Research Article

Analysis of Environmental Factors for Mobile Software Development Focused on Korean Companies

Kwang Yoon Song,1 In Hong Chang,1 and Hoang Pham2

1Department of Computer Science and Statistics, Chosun University, Gwangju 61452, Republic of Korea
2Department of Industrial and Systems Engineering, Rutgers University, Piscataway, NJ 08855-8018, USA

Correspondence should be addressed to In Hong Chang; ihchang@chosun.ac.kr

Received 16 March 2021; Revised 13 April 2021; Accepted 22 April 2021; Published 3 May 2021

Copyright © 2021 Kwang Yoon Song et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Software should be a product that can be used easily and accurately by the user and should be improved quickly and accurately when problems arise. In many software development projects, software requirements are frequently modified during the design or development phase, which tests the development of specific designers’ or developers’ capabilities. Software development environmental factors (SDEFs), such as differences in mutual work recognition among users, developers, and testers or knowledge differences, can hinder communication, which may lead to faulty development owing to erroneous job definition. Because the exact size and scope of the software cannot be calculated, the risk of excessive requirements, such as schedule, cost, and manpower, may increase. This study aims to investigate 32 SDEFs to examine the influence of factors affecting the reliability of software developed by Korean companies to identify factors with high influence and compare differences with previous studies. Moreover, we found whether any new SDEFs from the top 10 rankings only affected Korean companies, and also US companies included in previous studies. A factor analysis revealed several potential factors to identify the mutually independent characteristics of the factors. Through statistical analysis methods, the difference between the group means and the impact on improving the software reliability were found in Korean companies. These findings can provide useful benefits to software developers and managers working in countries with different or similar cultures and help increase working efficiency, i.e., work versus time investment and software reliability improvement.

1. Introduction

When using software for personal use, the schedule or quality of development is not a concern. However, good software that is meant for sale to the general public should be user-friendly, error-free, efficient, reliable, and utilitarian to satisfy its users. Software systems that offer user convenience must improve solutions for immediate problems in multiple industries. Software development becomes a more complex process as the requirements for system quality increase. With more elaborate software projects and pressing deadlines, software applications have also become larger and more integrated. Projects and their applications require interaction between diverse individuals in different roles to deliver projects on time, within budget, and while adhering to feature and functional requirements [1]. Software testing is indispensable to ensure quality, wherein critical contributions to strategic objectives set by the firm are met by the information technology (IT) unit [2]. However, the continuing mismatch among the goals of testers, developers, and the IT unit is large, and threatens the firm’s corporate success [3]. Paternoster et al. [4] conducted a systematic mapping study by developing a classification schema, ranking the selected primary studies according to their rigor and relevance and analyzing the reported software development work practices in start-ups. Turk et al. [5] explained the integrated agile development environment to identify underlying assumptions and limitations. Currently, many software companies are working hard to match the advantages of open-source software.
1.1. Motivation. As the era of the 4th Industrial Revolution arrives, the importance of software is increasing day by day, and it is a field of interest not only for global companies but also for all countries. Recently, many companies and countries have been investing in various fields to focus on software development. In addition, noting the importance of investing in research and development (R&D), the Organization for Economic Co-operation and Development (OECD) countries and other nations have been increasing support for R&D as part of their science, technology, and industrial policy. As technological development relies on R&D, governments have been leveraging institutions and policies to expand such investments in the public and private sectors to achieve qualitative improvements. In Table 1, looking at the gross domestic product (GDP) by country in 2019, the GDP is $21.4 trillion dollars for the USA, China, Japan, Germany, and India, respectively. In comparison, Korea’s GDP is relatively small at $1.6 trillion dollars. Moreover, looking at the GDP by country in 2018, the GDP is $20.6 trillion dollars for the USA, China, Japan, Germany, and UK, respectively. In comparison, Korea’s GDP is again relatively small at $1.7 trillion dollars [6]. Table 2 shows the gross domestic spending on R&D [7]. It shows that the US has higher R&D investment than Korea and Japan, but their investment ratio to GDP is the lowest. Korea has a lower total investment than the US and Japan, but the rate of investment and growth relative to GDP has been rising rapidly [8]. Many countries are making efforts to increase R&D investments in the private and public sectors.

Apple has once again risen to be the world’s largest technology company in 2020, according to the Forbes magazine. Despite Korea’s small GDP, there are relatively large number of Korean companies on the world’s top companies list, such as Samsung Hyundai Motor, SK Hynix, POSCO, and LG Electronics. This has been made possible not just because of hardware technology but because companies realized the importance of software development and considered the factors affecting the software development process. Samsung is one such example. Samsung's first smartphone saw a tremendous improvement in terms of hardware. However, when it was launched in an awkward fashion, it had the disadvantage of being the worst smartphone because of software optimization issues. It is important to ensure natural usability through software. Samsung is also making attempts to develop new software to lead the global smartphone industry. In addition, generally, we encountered various problems in software development. To overcome the problems arising in the process of software development, developers have used various approaches [9], and the development of communication technology is one such approach. In this study, we investigate the cultural differences, human cooperation, and tactics aimed at reducing the temporal distance and not the technical aspects. Considering the current status of the mobile content industry, the global app store consumer spending in 2019 was $120 billion, an increase of 2.1 times compared to 2016, and mature markets such as the US, Japan, Korea, and the UK have been driving the growth of app store consumer spending. In addition, the main factors include increased consumer awareness in the mobile industry, increased mobile device users and time spent on mobile devices, advertising revenue from apps, increased influence of mobile games, and increased in-app spending based on subscriptions. Games account for 72% of the app store consumer spending, while subscription services drive the spending growth for nongame apps. In 2019, the number of global app store app downloads (Google Play) was 204 billion, an increase of 45% compared to 2016 and an increase of 6% compared to the previous year. The number of mobile subscribers worldwide is expected to increase by 1.9% per year from 5.1 billion in 2018 to 5.8 billion by 2025, and the growth of the mobile industry is expected to continue until 2025 [10].

1.2. Research Objectives. Alexy et al. [11] found that the support for organizational innovation that allows commercial engagement with open-source software has varying impacts on the technical and administrative dimensions of different job roles through a mixed-method research design. They found that individual-level attributes can counterbalance job role changes that weaken support for adopting open-source software, while perceived organizational commitment has no effect. Ropponen and Lyytinen [12] examined how risk management and environmental factors, such as the development methods and managers’ experience, influence risk components. Their analysis results show that awareness of the importance of risk management and systematic practices to manage risks has an influence on risks related to scheduling, requirements management, and personnel management. Zhang and Pham [13] defined software development environmental factors (SDEFs) related to the software development process. In addition, the study conducted a survey to investigate rankings based on the factors’ influence on software reliability assessment, and also conducted exploratory analyses and presented various opinions about improvement of software reliability. Opinions were presented through additional investigations and analysis results to examine the changes made by software development practitioners. In many software development projects, software requirements are frequently modified during design or development, which tests the development of specific designers’ or developers’ capabilities. SDEFs, such as differences in mutual work recognition among users, developers, and testers, or knowledge differences, can hinder communication, which may lead to faulty development owing to erroneous job definition. Because the exact size and scope of the software cannot be calculated, the risk of excessive requirements, such as schedule, cost, and manpower, may increase.

The objectives of this study are as follows:

(1) This study investigates 32 SDEFs to examine the influence of factors affecting the software reliability of Korean companies to identify factors with high influence.
We find the most important SDEFs for Korean companies and find the correlations between specific SDEFs.

A factor analysis is conducted to identify the mutually independent characteristics of the factors by finding several potential (correlated) factors in the SDEFs, and the differences in group means are investigated through analysis of variance (ANOVA) in the statistical model. His study also uses regression analysis to describe the impact of configuration on improving software stability and analyzes the stages of the development lifecycle.

Finally, we compare the results with previous studies to find the characteristics that only affect Korean companies and discuss the causes of the differences.

Section 2 presents the details of the data collected. Section 3 presents the results of various analyses (factor, correlation, regression analysis, etc.) to examine the diversity of Korean companies. Section 4 compares the new findings related to Korean companies with the results of previous studies. It also explains the causes for the existence of differences and similarities in software development. Section 5 concludes the study and discusses the differences.

### 2. Data Collection

Likert [14] introduced the Likert scale and technique, wherein an individual is invited to define his attitude toward each statement by choosing among a number of r grades on the r-grade Likert scale. The most popular are the five-grade and seven-grade Likert scales. We used the survey by Zhang and Pham [13], which included 32 SDEFs and background information of the software developer. In the survey form [13], 1 indicates “not important,” i.e., it has almost no effect. Conversely, 7 indicates “most important,” which implies it has a significant impact on software reliability. The data were collected from a formal survey and Internet (e-mail) survey served directly to the managers, system engineers, and software developers of 11 organizations (mobile software development, general software development companies etc.) in Korea; 89 survey responses were obtained, among which, 75 were used, excluding a few surveys (written incorrectly or where all values are the same), which were deemed unfit for analysis.

R, SPSS V.24, and Excel 2014 were used for the analysis. Table 3 summarizes the demographic data of the participants. The sample had a relatively good mixture of software development experience, size diversity, and program categories, and also represented a good mixture of software development forms. The average number of years of software development was 8.41 and the average percentage of reused code was 49.13.

### 3. Results

#### 3.1. Relative Importance of SDEFs

**Table 1: GDP status of the world’s top 12 countries.**

| No. | Country    | 2017 (millions) | 2018 (millions) | 2019 (millions) |
|-----|------------|-----------------|-----------------|-----------------|
| 1   | United States | 19,519,353.69   | 20,580,159.78   | 21,433,226.00   |
| 2   | China      | 12,310,409.37   | 13,894,817.55   | 14,342,903.01   |
| 3   | Japan      | 4,866,864.41    | 4,954,806.62    | 5,081,769.54    |
| 4   | Germany    | 3,682,602.48    | 3,967,767.53    | 3,861,123.56    |
| 5   | India      | 2,652,754.69    | 2,713,165.06    | 2,868,929.42    |
| 6   | United Kingdom | 2,666,229.18  | 2,860,667.73    | 2,829,108.22    |
| 7   | France     | 2,595,151.05    | 2,787,863.96    | 2,715,518.27    |
| 8   | Italy      | 2,062,831.05    | 2,091,544.96    | 2,003,576.15    |
| 9   | Brazil     | 1,961,790.20    | 1,885,482.53    | 1,839,758.04    |
| 10  | Canada     | 1,649,878.05    | 1,716,262.62    | 1,736,452.63    |
| 11  | Russia     | 1,574,199.39    | 1,669,583.09    | 1,699,876.58    |
| 12  | Korea      | 1,623,901.50    | 1,724,845.62    | 1,646,739.22    |

**Table 2: Gross domestic spending on R&D of the world’s top 10 countries.**

| No. | Country     | 2016    | 2017    | 2018    |
|-----|-------------|---------|---------|---------|
| 1   | United States | 511,296.7 | 2.76    | 533,313.1 | 2.81    | 551,517.7 | 2.83    |
| 2   | China       | 398,951.6 | 2.10    | 429,092.4 | 2.12    | 462,577.6 | 2.14    |
| 3   | Japan       | 163,035.0 | 3.16    | 169,213.4 | 3.21    | 173,313.4 | 3.28    |
| 4   | Germany     | 117,109.8 | 2.94    | 125,175.5 | 3.07    | 129,647.1 | 3.13    |
| 5   | Korea       | 79,375.4  | 3.99    | 88,147.7  | 4.29    | 954,61.7  | 4.53    |
| 6   | France      | 61,093.6  | 2.22    | 61,962.0  | 2.20    | 627,71.3  | 2.19    |
| 7   | United Kingdom | 46,847.2 | 1.66    | 48,308.3  | 1.68    | 503,73.0  | 1.73    |
| 8   | Chinese Taipei | 35,398.5 | 3.15    | 37,975.5  | 3.28    | 411,04.7  | 3.46    |
| 9   | Russia      | 38,947.8  | 1.10    | 39,921.0  | 1.11    | 362,51.8  | 0.98    |
| 10  | Canada      | 27,847.6  | 1.73    | 27,735.5  | 1.67    | 264,97.1  | 1.56    |
and 0 points indicating the least important. The relative weighting method with each score was applied to determine the ranking [13]. Table 4 shows the results of analyzing the ranking of 32 software development SDEFs using the relative weighting method. Typically, an environmental factor with a higher normalized weight has a more significant impact on the software reliability assessment than the factor that has a lower weight. Therefore, software developers should pay more attention to SDEFs with higher normalized weights. As can be seen in Table 4, in Korean companies, the normalized weight value of factor f11 (requirement analysis) is the highest at 0.03667, followed by f15 (programmer skill) at 0.03634, f12 (relationship of detailed design to requirement) at 0.03578, f20 (human nature) at 0.03449, f22 (testing effort) at 0.03449, and f24 (testing methodologies) at 0.03449.

3.1.2. Correlation Analysis between SDEFs. This section describes a correlation analysis performed to examine the correlation between SDEFs. The Pearson correlation coefficient (Pearson’s r) is a measure of the linear dependence between two variables, where 1 is the total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation [15]. Table 5 shows the correlation of the top-10 SDEFs with high normalized weight values in Table 4. Correlation analysis was performed using 32 SDEFs; only the SDEFs that were significant in the 10 SDEFs shown in Table 5 were tabulated and the other SDEFs were excluded, and only those factors with a strong correlation were summarized ($r > 0.4$ or $r < -0.4$). As a fragmentary example, f11 (requirements analysis) and f12 (relationship of detailed design to requirement), and f11 (requirements analysis) and f13 (work standards) have a strong quantitative linear relationship.

3.2. Environmental Factor Analysis

3.2.1. Factor Analysis. Factor analysis was conducted to verify the construct validity of the questionnaire. Construct validity is determined by the suitability of the items included in the questionnaire to evaluate the set of hypothesized constructs. Factorial validity is a construct validity that proves the validity through factor analysis. Therefore, factor analysis was conducted to verify the validity of the items related to the improvement of the software reliability assessment [16]. Factor analysis was performed using the principal component method, and the varimax method was used as the orthogonal rotation method. The eigenvalues of the components, proportion of variance, and cumulative proportion of variance are presented in Table 6, and 70.973% of the variation can be explained by the first four components. The first to fourth eigenvalues are greater than 1; however, the value after the fifth eigenvalue is less than 1. Those with eigenvalues less than 1 are not considered to be stable. They account for less variability than a single variable.

Table 3: Demographic data.

| Demographic factor                  | Subfactor     | Sample size | Percent |
|-------------------------------------|---------------|-------------|---------|
| Application development             | Safety-critical| 10          | 13.3    |
|                                     | Commercial    | 22          | 29.3    |
|                                     | Inside user-oriented | 32          | 42.7    |
|                                     | Other         | 11          | 14.7    |
|                                     | Total         | 75          | 100.0   |
| Type of software development experience | Database     | 13          | 17.3    |
|                                     | Operation system | 8           | 10.7    |
|                                     | Communication control | 11          | 14.7    |
|                                     | Language processor | 16          | 21.3    |
|                                     | Other         | 27          | 36.0    |
|                                     | Total         | 75          | 100.0   |
| Title/Position                      | Manager       | 11          | 14.7    |
|                                     | System engineer | 10          | 13.3    |
|                                     | Programmer    | 22          | 29.3    |
|                                     | Tester        | 11          | 14.7    |
|                                     | Administrator | 11          | 14.7    |
|                                     | Other         | 10          | 13.3    |
|                                     | Total         | 75          | 100.0   |
| Number of years of software development | Less than 5 years | 29          | 38.7    |
|                                     | 6–10 years    | 22          | 29.3    |
|                                     | More than 11 years | 24          | 32.0    |
|                                     | Total         | 75          | 100.0   |
| Percentage of time spent (%)        | Analysis      | 75          | 21.33   |
|                                     | Design        | 75          | 25.33   |
|                                     | Coding        | 75          | 30.40   |
|                                     | Testing       | 75          | 22.93   |
|                                     | Total         | 75          | 100.0   |
| Average of number of years of software development (years) | 75 | 8.41 |
| Reused code (%)                     | 75            | 49.13       |

4 Mobile Information Systems
does, and hence, they are not retained in the analysis. In this sense, we end up with fewer factors than the original number of variables [17]; hence, the first four components are retained. As a result, the selected factors are equal to the top 13 most important SDEFs based on the ranking results presented by the relative weighting method in Table 4. We can observe that the SDEFs cover many aspects of the development and testing of software, and the majority of these factors have a similarly significant impact on software reliability.

As shown in Table 7, we retained four principal components: PCA1, PCA2, PCA3, and PCA4. The first component, PCA1, consisted of factors for f24, f25, and f22, which are related to testing ability, whereas PCA2 consists of factors for f15, f19, f20, f7, and f18, which are related to programmer ability. PCA3 consisted of factors for f11, f12, and f13, whereas PCA4 consisted of factors f2 and f6, which were related to requirements, standards, and program utility, respectively. All four components had high loading values, which exceeded 0.5. Several methods have been proposed as a measure of reliability, but in this study, Cronbach’s α test was used to test the internal consistency of the items. Cronbach’s α typically increases as the intercorrelations among test items increase and is, thus, known as an internal consistency estimate of the reliability of test scores. If the Cronbach’s α coefficient is more than 0.6, the scale is reliable [18]. As can be seen in Table 7, Cronbach’s α is 0.786, which is greater than 0.6.

### 3.2.2. **Hypothesis Test.**
A statistical hypothesis is a method of statistical interference that is testable based on observing a process that is modeled via a set of random variables [19]. Hypothesis tests are used to determine the outcomes of a study that would lead to a rejection of the null hypothesis for a pre-specified level of significance. The process of distinguishing between the null hypothesis and the alternative hypothesis is aided by identifying two conceptual types of errors, such as type 1 and type 2, and by specifying parametric limits on, for instance, how much type 1 error will be permitted. The null hypothesis ($H_0$) is a simple hypothesis related to the contradiction, with the theory to be proved. The alternative hypothesis ($H_1$) is a hypothesis related to the theory to be proved. The hypothesis will reveal whether their opinion can be considered "the same."

**Hypothesis 1.** People developing software for different applications have the same opinion on the importance of the four components of the new principle. Safety-critical,
Hypothesis 2. People with different types of software development experience have the same opinion on the significance of the four components of the new principle. The database, operation system, communication control, language processor, and others are considered.

\[ H_0: \mu_d = \mu_o = \mu_c = \mu_l = \mu_0, \]
\[ H_1: \text{atleast on } \mu_i \text{ is different.} \quad (2) \]

Hypothesis 3. People playing different numbers of years of software development have the same opinion on the significance of the four components of the new principle. Less than 5 years, 6–10 years, and more than 11 years are considered here.

\[ H_0: \mu_{<5} = \mu_{6-10} = \mu_{>11}, \]
\[ H_1: \text{atleast on } \mu_i \text{ is different.} \quad (3) \]

Hypothesis 4. People playing different roles in software development have the same opinion on the significance of the four components of the new principle. Managers, system engineers, programmers, testers, administrators, and others are considered here.
Table 6: Eigenvalue of the correlation matrix.

| Components | Eigenvalue | Proportion of variance | Cumulative proportion |
|------------|------------|------------------------|-----------------------|
| 1          | 4.041      | 31.082                 | 31.082                |
| 2          | 2.682      | 20.631                 | 51.713                |
| 3          | 1.393      | 10.717                 | 62.430                |
| 4          | 1.111      | 8.543                  | 70.973                |
| 5          | 0.835      | 6.420                  | 77.393                |
| 6          | 0.730      | 5.618                  | 83.011                |
| 7          | 0.506      | 3.896                  | 86.907                |
| 8          | 0.499      | 3.841                  | 90.747                |
| 9          | 0.359      | 2.758                  | 93.505                |
| 10         | 0.300      | 2.310                  | 95.814                |
| 11         | 0.265      | 2.041                  | 97.856                |
| 12         | 0.149      | 1.147                  | 99.003                |
| 13         | 0.130      | 0.997                  | 100.000               |

Table 7: New principles for each principal component.

| Component | New principle | Factor (name)                     | Loading         | Rotation sums of squared loadings | Cronbach's α |
|-----------|---------------|-----------------------------------|----------------|----------------------------------|--------------|
| PCA1      | Testing ability | f24 (testing methodologies)      | 0.898          | 2.764                            | 0.786        |
|           |               | f25 (testing coverage)           | 0.895          | 21.262                           |              |
|           |               | f22 (testing effort)             | 0.857          |                                  |              |
|           |               | f15 (programmer skill)           | 0.805          |                                  |              |
|           |               | f19 (domain knowledge)           | 0.731          |                                  |              |
|           | Programmer ability | f20 (human nature)        | 0.718          | 2.499                            | 0.786        |
|           |               | f7 (programming language)        | 0.618          | 19.225                           |              |
|           |               | f18 (program workload)           | 0.531          |                                  |              |
|           |               | f11 (requirements analysis)      | 0.871          |                                  |              |
| PCA2      | Requirements and standards | f12 (relationship of detailed design to requirement) | 0.827 | 2.202                            | 0.786        |
|           |               | f13 (work standards)             | 0.642          | 16.941                           |              |
| PCA3      | Program utility | f6 (percentage of reused modules) | 0.749 | 13.545                            | 0.786        |
|           |               | f2 (program categories)          | 0.682          |                                  |              |

\[ H_0: \mu_m = \mu_s = \mu_p = \mu_t = \mu_a = \mu_o, \]
\[ H_1: \text{atleast on } \mu_i \text{ is different.} \]  

Hypothesis 5. People playing different numbers of years of software development have the same opinion on the significance of the percentage of reused code. Less than 5 years, 6–10 years, and more than 11 years are considered here.

\[ H_0: \mu_{<5} = \mu_{6-10} = \mu_{>11}, \]
\[ H_1: \text{atleast on } \mu_i \text{ is different.} \]  

We performed ANOVA on hypotheses 1 to 5. ANOVAs are useful for comparing three or more means (groups or variables) for statistical significance. ANOVA is suitable for a wide range of practical problems. The results of the analysis for the given hypotheses are presented in Tables 8–12. As per the results of Hypothesis 1, there was no significant difference according to the group, as shown in Table 8. As per the results of Hypothesis 2, there is a significant difference only in the group of program utility component factors, as shown in Table 9. A post-hoc analysis was performed using Scheffe’s test method; however, there was no significant difference between the groups. As per the results of Hypothesis 3, there is a significant difference only in the programmer ability component factor group, as shown in Table 10. In the post-hoc analysis, there was a significant difference between groups of less than 5 years and more than 11 years. As per the results of Hypothesis 4, there is a significant difference only in the group of program utility component factors, as shown in Table 11. Post-hoc analysis showed no significant differences between the groups. According to the results of Hypothesis 5, there is a significant difference in the percentage of reused code, as shown in Table 12. In the post-hoc analysis, there was a significant difference between groups of less than 5 years and 5–10 years, and less than 5 years and more than 11 years. In Tables 8–12, \( N \) is the frequency, \( M \) is the mean value, \( SD \) is the standard deviation, \( F \) is the variance between groups.
indicating the degree of difference between groups, and \( P \) is the \( p \) value. Figures 1–5 show the results of ANOVA for Hypothesis 1–5.

3.2.3. Regression Analysis. A regression analysis was performed to determine the effect of the four constituent factors obtained through factor analysis on the improvement of the software reliability accuracy. Table 13 shows the correlation between the four component factors in the factor analysis. As can be seen in Table 13, testing ability is correlated with requirements, standards, and program utility, whereas programmer ability is correlated with requirements and standards. Furthermore, requirements and standards have a correlation with all and program utility has a correlation with testing ability and requirements and standards. Regression analysis was performed using a stepwise method to
examine the influence of the constructs on the improvement of software reliability accuracy. Figure 6 shows the correlation results for the four new component factors. Table 14 presents the results of the regression analysis. As a rule of thumb, with respect to the p values for our predictors, we say that the b coefficient is statistically significant if its p value is smaller than 0.05. As can be seen in Table 14, two factors of the four component factors were excluded because the p value was less than 0.05. The b coefficients indicate how many unit component factors
increase for a single unit increase in each predictor. Hence, a 1-point increase in testing ability corresponds to a 0.058-point increase. The beta coefficients allow us to compare the relative strengths of the predictors.

3.3. Analysis of Development Phase

3.3.1. Analysis between Environmental Factor Groups. We want to know how the four software development phases affect each other except the hardware systems phase in the five groups categorized by Zhang and Pham [13]. As a result, there was a significant difference between the four groups, as shown in Table 15. Tukey grouping of the post-hoc analysis was applied to group different development phases based on the mean value of participants’ scores; however, there was no significant difference between each group. The mean score of the analysis and design phase was the highest at 5.545, and the mean score of the coding phase was the lowest at 5.247. The mean scores of the general and testing phases were 5.497 and 5.265, respectively.

3.3.2. Significant Factor for Each Development Phase. In Section 1, we discuss software development projects that are subject to frequent modifications. As a result, SDEFs hinder communication between users, developers, and testers. Thus, every development phase requires data on significant SDEFs for software reliability assessment. This study applies a stepwise elimination method to eliminate nonsignificant SDEFs in each development phase. The variables of stepwise elimination for each development phase were environmental factors in this phase. Table 16 presents the significant environmental factors, parameter estimates, t-values, p values, and so on, for the significant environmental factors in each phase. The results show that significant environmental factors are positively correlated with software reliability improvement. In the analysis and design phase, f14, development management is one of the significant environmental factors. F18, program workload, and f20, human nature, are significant environmental factors in the coding phase. Moreover, f22, the testing effort, is also a significant environmental factor. However, in the general phase, there were no significant environmental factors. F18, program

Table 12: Results of ANOVA for Hypothesis 5.

| Demographic factor       | N   | M     | SD   | F     | P  |
|--------------------------|-----|-------|------|-------|----|
| Percentage of reused code|     |       |      |       |    |
| Less than 5 years        | 29  | 60.69 | 19.073| 12.662| <0.001***|
| 6–10 years               | 22  | 47.72 | 17.164|       |    |
| More than 11 years       | 24  | 36.458| 15.776|       |    |
| Total                    | 75  | 49.133| 20.091|       |    |

a,b: The same group of Scheffe’s test.
Figure 2: Results of ANOVA for Hypothesis 2.

Figure 3: Results of ANOVA for Hypothesis 3.
workload; f20, human nature; and f22, testing effort, have high rankings in Table 3.

4. Comparison

Zhang and Pham [13] conducted a survey in the early 2000s to assess the importance of SDEFs. Recently, Zhu et al. [20] investigated the changes in the impact of SDEFs. This section compares the results of this study with those of previous studies. By comparing with previous studies, we want to provide software developers working in different countries an important environmental factor in software development to increase the efficiency of work (with respect to work versus time investment) on software reliability improvement.

4.1. Comparison of SDEFs Ranking. Since the study uses different population samples from different countries in its three analyses, and the time of the studies is also different, the ranking results differ. It is interesting to see how the ranking of SDEFs according to the relative weighting method changes with time and how it varies according to country. The three results for the top 10 ranks are listed in Table 17. As shown in Table 17, the results of Zhang and Pham [13] and Zhu et al. [20] are similar. This can be studied in detail by Zhu et al. [20]. The results of the 10 new ranks also had similar SDEFs. However, there are three different environmental factors: f20, human nature; f18, program workload; and f7 programming language. First, the working area, working time, vacation, holiday, salary, relationship with boss, and culture are different. In Korea, software
Table 13: Correlation of the four component factors.

|                      | Testing ability | Programmer ability | Requirements and standards | Program utility |
|----------------------|-----------------|--------------------|---------------------------|-----------------|
| Testing ability      | 1               | 0.045              | 0.340**                   | 0.394**         |
| Programmer ability   | 0.045           | 1                  | 0.361**                   | 0.067           |
| Requirements and standards | 0.340**       | 0.361**            | 1                         | 0.330**         |
| Program utility      | 0.394**         | 0.067              | 0.330**                   | 1               |

** \( p < 0.01 \); *** \( p < 0.001 \).

Figure 6: Results of correlation for the four new component factors.

Table 14: Results of regression analysis.

| Name                  | Unstandardized coefficients | Standardized coefficients | t      | R²      | F          |
|-----------------------|----------------------------|---------------------------|--------|---------|------------|
|                       | B                          | Std. error                | Beta   |         |            |
| (Constant)            | -0.009                     | 0.143                     | -0.062 | 0.275   | 13.679***  |
| Testing ability       | 0.058                      | 0.015                     | 0.375  | 3.738*  |            |
| Programmer ability    | 0.071                      | 0.020                     | 0.350  | 3.487** |            |

** \( p < 0.01 \); *** \( p < 0.001 \).

Table 15: Final grouping for development phase.

| Phase                   | Tukey grouping | N  | M      | SD  | F    | P    |
|-------------------------|----------------|----|--------|-----|------|------|
| General                 | A              | 75 | 5.497  | 0.588|      |      |
| Analysis and design     | A              | 75 | 5.545  | 0.793| 3.086| 0.028*|
| Coding                  | A              | 75 | 5.247  | 0.739|      |      |
| Testing                 | A              | 75 | 5.265  | 0.895|      |      |

* \( p < 0.05 \).
engineers belong to several categories. The proportion of engineers who occupy the total economy is a part of the medium/large core product development departments, large IT companies, agencies, enterprises, national laboratories, online/mobile game developers, high-growth start-ups, and others. In the middle/large core product development departments, there are numerous high-class talents, and the working time is long, but the treatment is good. The "IT department" of a company is not a core department but rather a department that is responsible for the computerization of a company. Moreover, since agencies begin with price competition, rather than quality from the outset, it is not easy for engineers to receive good treatment. In the case of enterprises and national laboratories, the quality of human resources is high, but the intensity of work is not. However, remuneration is low. In the case of online/mobile game developers, experienced workers receive good treatment, but the average remuneration is not high, as the percentage of junior (1–3 years after graduation) engineers is high. In the case of "high-growth start-ups" in Korea where competition for securing high-quality talent is fierce, treatment is good—this means that developers are working hard to work in a pleasant environment. In comparison to the US Silicon Valley, the proportion of large IT companies, agencies, and online/mobile game developers is relatively low in Korea. IT/infrastructure management is outsourced to low-cost markets, such as India or China, or resorts to cloud services. On the other hand, high value-added industries, which correspond to the medium/large core product development departments, and high-growth start-ups of large enterprises are adequately developing.

4.2. Comparison of Principal Components. As shown in Table 18, Zhang et al. and Zhu et al. [20] selected the top 11 and 10 SDEFs, respectively. In this study, the top 13 SDEFs were selected for factor analysis. This study applied factor analysis to identify the four common factors. The new

| Phase                   | Significant EFs (name) | Unstandardized coefficients | Standardized coefficients | t      | R² | F      |
|-------------------------|------------------------|-----------------------------|---------------------------|--------|----|-------|
| General                 |                        | B                           | Std. error                | Beta   |    |       |
| Analysis and design     | f14 (development management) | 0.477                        | 0.083                     | 5.771  | 0.114 | 9.349** |
| Coding                  |                         | 0.321                       | 0.095                     | 3.369  |    |       |
| Coding                  | f18 (program workload)  | 0.036                       | 0.015                     | 0.283  | 2.497 | 0.205 | 9.269*** |
| Coding                  | f20 (human nature)     | 0.035                       | 0.015                     | 0.263  | 2.321 |       |
| Testing                 | (Constant)             | 0.368                       | 0.088                     | 4.164  | 0.188 | 16.871 |
| Testing                 | f22 (testing effort)   | 0.063                       | 0.015                     | 0.433  | 4.107 |       |

**p < 0.01; ***p < 0.001.

Table 17: Comparison of the ranking of SDEFs.

| Rank | Zhang and Pham [13] | Zhu et al. [20] | New                  |
|------|---------------------|-----------------|----------------------|
| 1    | f1 (program complexity) | f8 (frequency of program specification change) | f11 (requirements analysis) |
| 2    | f15 (programmer skill) | f22 (testing effort) | f15 (programmer skill) |
| 3    | f25 (testing coverage) | f21 (testing environment) | f12 (relationship of detailed design to requirement) |
| 4    | f22 (testing effort) | f25 (testing coverage) | f20 (human nature) |
| 5    | f21 (testing environment) | f1 (program complexity) | f22 (testing effort) |
| 6    | f8 (frequency of program specification change) | f15 (programmer skill) | f24 (testing methodologies) |
| 7    | f24 (testing methodologies) | f6 (percentage of reused modules) | f18 (program workload(stress)) |
| 8    | f11 (requirements analysis) | f12 (relationship of detailed design to requirement) | f6 (percentage of reused modules) |
| 9    | f6 (percentage of reused modules) | f24 (testing methodologies) | f19 (domain knowledge) |
| 10   | f12 (relationship of detailed design to requirement) | f19 (domain knowledge) | f7 (programming language) |
principal components of this study are well structured by the software development phase compared to other studies.

4.3. Comparison of Significant SDEFs within Each Development Phase. As shown in Table 19, this study and Zhu et al. [20] used the Tukey method to group the four development phases into mean scores, and Zhang et al. [21] used the SNK multiple comparison test. This study and Zhang et al. [21] only comprised one final group; however, they became two separate groups: testing and general phase in Zhu et al. [20]. In this study, the analysis and design phase is the most important development phase in software development; however, the testing phase is the most important phase in the software development phase.

A comparison of significant environmental factors is shown in Table 20. F6, the percentage of reused modules, is the most significant factor from Zhang et al. [21] and Zhu et al. [20] in the general phase; however, it is not the most significant factor in the new results in Table 20. In the analysis and design phase, f14, development management is one of the significant environmental factors. F18, program workload, and f20, human nature, are significant environmental factors in the coding phase. Moreover, f22, the testing effort, is also a significant environmental factor. The reasons for the difference in significant factors in the results.

### Table 18: Comparison of principal components.

| Component | Principle | Factor (name) | Component | Principle | Factor (name) | Component | Principle | Factor (name) |
|-----------|-----------|---------------|-----------|-----------|---------------|-----------|-----------|---------------|
| C1        | Overall   | f21 (testing environment) f22 (testing effort) f5 (level of programming technologies) f12 (relationship of detailed design to requirement) | PC1 Overall | f25 (testing coverage) f21 (testing environment) f22 (testing effort) f24 (testing methodologies) f12 (relationship of detailed design to requirement) f6 (percentage of reused modules) |
| C2        | Testing efficiency | f24 (testing methodologies) f25 (testing coverage) f6 (percentage of reused modules) | PC2 Specification and knowledge | f8 (frequency of specification chance) f19 (domain knowledge) |
| C3        | Requirements and specifications | f11 (requirements analysis) f8 (frequency of specification chance) | PC3 Program complexity and skill level | f15 (programmer skills) f1 (program complexity) |
| C4        | Program and skill level | f15 (programmer skills) f1 (program complexity) | PC4 Program utility | f2 (program categories) f6 (percentage of reused modules) |
of the three papers are due to the difference in working environment, time, and more, as described in Section 4.1.

4.4. Comparison of the Percentage of the Time Allocation for Software Development Phase. In this study, the percentages of time allocation for software development (analysis, design, coding, testing) were 21.33%, 25.33%, 30.40%, and 22.94%, respectively. The percentage of time allocation comparisons is presented in Table 21. Compared to Zhang and Pham [13] and Zhu et al. [20], the percentages of time allocation were 3.7% and 0.7% lower in the analysis phase and 5.6% and 3.6% lower in the coding phase, respectively; however, it was 7.3% and 5.3% higher in the design phase. The time allocation for the testing phase had a similar percentage. Overall, it is observed that there is a small difference in each phase, but it seems that the design phase is more important in Korean companies.

5. Conclusions and Remarks

Software systems that offer user convenience must improve solutions for immediate problems in multiple industries. When using software for personal use, the schedule or quality of development is not a concern. However, good software for sale to the general public should be user-friendly, error-free, efficient, reliable, and utilitarian to satisfy its users. In many software development projects, software requirements are frequently modified during design or development, during which the development of specific designers’ or developers’ capabilities is tested. SDEFs, such as differences in mutual work recognition between users, developers, and testers or knowledge difference, can hinder communication, which may lead to faulty development due to erroneous job definition. This study analyzes the degree of impact of each SDEF on software reliability assessment in Korean companies. This
study investigates 32 SDEFs to examine the impact of factors affecting the software development environment of Korean companies. The results were then compared to those of Zhang and Pham [13], Zhang et al. [21], and Zhu et al. [20]. It is worth noting that these three SDEFs, human nature (rank #4), program workload (rank #7), and programming language (#10) are in the top 10 rankings for the first time based on our survey data collected from Korean companies but have never been included among the top 10 in previous studies from US companies. These findings can provide useful benefits to software developers and managers who are working in countries with different or similar cultures, and how to allocate resources to identify up to date significant SDEFs in software development to increase working efficiency, i.e., work versus time investment in software reliability improvement.

Briefly summarizing the results,

1. Data were collected from formal survey questionnaires and Internet surveys, given directly by software developers or managers, in 11 organizations. Data from 75 surveys were used in the analysis.

2. For Korean companies, the normalized weight value of factor f11, requirements analysis, is the highest, followed by f15, programmer skill; f12, relationship of detailed design to requirement; f20, human nature; and f22, testing effort.

3. We found four principal components using factor analysis. The first component, PCA1, consisted of factors f24, f25, and f22, which are related to testing ability, whereas PCA2 consists of factors f15, f19, f20, f7, and f18, which are related to programmer ability. PCA3 consisted of factors f11, f12, and f13, whereas PCA4 consisted of factors f2 and f6, which were related to requirements, standards, and program utility, respectively. Cronbach’s α is 0.786, which is reliable because it is greater than 0.6.

4. We performed ANOVA on hypotheses 1 to 5. As per the results of Hypothesis 1, there was no significant difference according to the group. As per the results of Hypothesis 2, there is a significant difference only in the group of program utility component factors. A post-hoc analysis was performed using Scheffe’s method; however, there was no significant difference between the groups. As per the result of Hypothesis 3, there is a significant difference only in the programmer ability component factor group. The results of the post-hoc analysis showed a significant difference between groups of less than 5 years and more than 11 years. As per the results of Hypothesis 4, there is a significant difference only in the group of program utility component factors. In post-hoc analysis, there was no significant difference between the groups. According to the results of Hypothesis 5, there is a significant difference in the percentage of reused code. Post-hoc analysis showed significant differences between groups of less than 5 years and 5 to 10 years, and less than 5 years and more than 11 years.

5. As a result of regression analysis to determine whether the four constituent factors affect the software reliability accuracy, it was confirmed that testing ability and programmer ability influence the improvement of software reliability accuracy.

6. In the analysis and design phase, f14, development management is one of the significant environmental factors. F18, program workload, and f20, human nature, are significant environmental factors in the coding phase. Moreover, f22, the testing effort, is also a significant environmental factor. However, in the general phase, there were no significant environmental factors. F18, program workload; f20, human nature; and f22, testing effort, have high rankings based on the relative weighted method described in Table 3.

We compared the results of this study with those of previous studies. First, there are three environmental factors that differ from previous studies: f20, human nature, f18, program workload, and f7, programming language. These factors are influenced by the working area, working time, vacation, holiday, salary, relationship with bosses, and culture. Second, in the past two papers, testing plays a more important role; however, this study shows that analysis and design play a more important role than other phases. Third, as a result of comparing the significant environmental factors, the reason for the difference in significant factors in the results of the three papers is due to the difference in working environment, time, and more. Finally, we compared the time allocation for the software development phase. Overall, it can be observed that there is a small difference in each phase, but it seems that the design phase is more important in Korean companies. These results provide general guidance on important factors to consider for software developers and managers working in a culture or environment similar to a Korean company.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (Grant Nos. NRF-2015R1D1A1A01060050, NRF-2018R1D1A1B07045734, and NRF-2019R1A6A3A01091493).
References

[1] X. Zhang, T. F. Stafford, J. S. Dhaliwal, M. L. Gillenson, and G. Moeller, “Sources of conflict between developers and testers in software development,” Information & Management, vol. 51, no. 1, pp. 13–26, 2014.

[2] J. Dhaliwal, C. G. Onita, R. Poston, and X. P. Zhang, “Alignment within the software development unit: assessing structural and relational dimensions between developers and testers,” The Journal of Strategic Information Systems, vol. 20, no. 4, pp. 323–342, 2011.

[3] C. Onita and J. Dhaliwal, “Alignment within the corporate IT unit: an analysis of software testing and development,” European Journal of Information Systems, vol. 20, no. 1, pp. 48–68, 2011.

[4] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, “Software development in startup companies: a systematic mapping study,” Information and Software Technology, vol. 56, no. 10, pp. 1200–1218, 2014.

[5] D. Turk, R. France, and B. Rumpe, “Assumptions underlying agile software development processes,” 2014, http://arxiv.org/abs/1409.6610.

[6] https://data.worldbank.org/indicator/NY.GDP.MKTP.CD.

[7] https://data.oecd.org/rd/gross-domestic-spending-on-r-d.htm.

[8] H. D. Hong, “Comparative analysis on environmental change of R&D investment and S&T knowledge base in US, Japan and Korea,” The Journal of Korean Policy Studies, vol. 10, no. 3, pp. 487–507, 2010.

[9] E. Carmel and R. Agarwal, “Tactical approaches for alleviating distance in global software development,” IEEE Software, vol. 18, no. 2, pp. 22–29, 2001.

[10] Mobile Content Industry Report 2019. MOIBA. Korea. http://www.moiba.or.kr/main/bbs/info.

[11] O. Alexy, J. Henkel, and M. W. Wallin, “From closed to open: job role changes, individual predispositions, and the adoption of commercial open source software development,” Research Policy, vol. 42, no. 8, pp. 1325–1340, 2013.

[12] J. Ropponen and K. Lyytinen, “Components of software development risk: how to address them? A project manager survey,” IEEE Transactions on Software Engineering, vol. 26, no. 2, pp. 98–112, 2000.

[13] X. Zhang and H. Pham, “An analysis of factors affecting software reliability,” Journal of Systems and Software, vol. 50, no. 1, pp. 43–56, 2000.

[14] R. Likert, “A technique for the measurement of attitudes,” J. Social. Psycholvol. 5, pp. 228–238, 1932.

[15] J. Cohen, Statistical Power Analysis for the Behavioral Sciences, 2nd ed, Lawrence Erlbaum Associates, Hillsdale, NJ, USA, 1988.

[16] R. A. Johnson and D. W. Wichern, Applied Multivariate Statistical Analysis, Pearson Education, London, UK, 2007.

[17] E. R. Girden, Evaluating Research Articles, Sage, London, UK, 2nd edition, 2001.

[18] M. J. Allen and W. M. Yen, Introduction to Measurement Theory, Waveland Press, Long Grove, IL, USA, 2002.

[19] A. Stuart, K. Ord, and S. Arnold, Kendall’s Advanced Theory of Statistics: Volume 2A—Classical Inference & the Linear Model, Arnold, Boston, MA, USA, 1999.

[20] M. Zhu, X. Zhang, and H. Pham, “A comparison analysis of environmental factors affecting software reliability,” Journal of Systems and Software, vol. 109, pp. 150–160, 2015.

[21] X. Zhang, M.-Y. Shin, and H. Pham, “Exploratory analysis of environmental factors for enhancing the software reliability assessment,” Journal of Systems and Software, vol. 57, no. 1, pp. 73–78, 2001.