Optimal Experiment Strategy to Improve Trustability Testing of Management Information Systems

Choe, Hakyoul · Lee, Sangwon · Moon, Songchun · Shin, Kyungshik

Abstract

Failure of Management Information Systems (MIS) could cause loss of enormous development cost and delay of timely reaction to changing business situations. A testing must be performed in order to detect and correct possible errors in advance as well as to verify whether MIS was developed faithfully according to business requirements. To set up criteria for effective testing of MIS, a modeling for trustability to present basic performance of testing is needed. A model on the basis of testing power and trustability is very useful in finding this testing method and combination. To verify the effectiveness of analysis method extracted by a trustability model, a trial experiment where performance comparison is possible under various situations is necessary. So, we research on effective testing in order to resolve the failure of MIS in a preliminary stage. First of all, we point out some defects of both traditional testings related to MIS testing and application to trustability model with implementing a trustability model. An optimal testing model is implemented and proposed on the basis of the trustability model with some configurations for a trial testing. Finally, it is analyzed how this designed trustability model is effective.

Keywords: Trustability, Testing, Error Rate
1. Introduction

Failure of Management Information Systems (MIS) [16] could cause loss of enormous development cost and delay of timely reaction to changing business situations. Many researches estimated failure rates of MIS at 30 ~ 60 percent. If a failure of MIS is confirmed, many losses or corrections of business strategies would certainly arouse. For example, the solution cost used for notation errors of ‘year’ in MIS is estimated at approximately 1.5 trillion dollars. Since banks play a pivotal role of financial industries and most of their business works such account and loan are performed under computerized situations, errors or malfunctions of MIS means shutdown of their works. Especially, notation errors of ‘year’ could cause severe damage to banking in accuracy of interest calculation.

A testing must be performed to detect and correct possible errors in advance as well as to verify whether MIS was developed faithfully according to business requirements. Securing correctness of MIS development ought to be satisfied with ‘function integrity’ [12] which checks whether the results of development are the same as business requirements. These testings include creation of testing data, halt of work for testing, proposition and analysis of testing results, and re-correction. Automation tools that are widely used with major function to create appropriate testing data and to propose result do not support other functions. In fact, the testing period of MIS occupies half of the total development time because many testings should be performed repetitively. Just one testing does not secure complete detection of errors, and the process of error correction would cause another additional errors. A method to improve testing trustability with completeness of error detection and possibility of error non-detection would certainly decrease overly repetitive performances owing to scare trustability. It would also shorten preemptive action time of MIS development owing to limited time.

To improve error-testing power, many independent testings should be performed to decrease probability of error non-detection. So, as the number of target program increases, that of testings grows exponentially. Let us assume a program whose probability of error possession is 1 percent in order to figure out relationships between error-testing power and the number of testings. Then, let the number of testing performance be N. If a repetitive testing secures more than 99 percent of trustability in error-detection probability within a program, we can get 0.99 = 1 - (1 - 0.01)N. So, 461 testings are needed. In enterprise-wide testing with several thousand programs, it is almost impossible to perform a testing with 100 percent of error detection where high trustability is too heavy a burden. However, adding only information about specific methods with widely known detection possibility would decrease the number of necessary testing considerably. If the detection probability is 90 percent, the number of repetitive testings is 230 with 0.99 = 1 - (1 - 0.09)N. This means that the consideration about preliminary information decreases the number of testings.

If this preliminary knowledge is added to Testing Power Model [7] with probability of error detection, the overburden of MIS testing would be decreased. An expansion of the model with testing method could propose an effective combination of testing method and finish testing phases rapidly. But, in case of targeting many methods and programs, to analyzing with the model is not applicable owing to its complexity. A trial experiment using computers is applicable when this analytic approach is not applicable. In general, a trial experiment aims at using Trustability Analysis [3] or Sensitivity Analysis [9] when parameters of probability are designed by assumptions in its model. The former analyzes results through statistical methods, and the latter measures the effects which changes of input parameters result in. Especially, Discrete Event Trial Experiment [9] is applicable to this case for comparing the error rate of discrete probability parameters with the testing power with the methods.

Since the repetitive testing occupies most of the period required in a MIS testing, too heavy a burden due to repetitive testing should be decreased so as to shorten the period of preliminary error detection and settlement. A probability that a testing does not detect an error in an updated program could cause a repetitive testing. So, an error of MIS could be dealt with in advance after decreasing the burden on testing by selecting a testing method to elevate error-testing power and to mix a testing method. A model based on the testing power and trustability is very useful in finding this testing method and combination. To verify the effectiveness of analysis method extracted by a trustability model, a trail experiment where performance comparison is possible under various situations is necessary.

In Section II, we point out some defects of both traditional testings related to MIS testing and application to trustability model. Section III shows the implementation of a trustability model. In Section IV, an optimal testing model is implemented.
on the basis of the trustability model with some configurations for a trial testing. Section V analyzes the trial model about how this designed trustability model is effective. We make conclusions in Section VI.

2. Related Works

The aim of a program testing is to elevate trustability of the program by detecting and deleting some errors. In general, traditional researches could be classified into two methods, Error Detection Method[1][8][11][13][17] and Statistical Method[6][10][14][15]. The former aims at detecting errors and finding out their positions effectively, and the latter researches on the number of testings for sub-programs of main program. To correct an error, information of its position is necessary. Even though traditional testing methods are good at detecting errors, they are poor at finding out error position information when an error exists in another position that is not the point of origin. Information of dependency among programs is used to detect the origin of detected errors by improving traceability with reverse-detecting. There have been many researches on traceability that find out errors not only in the position of detected errors but also in other positions connected with other programs.

2.1 Traceability

To acquire high traceability, many methods use certain features such as mutual dependencies among programs, forms of input parameters and program structures. Error Connection Exploration Method[2] tests a program by using data types that describe how each error is performed and error indicators that point to each position. Semantic-Based Method[11][17] uses semantic dependency with information on control dependencies and data dependencies among programs. Program History Indicator Method[8][13] searches an error by comparing programs with data routes when a difference happens after performing same programs under different situations. While these traceability-related methods are excellent in usability of error detection and correction, they do not support information on how many errors are detected in programs and have limitations of traceability methods.

These methods are short of trustability when an error is detected by a testing. For example, a testing with high traceability corrects almost errors but there is still a probability that some error is not detected. In this case, the trustability declines since the method depends on only traceability and a possibility of error non-detection is neglected. The traceability-oriented testing is affirmative in detecting and correcting errors but negative in improving trustability of programs with focusing on the testing. The result of the testing depends on whether assumptions about errors are analogous to the reality. As the probability of error detection is getting high, the trustability of testing as degree of error minimization is getting high. High probability of error detection could make it easy to correct by use of many traditional traceability methods and so accomplish the goal of error deletion through testing. Verifying the trustability of testing needs a trustability approach that uses a statistical model considering the probability of error possession, in order to identify testing with low probability of error non-detection and improve the performance of testing.

2.2 Trustability

A testing method to acquire trustability is composed of Random Testing(RT) and Partition Testing(PT). It is one of the most controversial in information systems testing how to select the testing cases in order to guarantee some level of confidence on the reliability of software. RT selects testing inputs randomly from the whole input domain and PT divides the input domain into nonoverlapping sub-domains and selects one testing input from each sub-domain. Even though the efficiency of RT and PT has been discussed, analytical and mathematical models[5][7] show that RT is superior. While more testings detect more errors, it is impossible to increase the number of testings indefinitely due to cost and time restraints. Conflicting relationships between the probability of error detection and additional cost required for testing should be considered. To secure high trustability of testing with limited time and cost, it could improve the trustability of testing to distribute the number of testings by use of information on probability of error occurrence that are different in grade by each sub-program. In distributing the number of testings, there are RT[6][15] that distributes the total domain randomly and PT[4] that distributes the total domain with the same size.

The reason why testing is very important in MIS is time and cost restraints. Traditional approaches on trustability have focused on individual features and comparisons of RT and PT, not considering time and cost as important factors for testing to detect errors in MIS. PT assigns testing to each domain after separating a program into sub-domains and uses the whole information of the program such as requirements, de-
development procedures, and features of codes. Preliminary information would assign more testing to a part with high probability of error detection and partition of the target domain of testing would increase the effect of testing. RT performs testing for the whole program without considering this preliminary information and artificial tasks such as separating into sub-domains. RT as an alternative to PT is quoted as saying that it is superior to PT, excepting case where a sub-domain with high probability is defined by error localization. Analytic methods are used to compare RT with PT, whose criteria[6][15] are which testing improves more trustability as the probability of error detection in MIS. The result depends on assumptions in establishing a model by use and comparison of failure density probability and frequency probability of each sub-domain.

The reason why a testing of MIS is very important and difficult is that time to settle problems is limited by regulations and cost is limited. Less time and low cost with 90 percent of trustability would be selected instead of more time and high cost with 99 percent of trustability. Therefore, an optimal testing strategy should be decided with considering trustability of error detection probability and repetitive performance of testing.

3. A Trustability Model of Testing

To set up criteria for effective testing of MIS, a modeling for trustability to present basic performance of testing is needed. Some assumptions of trustability for optimal testing should be considered to cope with rapidly changing business requirements. This section proposes necessary assumptions and a basic trustability model.

Since trustability is the degree of confidence to secure no-error statistically, error-testing power of testing should be used. When there is an error in a program, testing power is defined as the probability for the testing to detect the error and then is generally measured by use of error rate of the program. RT and PT are classified according to the number of testing for domains with high error rate by use of information of error rate in a program, and they are models to evaluate the performance of testing by use of trustability. To implement these trustability models, information of error rate in the program is used. So, setting up accurate error rate of the program is very important for a trustability model.

3.1 Model Assumptions

A program is composed of many modules and is operated through functions among the modules. By this mutual functionality among modules, an error in a module could cause another error in other modules related to the module. It is very difficult to implement a model with considering all these cases because all information about mutual functionality among modules is needed. If possible, its complexity would make the model function theoretically. Since every importance for each module in the program is different, disregarding this importance should make errors of the whole program misunderstood. So, we need some assumptions as follows.

Model Assumption 1(Composition of Program): A program is composed of many modules. If each module is defined as \(D_i\), an independent execution domain, the program is made up of \(D_1, D_2, \ldots, D_n\). Even though the program could have a functional relationship among modules, errors in the domain \(D_i\) do not effect mutually. That is, each domain \(D_i\) is independent.

Model Assumption 2(Rate of Error Possession): Since each \(D_i\) is mutually independent, error rate is mutually different by each domain. Each domain has errors with error rate = \(\theta_i\).

Model Assumption 3(Execution Frequency): In executing a program, each domain does not have the same number of operations. It is reasonable to weigh in each domain in calculating error rate of the whole program. The weight can be calculated by execution frequency of each domain. If the execution frequency is \(f_i\), the error rate \(\theta\) as its weighted average of the whole program is as follows.

\[
\theta = \sum_{i=1}^{n} f_i \theta_i
\]

Model Assumption 4(Testing Power): Since testing is to detect errors, testing power is \(\theta\). Trustability, as the probability that at least one or more error would exist after executing one testing, is \(1 - \theta\) as the probability that any error would not exist. If this procedure is performed \(N\) times repeatedly, the trustability is \(1 - (1 - \theta)^N\) as the probability that at least one error exists in the program.

3.2 Random Testing and Partition Testing

RT is performed for the whole program without distinction of domain. Trustability of RT performed by repetitive execution of testing with error rate \(\theta\) is as follows.

\[
1 - (1 - \theta)^N
\]
In PT, differentiated testing by each domain with preliminary information is performed. Each domain has mutually different error rate, and as many testing as the mutual frequency $n$ shows the trustability as follows.

\[
1 - \prod_{i=1}^{n} (1 - \theta_i) > \sum_{i=1}^{n} n_i = N
\]

4. Optimal Testing Strategy Based on Trustability Model

A standard to select a testing with the highest trustability among testings is established by implementing a trustability model under various situations with PT considering features and preliminary information about the target program. This standard enables it possible to perform an optimal testing with the highest trustability, finding out testing time for maximal effects with the limits of cost and time. This section shows the methods to select and mix through trustability model on the basis of PT.

Trustability of program through testing is able to compare RT and PT. PT improves the whole trustability of testing by adding domain divisions with preliminary information of the program. Performance comparison could guess a profit in comparison with RT that is performed for the program without any additional task. To compare two testings, trustability of each testing should be regarded as performance of each testing. In conducting testings for the same program, testing with more performance could be judged to be superior to others. We can compare between testings by using the trustability rate of each testing.

**Theorem 1 (PT Performance):** The trustability rate of PT for RT could be greatly increased by mixing error rate and frequency.

Proof: The trustability rate of PT for RT is as follows.

\[
\frac{1 - (1 - \theta)^n}{1 - \prod_{i=1}^{n} (1 - \theta_i)}
\]

Let us assume that $\theta_i \neq 0$ with any very small $\theta_i (i \neq k)$ and $\theta_k$. In this case, because the sum of $\theta_k$ is 1, the following $\theta$ is very small.

\[
\theta = \sum_{i \neq k} \theta_i
\]

Also, as $\prod_{i=1}^{n} (1 - \theta_i) (i \neq k)$ approaches to 1, the trustability rate mentioned above is represented as follows.

\[
\frac{\theta}{1 - (1 - \theta)^n}
\]

Here, the performance of PT is very higher than that of RT.

**Theorem 2 (Limited RT Performance):** The trustability of RT is limited to less than $n$ (the number of partitioned domain) as $n$ increases.

Proof: The trustability rate of RT for PT is as follows.

\[
\frac{1 - (1 - \theta)^n}{1 - \prod_{i=1}^{n} (1 - \theta_i)}
\]

If $\theta_{\text{max}}$ is the maximum of the error rate of partitioned domain, the trustability of RT is limited as follows.

\[
1 - (1 - \theta)^n < 1 - (1 - \theta_{\text{max}})^n
\]

And the trustability of PT is larger than $\theta_{\text{max}}$ as follows.

\[
1 - \prod_{i=1}^{n} (1 - \theta_i) > 1 - (1 - \theta_{\text{max}}) = \theta_{\text{max}}
\]

So, the trustability rate of RT for PT is as follows.

\[
\frac{1 - (1 - \theta)^n}{1 - \prod_{i=1}^{n} (1 - \theta_i)} < \frac{1 - (1 - \theta_{\text{max}})^n}{\theta_{\text{max}}}
\]

To get the limit point of $F(\theta_{\text{max}})$ with $F'(\theta_{\text{max}}) = 0$, $F$ should be differentiated.

\[
\theta_{\text{max}} n (1 - \theta_{\text{max}}) ^ {n-1} - (1 - (1 - \theta_{\text{max}})^n) = 0
\]

By the limit point of $F(\theta_{\text{max}})$, we could get the following expression.

\[
(1 - \theta_{\text{max}})^n = \frac{1}{1 + \frac{\theta_{\text{max}} n}{\theta_{\text{max}}}}
\]

Also, at the limited point, the value $F(\theta_{\text{max}})$ is as follows.

\[
\frac{1 - \frac{\theta_{\text{max}}}{1 - \theta_{\text{max}}}}{\theta_{\text{max}} n} = \frac{n}{1 + (n-1)\theta_{\text{max}}}
\]

Since $0 \leq \theta_{\text{max}} \leq 1$, $F(\theta_{\text{max}})$ has the value between 1 and $n (n \geq 1)$. If $\theta_{\text{max}}$ approaches to zero, the $F(\theta_{\text{max}})$ converges to $n$. Hence, the performance of RT is smaller than that of PT by the number of partitioned domains.

\[
\frac{1 - (1 - \theta)^n}{1 - \prod_{i=1}^{n} (1 - \theta_i)} < F(\theta_{\text{max}}) = n
\]

This theorem shows that the performance of RT is limited.
rather than PT. If preliminary information is known, PT is superior to RT with limited performance.

5. Simulation

Until now, we looked into a model to select a testing maximizing trustability of program and to decide an optimal testing-conducting time in limited time and cost. Now, we use a simulation to check how a testing-conducting strategy by used trustability model and economic model is operated concretely in reality. We could also find other information by working out various cases that were not considered in a model. This section configures a simulation model to verify a trustability model and analyzes its result.

5.1 Simulation Assumptions

Assumptions for a simulation are needed to evaluate testing performances of a trustability model, to construct an experimental model for program testing, and to clear the scope and limitation of testing.

Simulation Assumption 1 (Domain Partition): A program is composed of modules for individual testing. Each testing is performed for each module. So, we assume that a module for testing is regarded as a domain and could be partitioned, in order to understand relationships between the number of domains and trustability through comparing performances of trustability model with variations of the number of domains.

Simulation Assumption 2 (Distribution of the Number of Testings): After deciding the number of the whole testing, the whole testing is distributed for each domain with the same number. Since differences of the testing number for each domain discriminate the total error rate for each domain with variations of the testing number, they could make initial conditions unfair and then cause errors in measuring the effects of considered input parameters.

Simulation Assumption 3 (Error Rate for Each Domain): Errors for each domain are mutually different. The error rate for each domain should be different so as to grasp features of PT that performs testings for each domain.

Simulation Assumption 4 (Immediate and Complete Correction): Detected errors are immediately corrected and reflected in the program. Through corrections, the survival rate of errors decreases after testings. Repeated testings and this procedure could decrease the whole error rate of the program and make it possible to secure trustability through testings.

5.2 A Simulation Model

The goal of this testing is to find out an optimal conditions for testing after seeing how trustability changes through various testings in a program with partitioned domains. So, an appropriate simulation model is needed so as to check the method of testing, the number of testing and relationships with the number of partitioned domains in the program. For these goals, a simulation model is composed of four components (Figure 1); Target Program with several domains, Testing Creator to control over testing, Error Logger to write detected errors and related information to correct them in the future and Error Corrector to correct errors.

5.2.1 Program and Partitioned Domains

A program could get high trustability by detecting and correcting errors before final authentication for target testing. A program could be composed of millions of partitioned domains according to circumstances. Even though each domain would have different weight in the program, it has the same weight since one goal of the testing is to check the number of domains and relationship with trustability.

5.2.2 Error Creator

Error Creator plays a role to create errors in each partitioned domain. It puts errors into each domain while maintaining given conditions and the error rate of the whole program. The reason why it creates errors differently in each domain is that only differentiated errors in each domain reflect the features of PT and RT.

5.2.3 Testing Creator
Testing Creator decides a testing and its method to detect errors in the program. Above all, one of PT and RT is selected and how many testings are performed for the program is decided. After deciding the number of total testings, selected testings should be assigned to each domain. The number of testings is equally distributed in order to prevent distortion of results that would occur through density difference in the number of testings for each domain. Reliabilities for each testing method could be compared only after finishing several testings. Since testings consider various circumstances, relationships between trustability and testing environment could be discovered by comparing each case. Through these procedures, a testing method for higher trustability is decided.

In experimenting in Testing Creator, a domain is randomly chosen among all domains with mutually different error rates. Let us test a program with \( n = 10 \) and the number of testing cases from 50 to 600. For each case, we assigned error rate for each sub-domains with random uniform distribution resulting in an expected average failure rate of 0.1. Each test selects some sub-domains with different error rates according to Assumption 3. After a domain is chosen in performing testing, Error Creator checks the error assigned to the domain. By Simulation Assumption 4, the error of the domain is immediately corrected and then the testing is performed again.

### 5.2.4 Error Logger

Error Logger writes detected errors, their locations and functions, and so on after detecting errors. These tasks could make it easy to correct errors.

### 5.2.5 Error Corrector

Error Corrector corrects detected errors by using information about errors that are stored in Error Logger. Correcting detected errors in each domain of the program would certainly improve trustability of the program in the end.

### 5.2.6 Parameters and Performance Indexes

Parameters used in the simulation model are composed with those for the target program and those for the testing. The number of testings is equally distributed in each partitioned domain in the program. It is so because each domain has the same weight according to assumptions. In order that each domain can have a different error rate, random numbers are created and distributed, the range of which is from 0 to 0.2. To increase own trustability of the simulation, 1,000 testings are repeated and its results are averaged. Since the simulation aims at seeing how the number of testings is distributed in each domain for higher trustability with mutually different error rates, the number of testings in each partitioned domain and the error rate of each domain are the most important parameters. The performances of each testing method could be checked by observing how the trustability of each testing changes as the error rate and the number of testings in each domain change.

### Table 1. Parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Definitions</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>TestingNum</td>
<td>The number of testings</td>
<td>50, 100, 200, 300, 400, 500, 600</td>
</tr>
<tr>
<td>DomainNum</td>
<td>The number of partitioned domains</td>
<td>10, 25, 50, 100, 200</td>
</tr>
<tr>
<td>DomainErrRate</td>
<td>Error rate in each domain</td>
<td>0 - 0.2</td>
</tr>
<tr>
<td>DomainTestingNum</td>
<td>The number of testings in each domain</td>
<td>TestingNum/DomainNum</td>
</tr>
<tr>
<td>DomainFreq</td>
<td>Weighted Value in each domain</td>
<td>Equally Distributed</td>
</tr>
<tr>
<td>TestingCycleNum</td>
<td>The repetition number of testings</td>
<td>1000</td>
</tr>
</tbody>
</table>

### Figure 2. NT and RES with Domains = 10

With changing the Rate of Error Survivals (RES) and the Number of Testings (NT) for each domain, RT and PT are performed. RES is a probability that an error exists in the system, and it is in inverse proportion to systems trustability that increases as testings are repeated and errors are corrected; RES = 1 – Trustability. RES varies as variables such as domain, NT and error rate of each domain because the variables...
affect trustability with testing strategies of PT and RT. The numbers of domains vary like 10, 25, 50, 100, and 200, where the error rates for each case are changed like less than 0.01, 0.05, 0.1, and 0.2. The choice of the number of domains is used to compare the small partitions like 10, 25 with the many partitions like 100, 200. In order to compare programs of very low RES with those of very high RES, the maximal limit of RES is constrained. Since trustability exists in inverse relationships with error rate, it can be analyzed through decreasing tendency of RES. In the simulation, testings are performed when there is no information about RES of each domain as well as when there is information that a specific domain has higher RES than others.

5.3.1 Without Preliminary Information

Let us consider the case that the number of domains is 10 among when the program is partitioned into small domains. On the whole, PT has lower RES than RT, and consequently has higher trustability. When the program has low RES such as Figure 2(b), the difference between PT and RT increases as NT increases. When the number of domains is 25, PT shows high performance rather than RT. With high RES such as Figure 3(b) and Figure 3(c), the difference between PT and RT decreases as NT increases. In program with low error rate such as Figure 3(d), the difference 0.17 between 0.6 and 0.77 of the early testing increases to more than 2.0.

In case of more partitioned domains, the results are different from the above ones. Trustability of RT and PT could not be almost found as seen in Figure 4, Figure 5 and Figure 6. In case of higher RES, RES approaches to 0 when the number of domains is more than 50 and when NT is more than 100. In comparison with Figure 4(c), Figure 5(a) and Figure 6(a), high RES in the early stage decreases markedly. It is so because it becomes possible to perform a detailed testing as the number of partitioned domains increases. If the number of domains increases to some degree, the merits of PT using information decrease and PT finally takes on the same aspect as RT. Since the cost is incurred in partitioning a program, it is realistically unnecessary to partition a program into more than hundreds of domains. It would be more rational to expect remarkable improvement of trustability with small partitions of a program. Especially, with small RES due to continuous testings and systematic development, the more effectiveness of information utilization increases, the more testings are performed. Hence, PT with low RES through small partitions is superior to RT.

5.3.2 With Preliminary Information

The comparison between two or four domains with error rate 0.2 or 0.1 and other domains with less than 0.01 is used to find out how much the performance increases. The results with 100 or 200 domains are not quietly different from those without preliminary information. In case of small partitioned domains, the PT performance with preliminary information is obviously superior to that without it. When the number of domains is 10, PT shows remarkable performance. As NT increases, the difference of RES between two testings decreases from 0.24 to 0.1. But, in domains where NT is less than 15 for each domain, Figure 7 shows that both PT and RT are superior. Even in case that the number of partitioned do-
mains increases to 25, PT is superior to RT. PT performs more than 0.2 when NT is less 200. Figure 9 where the number of partitioned domains is 50 shows both RT and PT which nearly make no difference.

![Figure 5. NT and RES with Domains = 100](image)

![Figure 6. NT and RES with Domains = 200](image)

![Figure 7. NT and RES with Domains = 10 after Using Preliminary Information](image)

![Figure 8. NT and RES with Domains = 25 after Using Preliminary Information](image)

![Figure 9. NT and RES with Domains = 50 after Using Preliminary Information](image)

Consequently, PT is superior to RT when the number of partitioned domains is small or RES is comparably high in some domains. Hence, a testing with using preliminary information and then partitioning a target program into scores of domains could certainly secure better trustability by using the same NT. Especially, when it is difficult to perform testings repeatedly due to time limit and when it is possible to perform testings with very small NT, its performance is very excellent.

6. Conclusions

We researched on effective testings to resolve the failure of MIS in a preliminary stage. In testing MIS, the limit of time and cost should absolutely be considered. With the limit of time, a standard to select an outstanding testing for error detection is needed in deleting errors related a program. So, we proposed a standard of trustability as the rate of no-error in a program with a testing method to maximize it and then tried experimental performance evaluations. Consequently, PT with preliminary information about errors in a program gets excellent results when the number of partitioned domains is small and many testings are not permitted.

In further studies, the comparisons between PT and RT are required on the basis of trustability model through simulation under various situations. We performed testings with the upper limit of error rates for simple comparisons, but it is necessary to check what results show with easing the upper limit. Even when preliminary information is used and the number of testings for each domain is differentiated, it should be check whether PT would be still superior by using simulation. Moreover, when domains and the number of testings are limited, algorithms that dynamic planning method can be applicable in reality to maximize trustability should be studied. Heuristic and empirical results are also needed in judging how much it is contributed to select and perform a testing method.
References


저자소개

Hakyoul Choe received his M.S. degree at School of Computer Science, Carnegie Mellon University. He is a Ph.D. candidate in Management Engineering from Korea Advanced Institute of Science and Technology. His current research interests include enterprise data modeling, requirements engineering, XML and information security.

E-mail: pullux@business.kaist.ac.kr
Tel: +82-2-958-3333

Sangwon Lee received his Ph.D. degree in Management Engineering from Korea Advanced Institute of Science and Technology in 2009. He has been working for Ewha Womans University as a research professor at Management Research Center since then. His research interests include data engineering, requirement engineering and knowledge engineering.

E-mail: sangwonlee@kaist.edu
Tel: +82-2-3277-3767

Songchun Moon received his Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign in 1985. He has been working for Korea Advanced Institute of Science and Technology as a professor at Department of Management Engineering since then. His research interests include enterprise data modeling, security, privacy and data warehousing.

E-mail: scmoon@business.kaist.ac.kr
Tel: +82-2-958-3315

Kyung-shik Shin is an associate professor of Management Information Systems, College of Business Administration at Ewha Womans University and head of the Ewha Knowledge & Information Center. He received his MBA from the George Washington University and Ph.D. from Korea Advanced Institute of Science and Technology. His research interests include decision support systems, knowledge based systems & management, virtual collaboration and social network analysis.

E-mail: ksshin@ewha.ac.kr
Tel: +82-2-3277-2799

◇ 이 논문은 2010년 6월 4일 접수하여 2010년 7월 6일 1차 수정을 거쳐 2010년 7월 12일에 게재확정 되었습니다.