INTERNATIONAL JOURNAL OF COMPUTERS
Issue 4, Volume 1, 2007

A Study on the Development of Rules for Effective Code Inspection : Case Study of Company "A" Information System

Taewon Kyung and Sangkuk Kim

***Abstract*—**Inspection is one of the most popular methods to develop high quality software. However, structured implementing method is not well defined, and so actual implementation of inspection is done in arbitral manner. In this research, eight categories and 30 indexes are defined to implement the Inspection in more structural way. Categories and indexes are developed based on the knowledge and guidelines of experienced developers and consulting companies. Suggested rules will clearly lead the programmers to avoid defects in advance or detect errors more easily by following the suggested rules in stricter manner. Proposed rules are also applied to real information case to test the effectiveness of the rules.

***Keywords*—**Inspection, Code-inspection, Walkthrough, S/W Test, Peer-review

I. Introduction

Small size software development projects or information system development projects were not serious issues. However, the scale of software development is getting larger and development time is getting longer, this problem gets more attention from information society. Various ideas and methods have been introduced to solve this problem. For example, during F-22 high-speed fighter plane development project, software development costs were soaring to take more than 80% of total project costs, even the number of involved software developers was not over hundreds person [17] .

Errors and omissions in software are inevitable, for software is not coded under clear or well defined environment. Software development is very much customer needs oriented, and their need, in most cases, changes and evolves as time passes. For this and other various reasons, to meet the customers' needs in software development is not easy task at all. Therefore, quality management of software development is one of most difficult tasks to deal with [11].

Quality of developed software, however, is the most important matter in software development. Quality of software decides the success and failure of the project. Correction of software defects costs time and money. Especially, the cost of corrections during the last period of development is more than 10 to 100 times expensive than that of starting period of system development [3]. So the earlier detection of software defects is top priority issue in system development. One of the popular detecting methods is "Inspection".

Inspection tries to check the possible defects in each and all phases of system development period based on the template, which lists the possible errors and omissions in each development phase. Inspection method was proposed by Michael Fagan in 1976[5], and argued that Inspection alone can detect 60% to 90% of software defects. However, Inspection method also has another virtue to avoid making mistakes in advance by programmers [6]. Russell (1991) reported that Inspection could also reduce maintenance time and costs [15], which is very natural by correcting the mistakes at each stage of system development. However, most software developers and managers do not clearly distinguish the differences between the Inspection and other test methods. Inspection also has no clear or widely accepted agreements between professionals. In this research, an Inspection quality enhancement method is suggested, and be compared to other existing test techniques. Also suggested rules will be applied to real case to check the effectiveness of the method. Company "A" information system, which is running in real world is used as a test bed to check the effectiveness of suggested method.

II. Comparisons of Test Methods

Most popular defect detecting methods are Test, Peer Review, Walkthrough, and Inspection. In this chapter, differences of those four methods are compared.

2.1Software defect detecting techniques

2.1.1 Test

IEEE defines Test as "Experimental and estimation process that verifies whether systems meet specific requirements, and distinguishes the differences between expected result and real result by manual or automated method."[10].

Test comprises three stages: Unit test, Integration test, and Acceptance test. In most cases, unit test is done by programmers who did programming. Major purpose of Unit test is that checking whether the module has proper functionality. Integration test examines all modules that consist of entire system as a whole. It tests the entire system whether interfaces between modules are adequate and the system as a whole can meet the requirements of users. Acceptance test is to check the acceptability of the system by the people who will actually use the system in the future. Purpose of Acceptance test is to unveil the system that is ready to be adopted [1].

2.1.2 Peer Review

Generally, whenever review is needed, peers review the outputs to check the rationality of the system in free, arbitral environment. Peer Review does not have some structured specific formats. Number of reviewers, time limit, actions to be

 Manuscript received May 1, 2007; Revised received October 28, 2007

executed, documents or other structured review methods are not clearly specified. The actual review process is similar to communicate ideas about the system between peer reviewers. Advantage of Peer Review is that various review ideas can be revealed, checked freely, and can be shared by peers. However, if not a capable leader is appointed, review process cannot get desired results, or can be jeopardized by someone who has strong opinion [2].

2.1.3 Walkthrough

Fewster and Graham (1999) define Walkthrough as "A step-by-step presentation of documents by author to fellow developers in order to gather information, to establish common understanding, and to find defects of its content. Walkthrough is a review conference in order to find defects by other fellow developers [7]." Generally, walkthrough is performed on the program source codes. Reviewers read the source code line by line, and report the found issues. Walkthrough also includes review of data definition, manual, and other specifications other than source code.

2.2 Inspection

In ISO 17000/17020, Inspection is defined as "Examination of a product design, product, service, process or plant/installation, and determination of their conformity with specific requirements or, on the basis of professional judgement, general requirements" [4]. Inspection is one of the most popular review practice found in software project developments. The goal of Inspection is to reach consensus for the inspectors that the developed system is good enough to be used in real world. Commonly, inspection references the software requirements from order company, specifications from software company and test plans from third parties. To proceed inspection, interim work product(s) is selected, and team members gathered for inspection review [16].

Inspection was proposed by Michael Fagan in 1976 to enhance the quality and productivity of software development [6]. Fagan defined that "Inspection is the method that several people examine software related document to find violations of development standards, non-conformance to higher level of documentation etc. Inspection is the most formal review technique that it is generally processed based on the documented procedure, and the reviewed result is also documented in predetermined manner. So the effectiveness of Inspection is proven by many researches in many real applications [12].

2.3 Comparison between Inspection and existing methods

Inspection, Peer Review and Walkthrough share common goal of detecting problems and enhancing the quality of software. However, the purposes of each method are slightly different. IBM proposed very good remarks on this subject. In 1992, IBM announced that Walkthrough is for team training, and Review is for getting consensus of technical issues and selected method. And Inspection is used to check work output quality, and also used as a quality enhancement tool for the process that makes the work output [9]. However, Inspection and other methods have two major differences [13].

First, in Inspection statistical data for defects are more well kept and tracked for quality enhancement. In other words, data gathering and analysis are essential part of Inspection process.

Second, Inspection procedure is more standardized than other methods, which enables early detection of defects is possible.

In Inspection method, possible defects and errors in each system development stage are predefined, documented, and required to follow the standardized Inspection procedure. So there can be less possibility that errors of former procedures inherit to the next procedures, which will substantially reduce the total number of errors in whole system.

To summarize the above arguments, Inspection is performed systematically based on the predefined rules and procedures throughout the entire system development. Also walkthrough and other test methods may be used simultaneously to enhance the quality of system development.

<Table 1> Feature comparison of review methods

	Features
Peer-Review	- attendance of peer and technical expert - can be formal or informal in actual affairs - (progress favorably) obtain consistent and quantitative effect regardless of reviewer - the main object : technical problem solution, discussion, decision making, alternation appraisement, defect detection, compatibility review of specification or standard
Walkthrough	- progress or control by writer - use scenario, dry run, review of peer group - open-ended session regardless of time and attendant number - can be formal or informal in actual affairs - the main object : learning, enhance understanding to system, defect detection
Inspection	- progress or control by moderator - be fixed role - collect and use a matrix - being a formal process based on check list with start and end condition - need prepare stage before meeting - making inspection report and incident list - being a formal follow-up process - the main object : defect detection

III. DEVELOPMENT OF INSPECTION RULE SET FOR EFFECTIVE SOFTWARE DEVELOPMENT

In this Chapter, new Code Inspection rules are introduced, and company "A" information system development project is reviewed to check the effectiveness of the developed Inspection rules.

<Table 2> Review items for Code Inspection

Category	NO	Index Item	Specifications
Readability & Maintenance (8)	1	Macro Naming	Macro name uses capital letter or underbar(_).
	2	Function Naming	The first letter of name uses small letter, then the first letter of the following word uses chapital To seperate words use underbar, nameing function to make hierarical sturcture
	3	Enum Constants Naming	Enun Constants describes by defined rule(all capital letter)
	4	Global Variable Naming	The first letter of global variable uses small letter, and the following letters include number, underbar and alphabet.
	5	Local Variable Naming	The fisrt letter uses small letter, and the following use numbers, underbar, alphabet.
	6	File Naming	The first letter of file name use alphabet, extention(C, c, H, h) can use capital or small letter
	7	#define or #undef within a block	Not allow #define or #undef in block
	8	File Comments	The first line should use /*(header comment)
Dead Code (2)	9	Failure Definition Local Variables	Remove the unused local variable(include parameter)
	10	Non-Null statements	Remove the meaningless statement
Potential Error (8)	11	Default in Switch	The last sentence of ‘ Switch Statement’ uses default
	12	Floating Point Comparison	Not allow comparision operation of floating point(float or double)
	13	Uninitialized Pointer	Initialize all point
	14	Variable Initialization	Initialize all variable
	15	Null pointer Assignment	All substitute sentence has valid value
	16	Assignments in boolean expression	Exclusive subtitute operation in conditional sentence
	17	Braces of loop body	Use Braces({}) in For sentence
	18	Three expressions of a for statement	Three equation of For sentence relates only loop control
Control Error(4)	19	Unreachable Code	Remove the inaccessible code
	20	Goto Statement	Not allow unstructured GOTO sentence
	21	Empty Block body	Not allow empty block setences(body of if, for, while and do)
	22	Loop Counter type	Loop Counter uses signed value
Performance(1)	23	Debug Statement	Not allow console(printf()) sentence
Storage Management(1)	24	Dynamic heap Memory	Minimize ‘ Dynamic Heap Memory’ allocation(function calloc, malloc, realloc, free)
Interface(5)	25	Number of Arguments and Parameters	Number of arguments same with number of parameters
	26	External Definition object	Not duplicate object(global variable) which is not static
	27	External Definition function	Not duplicate function name which is not static
	28	Internal linkage of object	Static is used to define object or function with internal connection, otherwise should use ‘ extern’
	29	Internal linkage of function	
Security(1)	30	Observe Prohibition Function	Reference prohibition function for detail items(reference program writing guide from order company)

3.1 Rule establishment for Code Inspection

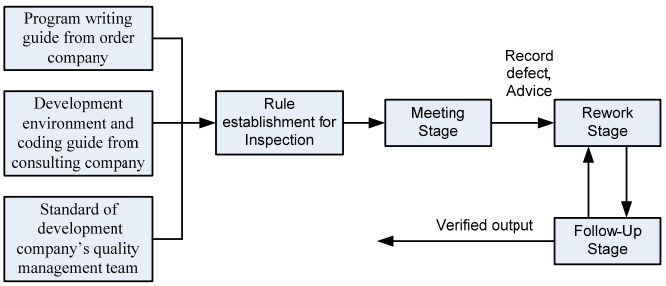
Inspection procedure is relatively structured than other review methods, still there are many rooms to explore, and lacks consensus in implementing method between professionals. In this research, new set of rules are developed for better implementation of Inspection procedures. Guidelines from three related organizations and experiences of high level system analysts are used to develop the rules.

- Program coding guide from Order Company.
- Development environment and coding guide from consulting company.
- Standards of development company's quality assurance team.

The new rule set has 8 categories and 30 indexes grouped by similarity of rule characteristic. <Table 2> shows the Code Inspection rules in detail.

3.2 Code Inspection procedure

<Fig. 1> shows the detailed steps of suggested method. Suggested Inspection procedure has four steps: First, rules for the inspection is set up based on the guide lines from three related organizations. Second, the inspection is performed by the established rules. Third, all defects found are reworked. Fourth, move to next step unless other defects are found.



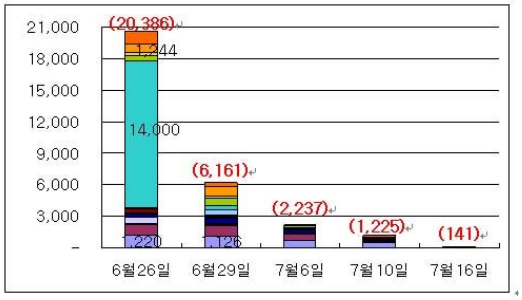
<Fig. 1> Code Inspection Stage

IV. Application of Code Inspection Rule to Real System

To check the rationality of developed rule set, rules are applied to the real information system. To avoid biases, Code Inspection team was consisted of three different groups, developers, QA members and project manager. Code Inspection was done by the rule set described in <Table 2>.

4.1 Analysis of Code Inspection result

Actual code Inspection was done on the 6 major procedures of company "A" and one common procedure ruling over the whole company. Code Inspection was performed 5 times, and spanned 3 weeks to follow the system development stage of company. <Fig. 2> shows the numbers of defects detected in each round of Code Inspection.

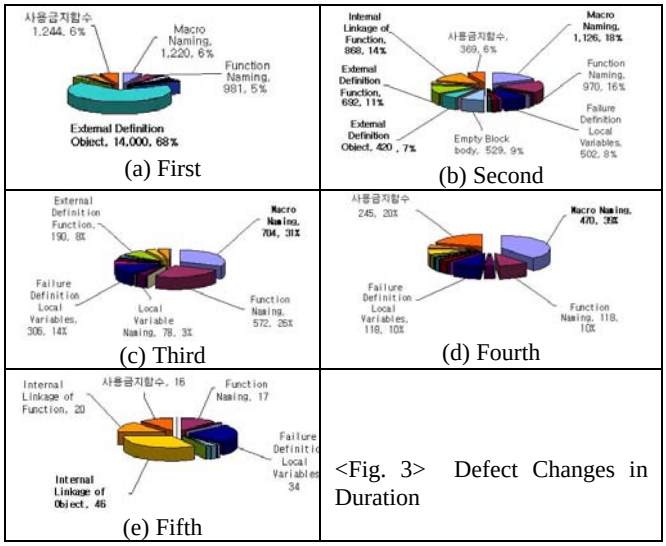


<Fig. 2> Defection Changes in Duration

As shown <Fig. 2>, 20,386 defects found in first Code Inspection, but the numbers are dramatically decreased as Inspection stage processed.

<Fig. 3> shows the numbers and types of defects revealed in each Inspection stage.

Fig.3 Numbers and types of defects in each Inspection



<Fig. 3> Defect Changes in Duration

<Fig. 3> (a) shows that total of 20,386 defects are found, and in <Figure 3> (b), second Code Inspection, 30% of defects are reduced. Through the third and Fourth Code Inspection, the number of defects decreases dramatically. In fifth Code Inspection, only 141 defects are detected. Also the types of defects in prior stage did not occur or reduced in number substantially in next stage. It clearly means that mistakes of prior stage do not inherit to the next stage. Considering the over four hundred thousand lines of total system code, number of defects and decreasing speed of defects in each stage are remarkable.

V. Conclusions and Future Research

Inspection is one of the well known methods to prevent or reduce the number of defects in programming. However, actual implementation is not performed systematically, so more refined way of Inspection procedure needs to be developed. In this research, Inspection and other existing test methods are compared, and more elaborated Code Inspection method is

suggested. Also their rationality is checked by applying the rules to real case.

The results founded in this research are as follows.

First; Clearly defined the differences between Inspection and other review methods

Inspection method is a popular way to enhance the quality of software. However, definition of Inspection and performing method are not clearly defined. And many of system developers tend to confuse Inspection method with other Peer Review, Walkthrough, and Test. In this research, differences of Inspection and other existing methods are compared and presented.

Second; Develop the more effective Code Inspection rules.

Based on the experiences of high level system developers and guidelines of consulting Companies, new set of Inspection rules, 8 categories and 30 indexes, are defined for better implementation of Inspection.

Third; Rationality and Effectiveness of new rule set are tested throughout the real system development period.

New set of rules were applied to the each stage of system development, and carefully analyzed the numbers and types of defects. Code Inspection was done on each output, functions and interfaces of system. We found very steep decrease in defect numbers and types of errors. New rule set is anticipated to raise the quality of software, and may lessen the burden of maintenance. If we can decrease the time to fix errors, we may spend more time with users, and use more efforts to upgrade the system. Saving time in fixing errors and maintenance not simply means the saving the time. Rather, it means increase of system quality, more user oriented system development and saving costs.

It may be difficult to say that effectiveness of suggested rule set is verified only through one system application. And comparative increase in effectiveness was not quantified in this research. This is limitation of this work. Suggested rule set also needs to be modified or be extended for general purposes. However, the 8 categories and 30 indexes developed will provide good stance to the future research works.

REFERENCES

[1] Choi, E. M., *Software Engineering*, Jungiksa, 2006.
[2] CMU/SEI, The Capability Maturity Model: Guides for Improving the Software Process, Addison Wesley, 1994.
[3] Davis, A., *Software Requirements: Analysis and Specification*, Prentice-Hall, 1990, p. 20.
[4] Eberhardt, H., Examples of Inspection requirements and ISO/IEC 17020, ISO REGIONAL WORKSHOP on CONFORMITY ASSESSMENT, 2007
[5] Fagan, M. E., "Design and Code Inspections to Reduce Errors in Programming Development", IBM Systems, Vol. 15, No. 3, 1976.
[6] Fagan, M. E., "Advances in Software Inspections, IEEE Transactions in Software Engineering", Vol. 12, No. 7, 1986, pp. 744~751
[7] Fewster, M. and D. Graham, *Software Test Automation, Effective Use of Test Execution Tools*, Addison-Wesley, 1999.
[8] Freedman, Daniel P. , Gerald M. Weinberg, *Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products*, Dorset House Publishing Company, 1990.
[9] Gilb, T., Dorothy Graham, Susannah Finzi, *Software Inspection*, Addison-Wesley, 1993.
[10] IEEE, *IEEE Standard Glossary of Software Engineering Terms*, IEEE Society Press, 1983.
[11] Jones, C., *Applied Software Measurement*, McGraw-Hill, 1996.
[12] Kelly, J. C., J. S. Sherif and J. Hops, "An Analysis of Defect Densities Found During. Software Inspections", Journal of Systems Software, Vol. 17, 1992.
[13] Lee, S. Y., *S/W Inspection Effectiveness Analysis and Improvement Methode*, Sogang Univ., 2002.
[14] McGarry, F., Gerald Page, Victor Basili, *Software Process Improvement in the NASA Software Engineering Laboratory*, CMU/SEI-94-TR-22, 1994.
[15] Russel, G. W., "Experience with Inspection in Ultralarge-Scale Developments", IEEE Software, Vol. 8, No. 1, 1991, pp. 25~31
[16] Stellman, Andrew & Jennifer Greene, *Applied Software Project Management*, Oreilly & Associates Inc, 2005.
[17] Wolfhart Goethert, Matthew Fisher, *Measuring Acquisition Process*, SEPG 2002, 2002.

Taewon Kyung received the M.S. degree in computer engineering from the Kyunghee University, Korea in 2002 and he is currently working toward the Ph.D. degree in industrial engineering from Kyunghee University. He is a Project Management Professional and Professional Engineer Broadcasting Information. His current research interests include IT project management, IT plan and strategy, and MIS.

Sangkuk Kim received the B.A. degree from Seoul National University, Seoul, Korea and the Ph.D. degree from University of Wisconsin, Madison, Wisconsin, USA, 1989. Since 1989, he has been with the Kyunghee University, Korea, where he is a Professor of Industrial Engineering. He was the president of Korea Intelligent Information Systems Society, and the audit of The Korea Society of Management Information Systems. His current research interests include BPR(Business Process Reengineering), management strategy, and MIS.