least five observations for these dependent variables.

In the analysis of relationships with *zero defects*, we used the following techniques:

- measures of association based on cross-tabulations such as phi coefficient (ϕ), contingency coefficient (C), Cramer's V, uncertainty coefficient (UC),
- visual techniques – frequency plots and categorized histograms,
- rules generated by CN2 rule induction algorithm [4].

We performed all analyses using Statistica package [21], except for generating CN2 rules, where we used an Orange tool [16].

**4. Interpretation of results**

The aim of this stage was to interpret the results from the previous stage. This involved the analysis of identified relationships and the relationships that were surprisingly not confirmed in the analysis. Based on obtained results we discussed the efficiency of some some software development initiatives. We also set the ratings to each independent variable. These ratings reflect the aggregated measures of the strength of relationships with particular dependent variables. Managers may use these results to make informed decisions in future software projects.

# 3. Results

## 3.1. Basic data analysis

In this basic analysis, we investigated the nature of key numeric variables using frequency tables, basic statistics and histograms. Figure 1 illustrates the histograms for these variables; the solid line indicates the Normal distribution fitted to the histogram. The distributions of original data, i.e. without any transformation, clearly do not follow Normal distribution. After transformation with ln() function, the distributions are closer to Normal, especially for *functional size* and *summary work effort*.

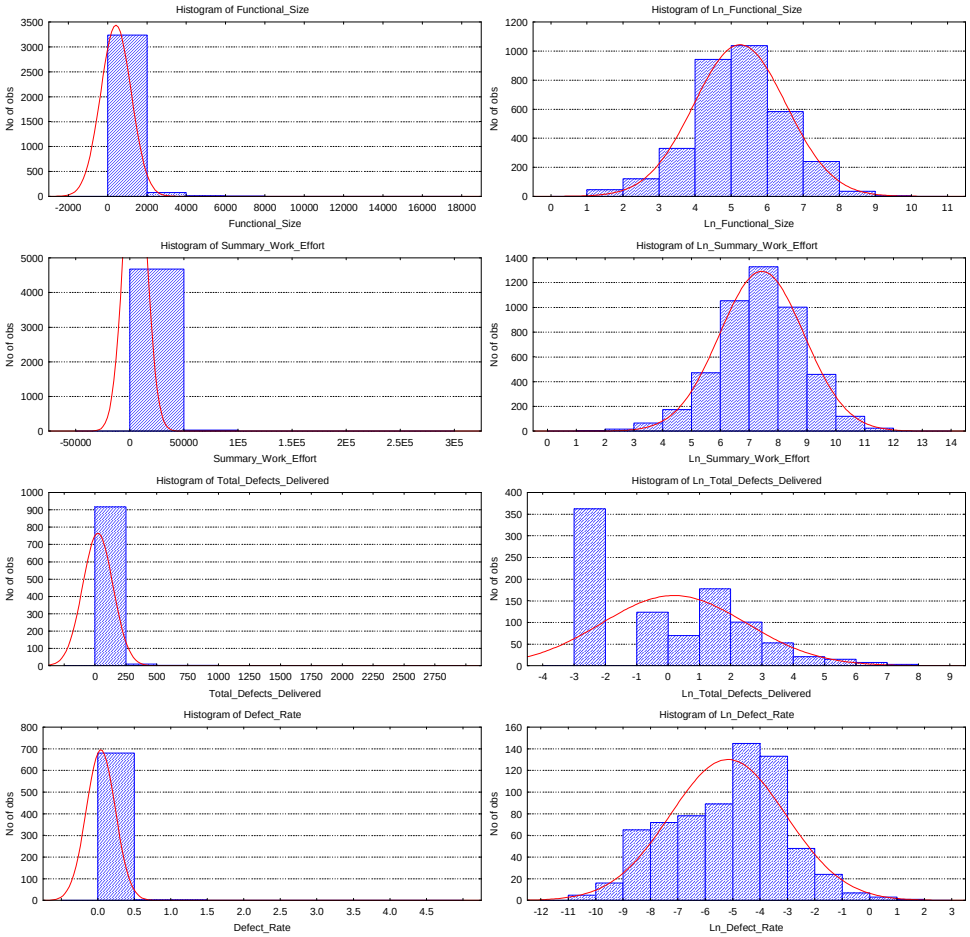


**Figure 1. Histograms for key numeric variables without data transformation (left) and after transformation with natural logarithm (right)**

Although distributions for *total defects delivered* and *defect rate* are much closer to Normal after log-transformation than without it, they are still not matching it. The first interval for *total defects delivered* (-3, -2) reflects projects with no defects. The high number of such projects causes that the distribution is not symmetrical, even after log-transformation. When not considering this interval, other intervals still do not follow the Normal distribution. The distribution for *defect rate* after log-transformation is closer to Normal than for *total defects delivered*. However, there are still some intervals, especially (-9, -8), (-8, -7), (-5, -4) and (-4, -3), which cause that the whole distribution is not Normal.

This lack of normality *for total defects delivered* and *defect rate* is even more clearly visible in distributions categorized by the states of nominal and Boolean variables. Available data do not meet the assumptions of using statistical parametric measures. Thus, we decided to use non-parametric measures that do not have such strong assumptions.

## 3.2. Relationships with numeric and ranked variables

In the first step of detailed analysis, we analyzed correlations between two dependent variables: *total defects delivered* and *defect rate* with various numeric and ranked variables. Table 2 lists the values of Spearman's rank correlation coefficient ρ (values of $p<0.05$ highlighted in bold).

For *total defects delivered* there are six variables with ρ statistically significant at $p<0.05$. The correlations with *functional size* and *meet stated objectives* are moderate, while the correlations with *project manager experience*, *quality of functionality*, *summary work effort*, and *year of project* are weak.

For *defect rate*, there are only two variables with ρ statistically significant at $p<0.05$: *meet stated objectives* and *project manager experience*.

None of four variables reflecting *project objective* seem to be correlated with *total defects delivered* and *defect rate*. Still, the correlation between *project objective B: minimum defects* and *total defects delivered* is weak and the significance level only slightly exceeds assumed value of 0.05. For *ease of use* the significance level also only slightly exceeds assumed value.

We included the *year of project* in this analysis to investigate the level of software defectiveness as the time progresses and new technologies and development processes are applied. Obtained results show that *year of project* is only weakly negatively correlated with *total defects delivered*, indicating fewer defects in newer projects. We found no correlation of *year of project* with *defect rate*.

Using scatterplots we further investigated the relationships between selected numeric variables. Figure 2 illustrates some of them. They confirm the relationship between *functional size* and *total defects delivered* but it is not straightforward because other factors also influence *total number of defects*. The relationship between *summary work effort* and *total defects delivered* can be explained by the fact that *summary work effort* is clearly correlated with *functional size*, which in turn is correlated with *total defects delivered*. These graphs also show the lack of clear relationships with *defect rate*.

**Table 2. Values of Spearman's rank correlation coefficient**

| Variable | Total defects delivered | | Defect rate | |
|---|---|---|---|---|
| | ρ | p | ρ | p |
| Ease of use | -0.19 | 0.074 | -0.27 | 0.063 |
| Functional size | **0.33** | **0.000** | -0.02 | 0.649 |
| Meet business requirements | -0.14 | 0.158 | -0.09 | 0.540 |
| Meet stated objectives | **-0.44** | **0.000** | **-0.35** | **0.013** |
| Personnel changes | 0.02 | 0.722 | -0.10 | 0.144 |
| Project manager changes | 0.06 | 0.373 | 0.01 | 0.913 |
| Project manager experience | **-0.21** | **0.002** | **-0.18** | **0.010** |
| Project objective A: all functionality | 0.05 | 0.465 | 0.10 | 0.188 |
| Project objective B: minimum defects | 0.14 | 0.053 | 0.01 | 0.915 |
| Project objective C: minimum cost | 0.11 | 0.134 | 0.09 | 0.212 |
| Project objective D: shortest time | 0.03 | 0.669 | 0.08 | 0.307 |
| Quality of documentation | 0.08 | 0.423 | -0.09 | 0.491 |
| Quality of functionality | **-0.27** | **0.010** | -0.21 | 0.156 |
| Speed of defining solution | 0.07 | 0.469 | -0.03 | 0.839 |
| Speed of providing solution | 0.08 | 0.416 | -0.05 | 0.707 |
| Summary work effort | **0.24** | **0.000** | 0.03 | 0.400 |
| Training given | -0.12 | 0.258 | -0.26 | 0.087 |
| Year of project | **-0.13** | **0.000** | -0.02 | 0.640 |

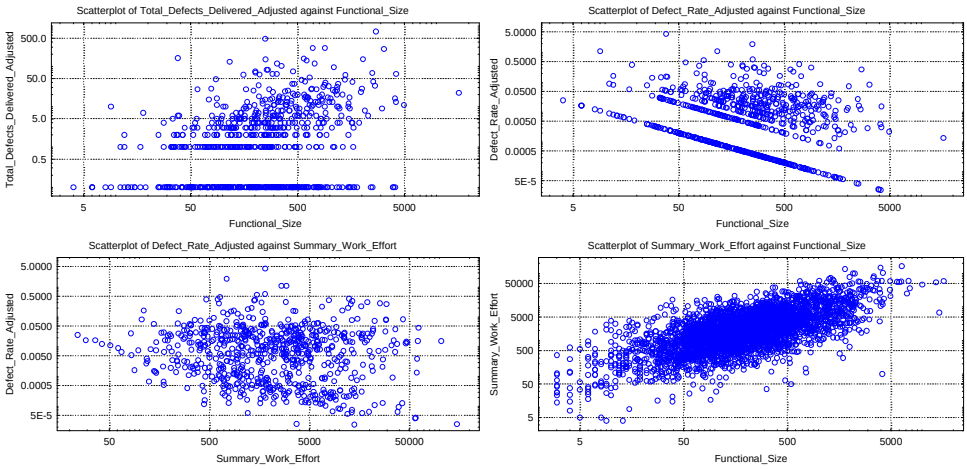


**Figure 2. Relationships between key numeric variables**

### 3.3. Relationships with nominal variables

We analyzed relationships between nominal independent variables and dependent variables:

- for Boolean independent variables – using Mann-Whitney U,
- for nominal independent variables with more than two states – using Kruskal-Wallis H.

Table 3 lists the values of Mann-Whitney U. The majority of preselected variables seem to be in relationship with *total defects delivered*. However, only 16 variables seem to be in relationship with *defect rate*.

**Table 3. Values of Mann-Whitney U**

| Variable | Total defects delivered | | Defect rate | |
|---|---|---|---|---|
| | U | p | U | P |
| Activity Build | 1036 | 0.050 | - | - |
| Activity Design | 44176 | **0.002** | 29798 | 0.684 |
| Activity Implement | 33584 | **0.001** | 22819 | 0.282 |
| Activity Planning | 35548 | **0.018** | 21451 | **0.007** |
| Activity Specification | 24992 | **0.016** | 15453 | 0.072 |
| Activity Test | 8353 | **0.000** | 4015 | 0.564 |
| Application type: Billing | 3610 | **0.001** | 3234 | 0.052 |
| Application type: Catalogue register | 8521 | 0.125 | 6407 | 0.334 |
| Application type: Document Management | 10608 | **0.006** | 7900 | 0.050 |
| Application type: DSS | 2318 | 0.823 | 1363 | 0.377 |
| Application type: EDI | 12369 | **0.045** | 8608 | 0.218 |
| Application type: EIS | 2738 | **0.009** | 1106 | 0.371 |
| Application type: Embedded | 1650 | **0.000** | - | - |
| Application type: Financial accounting | 64019 | 0.504 | 21870 | **0.002** |
| Application type: GIS | 2747 | 0.198 | 1219 | 0.554 |
| Application type: Logistic supply | 2957 | **0.000** | 2187 | **0.004** |
| Application type: MIS | 17551 | **0.000** | 13675 | 0.786 |
| Application type: Multimedia | 8901 | 0.556 | 5799 | 0.915 |
| Application type: Office automation information | 8679 | **0.006** | 6381 | 0.964 |
| Application type: Onlie analysis reporting | 9943 | **0.003** | 6968 | **0.027** |
| Application type: Performance reporting | 11084 | 0.364 | 6797 | 0.852 |
| Application type: Project management | 4120 | **0.013** | 1886 | **0.024** |
| Application type: Relationship management | 3610 | **0.001** | 3234 | 0.052 |
| Application type: Transaction-production | 51705 | 0.927 | 31000 | 0.367 |

| Variable | Total defects delivered | | Defect rate | |
|---|---|---|---|---|
| | U | p | U | P |
| Application type: Workflow | 10852 | 0.105 | 7546 | 0.422 |
| Build products: Content | 2977 | **0.000** | 3050 | **0.001** |
| Build products: Database objects | 4540 | **0.000** | 4331 | **0.002** |
| Build products: Graphics | 4776 | **0.002** | 5049 | 0.255 |
| Build products: Inhouse | 1327 | **0.001** | 1363 | **0.005** |
| Build products: Integration scripts-checklist | 2403 | **0.044** | 2288 | 0.249 |
| Build products: Source code | 1390 | **0.001** | 1463 | **0.007** |
| Build products: Unit test plans/designs | 5645 | 0.554 | 5101 | 0.786 |
| CASE tool used | 16672 | **0.002** | 13034 | **0.005** |
| Client-server | 38134 | **0.004** | 28364 | 0.051 |
| Debugging tools | 12929 | 0.111 | 9783 | **0.033** |
| Integrated CASE used | 583 | 0.256 | 406 | 0.063 |
| Lower CASE (no code gen) used | 5484 | **0.001** | 5807 | 0.179 |
| Lower CASE (with code gen) used | 1326 | 0.738 | 1029 | 0.555 |
| Metrics program | 2898 | **0.019** | 2750 | **0.025** |
| Package customisation | 3288 | **0.020** | 2234 | 0.471 |
| Performance monitoring tools | 8229 | 0.165 | 6535 | 0.841 |
| Portability requirements | 3392 | **0.000** | 2435 | 0.108 |
| Process improvement program | 4644 | 0.315 | 4286 | 0.879 |
| Project management tools | 4149 | **0.000** | 3145 | **0.000** |
| Project user involvement | 715 | **0.000** | 333 | 0.148 |
| Testing tools | 10401 | **0.006** | 8707 | 0.058 |
| Upper CASE Used | 9660 | **0.006** | 7350 | **0.004** |
| Used methodology | 11032 | **0.000** | 8557 | **0.041** |
| User satisfaction survey | 715 | **0.000** | 463 | **0.000** |

The dataset contains several variables reflecting tools, processes, initiatives etc, for software quality management. However, we found no statistically significant values of U for *performance monitoring tools*, *process improvement program* in analysis of *total defects delivered* and *defect rate*. Additionally, for *defect rate* the list of such variables includes also: *process improvement program*, *project user involvement*, and *testing tools* (although for the latter the significance level exceeds only slightly the assumed threshold).

In the analysis of *defect rate,* there were not enough observations to calculate U for *activity build* and *application type: embedded*.

Table 4 lists the values of Kruskal-Wallis H for both dependent variables and all nominal independent variables. In the analysis of relationships with *total defects delivered*,

seven variables have the value of H statistically significant at $p<0.05$: *architecture*, *development platform*, *development type*, *ease of use*, *language type*, *meet stated objectives* and *quality of functionality*. Five variables appear to be correlated with *defect rate* – *architecture*, *development platform*, *development type*, *meet stated objectives* and *training given*.

Note, that here we also investigated selected ranked independent variables, previously analyzed using ρ coefficient. For these variables existence of confirmed relationships match the results in Table 2, except the level of user satisfaction with the *training given* – relationship with *defect rate* found using H but not confirmed using ρ. However, the level of statistical significance only slightly exceeds assumed threshold for *project objective B: minimum defects* in analysis of *total defects delivered* and *project objective C: minimum cost* for both *total defects delivered* and *defect rate*.

**Table 4. Values of Kruskal-Wallis H**

| Variable | Total defects delivered | | Defect rate | |
|---|---|---|---|---|
| | H | p | H | p |
| Architecture | 19.94 | **0.000** | 15.02 | **0.002** |
| Development platform | 19.69 | **0.000** | 24.71 | **0.000** |
| Development type | 30.79 | **0.000** | 9.94 | **0.007** |
| Ease of use | 4.87 | **0.027** | 2.10 | 0.147 |
| Language type | 17.13 | **0.000** | 1.60 | 0.449 |
| Meet business requirements | 2.49 | 0.288 | 2.60 | 0.272 |
| Meet stated objectives | 17.21 | **0.000** | 5.94 | **0.015** |
| Project objective A: all functionality | 0.85 | 0.838 | 3.46 | 0.327 |
| Project objective B: minimum defects | 6.97 | 0.073 | 0.36 | 0.948 |
| Project objective C: minimum cost | 7.69 | 0.053 | 7.61 | 0.055 |
| Project objective D: shortest time | 1.72 | 0.633 | 1.00 | 0.802 |
| Quality of documentation | 0.78 | 0.855 | 2.09 | 0.352 |
| Quality of functionality | 13.63 | **0.000** | 1.89 | 0.169 |
| Speed of defining solution | 1.46 | 0.690 | 0.00 | 0.970 |
| Speed of providing solution | 0.96 | 0.812 | 0.06 | 0.803 |
| Training Niven | 2.88 | 0.236 | 5.64 | **0.018** |

As a further step of this analysis, we used box-plots that indicate the ranges of particular dependent variables for selected states of nominal or Boolean variables. Figure 3 illustrates sample box-plots. When analyzing *development type*, we can see that projects that are of type ‘re-development’ are usually more defective than projects for new software or enhancements of existing software. We found no statistically significant relationship between new projects and enhancements in the analysis of *defect rate*.

We further investigated why earlier quantitative measures did not confirm existence of relationship between *project objective B: minimum defects* and dependent variables. With

*project objective B: minimum defects* = 1, i.e. that this objective was the priority for the project, the median *total defects delivered* is actually the lowest compared to projects for which *project objective B: minimum defects* had lower priority. However, for projects with *project objective B: minimum defects* = 4, i.e. that this objective was only the fourth, the median of *total defects delivered* was lower than for projects with *project objective B: minimum defects* = 3. Additionally, the ranges of possible values of *total number of defects* for different states of *project objective B: minimum defects* were very high, what confirmed the lack of clear relationship between these variables. The median values of *defect rate*, as well as the ranges and percentiles, for different states of *project objective B: minimum defects* were very similar.

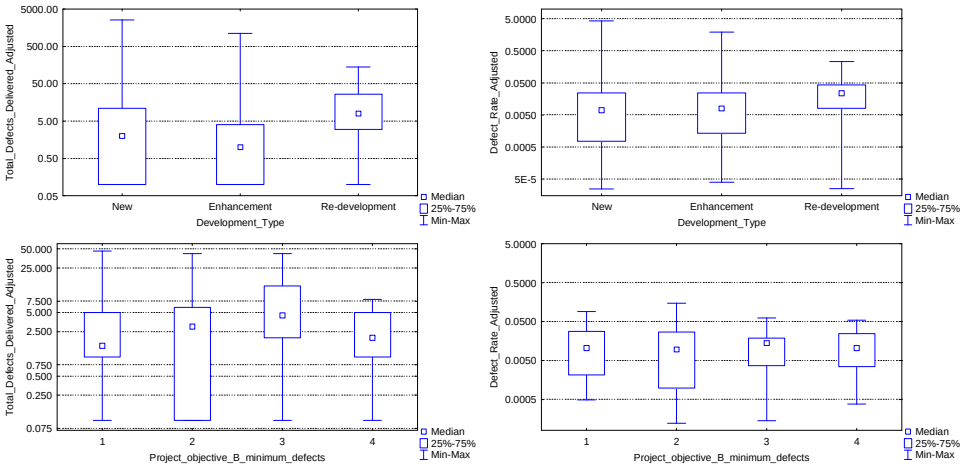


**Figure 3. Box-plots for number of defects (left) and defect rate (right) grouped by development type (top) and project objective B: minimum defects (bottom)**

## 3.4. Associations with Zero defects

We used various measures to analyze the associations between *zero defects* and other variables (nominal, Boolean and ranked). These measures include Phi coefficient ($\phi$), contingency coefficient (C), Cramer's V and uncertainly coefficient (UC). Table 5 lists the results (only those variables with $p<0.05$ are shown). We calculated C and UC for all variables, $\phi$ for Boolean variables (as it can be computed only for 2x2 tables) and V for non-Boolean variables (for 2x2 tables V is equal to $\phi$).

The interpretation of values for these measures is more difficult than for measures of correlation. However, the two general rules are the following:

- the values cannot be lower than '0' or higher than '1',
- for a pair of variables, the higher value indicates the stronger relationship.

Based on obtained results we consider the following variables as associated with *zero defects*: *user satisfaction survey*, *project user involvement*, *ease of use*, *meet stated*

*objectives, build products: database objects, training given, meet business requirements, quality of functionality, upper case tool used* and *project objective C: minimum cost.*

**Table 5. Values of measures of association for zero defects**

| Variable | ϕ | C | V | UC |
|---|---|---|---|---|
| Activity Build | 0.08 | 0.08 | | 0.01 |
| Activity Design | 0.09 | 0.09 | | 0.01 |
| Activity Implement | 0.09 | 0.09 | | 0.01 |
| Activity Test | 0.12 | 0.12 | | 0.02 |
| Application type: Billing | 0.10 | 0.10 | | 0.02 |
| Application type: Catalogue register | 0.07 | 0.07 | | 0.01 |
| Application type: Document management | 0.07 | 0.07 | | 0.01 |
| Application type: EDI | 0.09 | 0.09 | | 0.01 |
| Application type: Embedded | 0.15 | 0.15 | | 0.04 |
| Application type: Financial/accounting | 0.10 | 0.10 | | 0.01 |
| Application type: Logistics/supply | 0.11 | 0.11 | | 0.02 |
| Application type: MIS | 0.07 | 0.07 | | 0.01 |
| Application type: Onlie analysis reporting | 0.11 | 0.11 | | 0.02 |
| Application type: Project management | 0.09 | 0.09 | | 0.01 |
| Application type: Relationship management | 0.10 | 0.10 | | 0.02 |
| Application type: Workflow | 0.07 | 0.07 | | 0.01 |
| Architecture | | 0.17 | 0.17 | 0.02 |
| Build Products: Content | 0.23 | 0.22 | | 0.05 |
| Build Products: Database objects | 0.27 | 0.26 | | 0.05 |
| Build Products: Graphics | 0.18 | 0.18 | | 0.03 |
| Build Products: Inhouse | 0.22 | 0.21 | | 0.05 |
| Build Products: Source code | 0.20 | 0.20 | | 0.04 |
| CASE tool used | 0.15 | 0.15 | | 0.02 |
| Client-server | 0.14 | 0.14 | | 0.02 |
| Development platform | | 0.18 | 0.18 | 0.02 |
| Development type | | 0.12 | 0.12 | 0.01 |
| Ease of use | | 0.31 | 0.32 | 0.10 |
| Language type | | 0.15 | 0.16 | 0.02 |
| Meet business requirements | | 0.26 | 0.27 | 0.08 |
| Meet stated objectives | | 0.29 | 0.31 | 0.09 |
| Metrics program | 0.13 | 0.13 | | 0.02 |
| Portability requirements | 0.16 | 0.15 | | 0.03 |

| Variable | ϕ | C | V | UC |
|---|---|---|---|---|
| Project management tools | 0.16 | 0.16 | | 0.02 |
| Project objective C: minimum cost | | 0.24 | 0.25 | 0.03 |
| Project user involvement | 0.30 | 0.29 | | 0.13 |
| Testing tools | 0.13 | 0.13 | | 0.01 |
| Training given | | 0.28 | 0.29 | 0.08 |
| Upper CASE used | 0.18 | 0.32 | | 0.03 |
| Used methodology | 0.14 | 0.13 | | 0.02 |
| User satisfaction survey | 0.56 | 0.49 | | 0.30 |

To further investigate the associations between *zero defects* and two we analyzed the frequencies from cross-tabulations. Figure 4 illustrates them for two variables with the highest values for measures of associations. When there was no *user satisfaction survey* performed, only two projects were delivered with zero defects (3% of all projects for which no such survey was performed). When there was user satisfaction survey performed, 84 projects were delivered with zero defects (63% of all projects for which such survey was performed).

When users were not involved in a project (32 cases) none project was delivered with zero defects. However, when users were involved in the project (92 cases), 26 projects (28%) were delivered with no defects. Still, 66 projects (72%) were delivered with at least one defect.

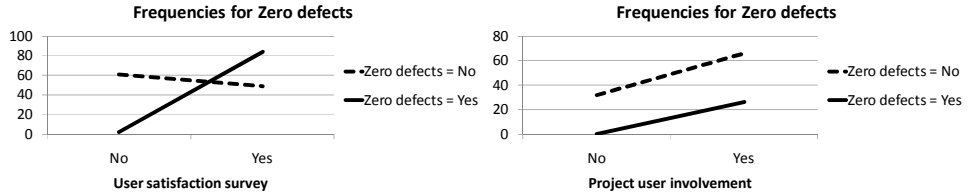


**Figure 4. Frequencies for zero defects**

Apart from the above statistical analysis, we used the CN2 algorithm to generate rules that provide more explanation for *zero defects*. These rules have a form of if-then statements: *IF <set of conditions> THEN <the outcome>*. In the case of this analysis, the outcome is a predicted state for *zero defects* and may take one of two forms: *zero defects* = 'Yes' or *zero defects* = 'No'. Table 6 lists the most relevant rules with *rule quality* > 0.9 and *rule coverage* > 10. In the analysis of associations, we used only nominal, Boolean and ranked independent variables (no variables of type Integer or Double). However, for CN2 rules we also used numeric variables to get a clearer picture of possible influences for *zero defects*.

Among 15 most important rules, only four of them provide the outcome of '*zero defects* = 'Yes', while eleven for *zero defects* = 'No'. Although the quality of these rules is high, the coverage is rather low. It means that the rules are quite accurate but they were built on limited number of projects.

**Table 6. Rules induced for zero defects**

| Quality | Coverage | Conditions | Predicted class |
|---|---|---|---|
| 0.982 | 54 | User satisfaction survey = No<br>Summary work effort > 652 | No |
| 0.968 | 29 | Build products: Content = Yes<br>Testing tools = Yes<br>Functional size > 140 | No |
| 0.967 | 28 | Application type: Financial/accounting = Yes<br>Development type = New<br>Development platform = Multi | No |
| 0.955 | 20 | Project manager experience ≤ 4<br>Summary work effort ≤ 7722<br>Summary work effort > 1011 | No |
| 0.955 | 20 | Upper CASE used = No<br>Functional size > 795<br>Summary work effort > 1210 | No |
| 0.955 | 20 | Summary work effort < 51 | Yes |
| 0.944 | 16 | Application type: Logistics/supply = Yes | No |
| 0.944 | 16 | Upper CASE Tool = No<br>Activity Implement = No<br>Activity Specification = Yes<br>Functional size ≤ 346 | No |
| 0.941 | 15 | Development platform = PC<br>Summary work effort > 4266<br>Summary work effort ≤ 13980<br>Functional size ≤ 638 | Yes |
| 0.938 | 14 | Development type: Re<br>Activity specification = Yes | No |
| 0.929 | 12 | Application type: project management = Yes<br>Summary work effort < 7830 | No |
| 0.929 | 12 | Metrics program = No<br>Used methodology = Yes | No |
| 0.923 | 11 | Project user involvement = No | No |
| 0.923 | 11 | Application type: embedded = Yes | Yes |
| 0.923 | 11 | Development platform: PC = Yes<br>Upper CASE used = Yes<br>Package customization = No<br>Summary work effort > 1200 | Yes |

Most of generated rules were built on values of *functional size*, *summary work effort* and typically values of 1-3 other variables, such as *user satisfaction survey*, *build products: content*, *testing tools*, *application type*, *development type*, *development platform*, *upper CASE tool used*, variables describing development activities involved, and other variables. Some of these variables were identified in analysis of associations with *zero defects*, but for

other variables no strong statistically significant level of association was found and they appeared only in the generated rules.

Only four rules have simple conditions, i.e. with a single variable in the condition statement. These rules use the following four variables: *summary work effort*, *application type: logistics/supply*, *project user involvement* and *application type: embedded*. Two of such rules explain existence of projects with no defects and the other two rules existence of projects with defects.

To investigate the relationships between *functional size*, *summary work effort* and *zero defects* we plotted the categorized scatterplot shown in Figure 5. Projects with size up to 50 function points and involving effort up to 500 person-hours were more often delivered with zero defects. With the increase of *functional size* and *summary work effort*, the proportion of projects with no defects decreases. However, there were some large projects with high amount of effort spent, which were still delivered with zero defects.

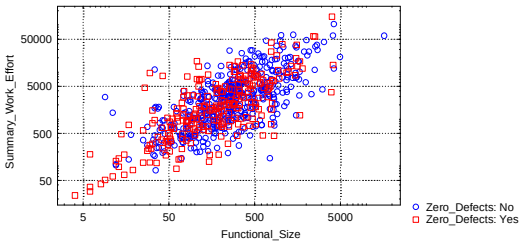


**Figure 5. Frequencies for zero defects**

## 4. Lessons Learned and Discussion

To answer the research questions stated in Section 1 in synthetic way, we assigned ratings to each variable. These ratings quantify the strength of relationship with each dependent variable. These ratings are integer values with the following interpretation:

- '0' – no or negligible relationship,
- '1' – weak relationship,
- '2' – moderate relationship,
- '3' – strong relationship.

Assigning the rating of '0' does not mean that particular variable is not in relationship with dependent variable. It rather means that we did not find such relationship in this dataset using discussed techniques.

We set the values for ratings in a heuristic process based on several indicators obtained in analysis discussed in Section 3. The first source of information for these ratings was the set of values of statistical measures (including *p*). The second source were the visualizations of relationships, including scatterplots, box-plots, and histograms. For space constraints, in Section 3 we showed and discussed in details only examples of them. We investigated graphs for all pairs of variables, for which we calculated quantitative measures. The third source of information for these ratings was the author's expert belief based on the strength of support provided by the data and personal experience. We explain this on the following

example: let us assume that for two variables, A and B, the calculated measures were similar, and the graphs showed similar power of relationships with dependent variable. However, the relationships for the variable A were based on fewer cases than for variable B. In such case, the aggregated rating for variable A is lower than for the variable B. The fact of using these three sources of information made it very hard, if not impossible, for the rating to be set with some standardized and objective procedure or equation. Table 7 lists analyzed variables and these ratings.

**Table 7. Variables and their relationship ratings**

| Variable | Dependent variable* | | | Variable | Dependent variable* | | |
|---|---|---|---|---|---|---|---|
| | TD | DR | 0D | | TD | DR | 0D |
| Year of Project | 1 | 0 | 0 | Lower CASE (with code gen) used | 1 | 0 | 0 |
| Functional size | 3 | 0 | 2 | Lower CASE (no code gen) used | 0 | 0 | 0 |
| Summary work effort | 2 | 0 | 2 | Integrated CASE used | 0 | 0 | 0 |
| Activity planning | 1 | 1 | 0 | Project user involvement | 3 | 0 | 2 |
| Activity specification | 1 | 0 | 0 | Project manager experience | 2 | 1 | 0 |
| Activity design | 1 | 0 | 0 | Project manager changes | 0 | 0 | 0 |
| Activity build | 1 | - | 0 | Personnel changes | 0 | 0 | 0 |
| Activity test | 2 | 0 | 1 | Metrics program | 1 | 1 | 1 |
| Activity implement | 2 | 0 | 0 | User satisfaction survey | 3 | 2 | 3 |
| Development type | 2 | 2 | 1 | Meet stated objectives | 3 | 2 | 2 |
| Application type | 2 | 2 | 1 | Meet business requirements | 0 | 0 | 2 |
| Package customization | 1 | 0 | 0 | Quality of functionality | 2 | 0 | 0 |
| Architecture | 2 | 2 | 1 | Quality of documentation | 0 | 0 | 0 |
| Client-server | 1 | 0 | 1 | Ease of use | 1 | 0 | 2 |
| Development platform | 2 | 2 | 1 | Training given | 0 | 1 | 2 |
| Language type | 1 | 0 | 1 | Speed of defining solution | 0 | 0 | 0 |
| Portability requirements | 2 | 0 | 1 | Speed of providing solution | 0 | 0 | 2 |
| Used methodology | 2 | 1 | 1 | Process improvement program | 0 | 0 | 0 |
| Project management tools | 2 | 2 | 1 | Project objective A: all functionality | 0 | 0 | 0 |
| Debugging tools | 0 | 1 | 0 | Project objective B: minimum defects | 1 | 0 | 0 |
| Testing tools | 1 | 0 | 1 | Project objective C: minimum cost | 0 | 0 | 2 |
| Performance monitoring tools | 0 | 0 | 0 | Project objective D: shortest time | 0 | 0 | 0 |
| CASE tool used | 1 | 1 | 1 | Build products | 2 | 2 | 2 |
| Upper CASE used | 1 | 1 | 2 | | | | |
| * Dependent variable: TD – total defects delivered, DR – defect rate, 0D – zero defects | | | | | | | |

There are only four variables with rating of '3' – *functional size*, *project user involvement*, *user satisfaction survey*, *meet stated objectives*. All of them have such high rating for *total defects delivered*, only one – *user satisfaction survey* – for *zero defects*. There is no single variable with such rating for *defect rate*.

For *total defects delivered* there are 13 variables with rating of '2', for *defect rate* eight such variables, and for *zero defects* – eleven. Further, the highest total number of variables with rating of '0' is for *defect rate*.

One of the aims of the past analysis [17] was to identify the factors in relationship with defect rate. That analysis also involved using ISBSG dataset, but its earlier edition and with standard set of variables. In that analysis the following variables were identified as related with *defect rate*: *project elapsed time*, *summary work effort*, *architecture*, *CASE tool used*, *client-server*, *development platform*, *intended market*, *used methodology*, *user base: locations*. In the current analysis, we did not use some of these variables. However, nearly all variables from the past study, which we also used in the current study, were confirmed as related with defect rate. The only exception was *client-server* – but the significance level of U for this variable was at $p<0.051$, so only slightly exceeded assumed level.

In the past study, there were several variables for which no confirmation of relationship with defect rate was found. The current study confirmed statistically significant relationship with two of these variables: *development type* and *application type*. However, in the previous study, *application type* was used as a single-response variable, while in the current as a multiple-response variable, for which we created a set of dummy Boolean variables.

Jones provides a range of results on factors of software quality [9]. Those results are based on various sources of data, including ISBSG, but mainly refer to US software industry. Even though the author does not provide detailed methodology on how the data were gathered, aggregated and analyzed (it is not purely research-oriented book), the results are useful and widely cited in software engineering literature. They often are consistent with the results of our experiment. Specifically, the author confirms the similar levels of relationships between software quality variables and *functional size*, *application type*, *activity test*, *testing tools*. However, in contrast with our results, the author states no impact of *project management tools* and *CASE tool used* neither on reducing potential defects nor removing defects. Additionally, the author suggests the increase of *defect rate* together with the increase of *functional size*. Our experiment did not confirm such relationship.

Finally, Jones provides a list of factors which strongly influence software quality but which were not used in our analysis. These factors include: *Team Software Process* (TSP), *Personal Software Process* (PSP), *Six-Sigma for software*, *code inspections*, *Quality Function Deployment* (QFD), *reusable code*, *reusable design*, *design inspections* or *service-oriented architecture*. In fact some of these variables are included in the extended ISBSG dataset, but we did not use them because of very high number of missing values.

## 5. Limitations and threats to validity

The first problem is related with the fact that the ISBSG dataset contains missing data. In fact, there is no single project with no missing data. This may cause the difficulties in the analysis and the interpretation of results.

We selected to remove missing values pairwise rather than casewise. Thus, in the analysis of correlations, a correlation measure between a pair of variables was calculated using all valid cases for those two variables. As a result, a correlation measure between one pair of variables was often calculated using different cases than for another pair of variables.

Also due to missing data, some states of nominal variables have very few cases for calculating categorized *total defects delivered* and *defect rate*. We only used those states, for which the number of cases was at least five. We did not consider other cases, but it does not mean that they do not exist in reality.

Because our data did not meet assumptions for using statistical parametric measures, we used non-parametric measures, which have fewer assumptions. However, such measures have lower explanatory and generalization power.

Although the dataset contains very high number of variables, we did not use several of them. Partially, it was because they contain lots of missing data and might cause problems discussed above. Additionally, some variables and/or their states are not adequately explained what might cause incorrect interpretation of results. An example of such variable is *intended market* that might influence at least two dependent variables, defect rate and zero defects, but we did not use it in analysis due to unclear meaning of its states. Furthermore, the dataset contains data on effort breakdown between development phases. However, these values very often do not sum up to the value provided for *summary work effort* and the dataset does not provide enough explanation of this discrepancy. Thus, we did not use also these variables in the analysis.

Finally, the number of defects provided in the dataset refers only to defects reported in the first month after release. Thus, it is not truly the total number of defects because other defects might have been reported after the first month. However, such data are not available in the dataset.

## 6. Conclusions and Future Work

In this study, we investigated relationships between software quality, expressed by three dependent variables: *total defects delivered*, *defect rate* and *zero defects*, and a set of variables describing software projects. Obtained results led us to making the following main conclusions:

1. There are very few factors significantly influencing these three aspects of software quality, the fewest of them were identified for *defect rate*. Such results might suggest that software quality depends rather on a wider set of factors, possibly also factors not listed in analyzed dataset, and on various combinations of these factors.
2. Performed analysis did not confirm the efficiency of selected initiatives aimed at software quality improvement. Most surprisingly, the priority level for project objective in delivering software with minimum defects had almost no influence on software quality.
3. There is a wide range of techniques and measures that can be used in analysis of relationships between variables, including the factors of software quality. Based on performed analysis we provide aggregated ratings for each variable. These ratings

express the strength of relationship of particular variables with each aspect of software quality.

We believe that provided ratings, as well as detailed results of performed analysis might be valuable for research community and practitioners who are active in areas of software quality or project management.

In future, we plan to extend the analysis discussed here. The extension might involve using other techniques and measures of relationships as well as analysis of relationships of various combinations of factors on software quality.

## 7. Acknowledgment

I am indebted to Professor Norman Fenton from the Queen Mary, University of London, for funding the ISBSG dataset, and the ISBSG Organization for gathering and sharing the data.

## References

[1] Agrawal M., Chari K., Software Effort, Quality, and Cycle Time: A Study of CMM Level 5 Projects, *IEEE Transactions on Software Engineering*. **33**, 3, 2007, 145–156.

[2] Azzeh M., Neagu D., Cowling P. I., Fuzzy grey relational analysis for software effort estimation, *Empirical Software Engineering*, **15**, 1, 2010, 60-90.

[3] Briand L.C., Freimut B., Vollei F., Assessing the Cost-Effectiveness of Inspections by Combining Project Data and Expert Opinion, in: *Proceedings of the 11th International Symposium on Software Reliability Engineering (ISSRE '00)*, IEEE Computer Society, Washington, DC, USA, 2000, 124.

[4] Clark P., Niblett T., The CN2 Induction Algorithm, *Machine Learning*, **3**, 4, 1989, 261-283.

[5] Fenton N.E., Pfleeger S.L., *Software Metrics. A Rigorous and Practical Approach*, PWS Publishing Company, Boston, 1997.

[6] Hall T., Fenton N., Implementing Effective Software Metrics Programs, *IEEE Software* **14**, 2, 1997, 55-65.

[7] *ISBSG Repository Data Release 11*, International Software Benchmarking Standards Group, 2009, www.isbsg.org.

[8] ISBSG, *ISBSG Comparative Estimating Tool V4.0 – User Guide*, International Software Benchmarking Standards Group, 2005, www.isbsg.org.

[9] Jones C., *Applied Software Measurement: Global Analysis of Productivity and Quality*, Third Edition, McGraw-Hill, New York, 2008.

[10] Kan S. H., *Metrics and Models in Software Quality Engineering*, Addison-Wesley, Boston, 2003.

[11] Liu Q., Qin W. Z., Mintram R., Ross M., Evaluation of preliminary data analysis framework in software cost estimation based on ISBSG R9 Data, *Software Quality Journal*, **16**, 3, 2008, 411-458.

[12] Lyu M., *Handbook of software reliability engineering*, McGraw-Hill, Hightstown, NJ, 1996.

[13] Maxwell K.D., *Applied Statistics for Software Managers*, Prentice Hall PTR, Upper Saddle River, NJ, 2002.
[14] Mendes E., Lokan C., Replicating studies on cross- vs single-company effort models using the ISBSG Database, *Empirical Software Engineering*, **13**, 1, 2008, 3-37.
[15] Munson J.C., Nikora A.P., Sherif J.S., Software faults: a quantifiable definition, *Advances in Engineering Software*, **37**, 5, 2006, 327-333.
[16] *Orange*, Laboratory of Artificial Intelligence, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, 2010, http://orange.biolab.si.
[17] Radliński Ł., Fenton N., Marquez D., Estimating Productivity and Defect Rates Based on Environmental Factors, in: *Information Systems Architecture and Technology: Models of the Organisation's Risk Management*, Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2008, 103-113.
[18] Radliński Ł., Predicting Defect Types in Software Projects, *Polish Journal of Environmental Studies*, **18**, 3B, 2009, 311-315.
[19] Robinson B., Francis P., Ekdahl F., A defect-driven process for software quality improvement, in: *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM '08)*, ACM, New York, NY, USA, 2008, 333-335.
[20] Schulmeyer G.G., McManus J.I. (eds.), *Handbook of Software Quality Assurance*, Prentice Hall PTR, Upper Saddle River, NJ, 1999.
[21] StatSoft, Inc., *STATISTICA (data analysis software system)*, version 9.1, 2010, www.statsoft.com.