Comparing Inspection Strategies for Software Requirement Specifications

Benjamin Cheng and Ross Jeffery
The University of New South Wales, Australia

Abstract

This paper reports on a laboratory experiment into the use of decomposition strategies in software requirements inspections. The experiment follows on from the work of Porter, Votta, and Basili who compared the use of scenarios with ad hoc and checklist techniques, finding (among other things) that the scenario technique was superior. This experiment compares the scenario technique with inspection strategies which are self set by the inspection team prior to the inspection but after they have seen the documents to be inspected. The specification used was a system developed by a software company for a client in the commercial sector. It was found that the commercial scenarios developed for the experiment were not superior to self set strategies. This suggests that the benefits to be derived from scenarios are derived through the decomposition process and that experienced people may be able to derive strategies that are at least as good, if not better, than a provided set of scenarios. An advantage we noticed with the provided scenarios was the manner in which this technique could be used to focus the reviewers' attention on particular defect types. This could be used to advantage in industry. The overall findings of this experiment supports and extends the earlier research on inspections.

1. Introduction

"There is still a great deal of room to make mistakes - which, of course, is why the software field remains so challenging." (B. Boehm)

The technique of formal inspections, begun by Michael Fagan of IBM in 1972, has been practiced in industry for over twenty years now. While initially designed for use at the code level, there has been an extension of practice to cover earlier stages of the system development life-cycle (SDLC) [1,5,6,8,9]. Kelly et al. [14] report that there are more defects found in software requirement specifications (SRS) than any other documents produced during the SDLC. Despite the general recognition of the need to carry out inspections earlier, SRS inspections are not commonly practiced in the industry [18,23]. This article addresses the methods or process used in the inspection of an SRS. This study has broadened our understanding of the use of scenarios as a method for requirements inspections. In view of the need for an inspection strategy for commercial systems, we have developed a set of scenarios (Function Point Scenarios - FPS) based on Function Point Analysis. We conducted an experiment and found the technique had enhanced the skill level of the reviewers and had also directed them to investigate more crucial areas of the SRS.

This article presents the background, initiatives, and the results of the experiment. Discussion of the FPS is also included.

2. Background

2.1 SRS & Inspection

A software requirement specification (SRS) is a "...specification for a particular software product, program, or set of programs that does certain things." [11] An inspection is "... a formal evaluation technique in which software requirements, design, or code are examined in detail by a person or group other than the author to detect faults, violations of development standards, and other problems..." [8] (ANSI/IEEE std 729-1983). The objective of an inspection is therefore to:

- verify that the software element(s) satisfy its specifications,
- verify that the software element(s) conform to applicable standards,
- identify deviation from standards and specifications, and
- collect software engineering data (such as defect and effort data).

0-8186-7635-3/96 $05.00 © 1996 IEEE

2.2 Defect detection

Major areas of defects include specification or requirement defects, design defects, code defects, user document defect, and environmental support defect [5,9]. Current techniques for SRS defect identification have been focused on two major areas:

- error avoidance: use of formal specification languages
- error detection: formal inspection, walkthrough, review

Our research attention is on the latter. Many researchers have prescribed techniques for defect detection activities. These include Reviews [7], Walkthroughs [24], and Inspection [5,6,8,10]. An extensive summary of the features and characteristics of these is presented in Kim et al. [17].

2.3 Inspection techniques

Most inspection techniques have adopted a non-systematic way of identifying defects such as the *ad-hoc* and *checklist* approaches [8,6,14,20]. No direction is provided in the *ad-hoc* approach and the inspectors are required to utilise their own experience and knowledge to identify problem areas in the document. For the *checklist* approach, the inspectors are provided with a list of general defect classes to check against. For both *ad-hoc* and *checklist* approaches, coordination of reviewers' responsibilities is relatively minimal.

Recent research reveals inefficiencies in the *ad-hoc* and *checklist* approaches. Some of the major problems of these conventional reviews are identified as [19,20,23]:

1. The reviewers are overloaded with excessive information.
2. Ignorance of reviewers as to the assumptions and details of the design.
3. Ambiguous responsibilities.
4. No clear path to start (because of 1,2,3)
5. Limited interaction between reviewers and design team.
6. Participation of unskilled reviewers
7. No systematic review procedure and no prepared set of questions to be asked about the design.

Based on the suggestion of Parnas and Weiss on the separation of the reviewer's duties, Porter, Votta, and Basili [21] (hereafter referred to as PVB) pioneered a *Scenario* approach, claimed to provide a more systematic environment for defect detection. The 'scenarios' are a set of procedures for identifying specific classes of defects. Each reviewer in the team follows a different scenario (in the *ad-hoc* and *checklist* methods all reviewers follow the same rules). Three detection methods used by PVB were:

- Ad hoc: non-systematic technique which all reviewers performed the same tasks utilising their own experience.
- Checklist: similar to ad-hoc but a defect checklist was provided.
- Scenario: partitioned and coordinated responsibilities

The result of an experiment comparing *Ad hoc*, *Checklist*, and *Scenarios* indicated that the *Scenario* approach was superior. Compared with the *Ad hoc* and *Checklist*, the defect detection rate was improved by about 35% when using *Scenarios*.

3. New approach

3.1 Inspection strategy for commercial systems

As the need for an effective inspection technique arises, we attempt to extend the application of Scenarios to a broader range of applications. To prove the effectiveness of the tool, we also planned for an inspection experiment to be conducted with a set of Scenario tools. We have adopted a different approach to PVB due to the following reasons.

First, the application domain addressed by PVB were engineering-based embedded systems requiring substantial mathematical computation. Requirements of such nature are typically better formulated and are well-bounded in comparison with commercial systems. Our interest was to investigate the application of Scenarios in the commercial application systems domain. System requirements in this domain are often not as clear cut at the design and specification level than the engineering counterpart. Also the nature of commercial applications focuses more on data update and retrieval, file manipulation, and user enquires. Mathematical computation is basic. The tools developed by PVB are not adequate for application of such nature.

Several difficulties became apparent in planning for our experiment. After careful investigation of PVB's Scenarios, we were not able to determine the level of orthogonality provided in the Scenarios. PVB claimed that their Scenarios were decomposed from the Checklist they produced. We had made substantial attempts to correlate the coverage in the Checklist to the Scenarios but this was not successful due to the ambiguity in the items addressed by the Scenarios. There was no indication about the completeness of the Scenarios in covering most of the defect classes. In addition, we were not able to uncover literature support for the way the

Scenarios were composed. This suggested to us that more research needs to be done particularly with respect to Scenario partitioning.

We also realised that the issues identified by Parnas and Weiss [19], as well as in PVB's hypothesis, was that coordinated responsibilities were superior to uncoordinated. However, we could find no indication in the research that the Scenario approach used in the experiments is superior to any other means of coordination since the experiments compared scenarios with ad hoc and checklist.. The issue of comparative coordination methods was not investigated in PVB's experiments.

We hypothesise other means of decomposition may perform as equally well as Scenarios. Kelly et al. [14] found that an increased number of pages to be inspected caused a drop in the number of defects found by the reviewers. Therefore, there was some support that the increase in effectiveness of the Scenario approach could possibly be the consequence of a reduction in work load. Our interest was in this question.

3.2 Function Point Scenarios (FPS)

In view of the need for an inspection tool for commercial transaction processing systems (TPS), we developed a set of decomposition scenarios based on the Function Point Analysis (FPA). The technique is applicable to most commercial systems and has extensive support in the literature as a method of describing commercial systems requirements [2,3].

Operational functions of conventional TPS consist of mainly: inputs, files, enquiries, and outputs. These categories represents an orthogonal functional dissection of typical transaction processing systems. FPA has adopted this perspective and incorporated the following distinctions:

1. Internal Logical File (ILF): A group of data that is maintained within the application boundary (eg. master files, audit files).
2. External Interface File (EIF): A group of data that is referenced by and external (not maintained by) to the application (eg. reference data).
3. External Input (EI): A process that receives data or control information that comes from outside the application's boundary (eg. screen input, batch input).
4. External Output (EO): A process that generates data or control information sent outside the application's boundary. (eg. reports, data transfer to other processes).
5. External Inquiry (EQ): A process that is made up of an input-output combination that results in data retrieval (eg. data retrieval, help facilities).

More details of these functions can be found in [12].

FPA has been proven to be an effective measuring technique for commercial system size which counts the number of functions in each of the categories that are to be implemented [13,15,16]. The Function Point Scenarios (FPS) are based on the functional dissection of typical commercial systems. This particular technology has provided us with a basis for the scenario construction in two important aspects:

- the scenarios describe the whole of the system to be inspected; and
- the scenarios provide a method of decomposing the system into constituent parts.

The materials for constructing the Function Point Scenarios are based upon the original Function Point Counting Practices Manual version 4 and the Checklist items in PVB [21].

A modification is made to combine ILF and ELF to form a 'File' section for their characteristics are similar in regard to our inspection requirements. In addition, an overview section is added to inspect the overall relevance, completeness, and correctness of the SRS. Our final FPS therefore consists of the following sections:

1. Overview: Overall description of the entire system.
2. File: A file that contains attributes for the purpose of the application.
3. Input: A process that requests information from a user or another system.
4. Output: A process that generates information for a user or another system.
5. Inquiry: A process that retrieves data in response to an input from a user or a process and outputs the data to a user or another system. The process does not involve update of data.

Key issues covered by each FPA area are first constructed into domain-related questions. The questions are supplemented by relevant items in PVB's Checklist. Each of the FPS areas are then constructed into FPS Scenarios that investigate different aspects of the SRS. A checklist item may be repeatedly introduced in various FPS items if it helps the reviewer to identify the defect in that respect. For example, the item "*Is the processing logic correct?*" is repeated in 'Input', 'Output', and 'Inquiry' sections of the FPS to address the processing logic of each respect.

When using FPS in inspection, each of the reviewers are directed to focus on specific areas of the SRS. For example, in the File Scenario, it requests the reviewer to identify all occurrences of each attribute in the document

(for example, ER diagram, Data Flow Diagram, data definitions in Data Dictionary).

4. Experimentation

We have conducted an experiment to examine the usefulness of FPS. The quality of our experiment is improved in the following ways:

- The FPA decomposition is well supported by research.
- The FPS address the inspection criteria for commercial systems.
- There is an opportunity to investigate the effectiveness of other means of decomposition strategies for inspection.

4.1 Hypothesis

We have generalised the problem domain and established the missions of this research as:

1. Do reviewers who use Function Point Scenarios perform better than those who do not?
2. Will a provided decomposition strategy (such as FPS) result in more faults being detected than a self set strategy for team-based inspection of a commercial SRS?

The first question addresses the effectiveness of FPS on commercial systems. The second question attempts a broader perspective by investigating the fundamental implication behind the Scenarios (either FPS or the PVB's). It is to determine the question of whether the improvement (if any) in inspection performance is attributable to the provision of a written scenario or whether the improvement in performance is attributable to the act of decomposing the system into elements and then attacking the task with individuals working in parallel.

Rather than examining the total number of defect found in a lump-sum, it is necessary to distinct between true defects and false positive. The former contributes to the effectiveness of inspection but the latter constitutes a waste of resource. Our null hypothesis is therefore as follows:

H_0: There is no difference in the average number of true defects found (m) between the FPS reviewers and the reviewers who adopt self-set strategies (SSS).

$$m_{(FPS)} = m_{(SSS)}$$

Later in this article, we will illustrate a defect categorisation scheme we derived which provides a finer understanding of the reviewers' behaviour in defect identification.

4.2 Experimental Method

Subjects

Having constructed the FPS, which provided both inspection method and system decomposition, we developed an experiment which was conducted in late 1994. There were 53 subjects, 31 of these in a graduate Masters program in Information Systems. Entry to this program requires industry experience of at least one year. The remaining 12 subjects were in final year of a 4-year undergraduate degree which also mandates industrial experience for at least 18 months with experience in both design and programming.

For this reason we expected little difference between the two experimental subject groups, the vast majority of whom had significant industrial experience.

Team formation

The experiment was run in three sessions of three hours duration each. Each team had 3 members (except one team which had 2). Two techniques were used:

- Self-Set-Strategy (SSS): Reviewers are required to developed their own inspection strategies prior to start of individual and group inspections.
- Function Point Scenario (FPS): Reviewers are provided with a set of FPS scripts. Allocation of the scripts is arbitrary with a team.

The masters students completed the experiment in two separate groups (Sessions B and C) while the undergraduates were assigned to Session A. It would have been preferable to randomly allocate treatments across all 53 subjects, but that was not possible because of teaching schedule requirements. (See below for effects on internal validity). However, assignment of subjects to inspection teams was random within each of the three sessions.

Apparatus

The following material were used in the experiment[1]:

1. Pre-experimental survey form: Distributed and collected one week before the experiments.

[1] A full set of the apparatus is available from the authors of this paper.

Questions about the reviewer's background are asked.

2. Instructional material: It consists of lecture notes and a follow-up articles [18] on inspection.
3. SRS: A design specification document selected from a real industrial case.
4. Instructions: A simple defect taxonomy list for SSS teams and a set of FPS scripts for the FPS teams.
5. Defect report forms (individual & group): One for use during individual reading time and the other for group discussion time. Both the SSS and FPS groups have slightly different forms.
6. Post-experimental survey form s (individual and group): Questions about the reviewer's attitude towards the inspection session and further comments that they want to make.

In this experiment only one SRS was used by all subjects. The SRS describes a commercial system which was developed and implemented in 1992 by a software contracting corporation for a company in Australia. We seeded the document with faults in addition to any which were pre-existing.[2]

Procedures

The experimental procedures can be summarised as:

1. *Reviewer Background Survey.* All subjects completed a pre-experimental questionnaire which provided us with details of education, industry experience, inspection experience, and so cn.
2. *Instruction and Training.* Forty minutes duration in which subjects were given details of the experiment, the method to be used (either FPS or SSS), the general functionality of the system to be inspected, group member assignment, and the inspection aims and process. Subjects were provided with instructional material.
3. *Individual Preparation.* This was for a period of one hour in which groups either selected their strategy or were given detailed instruction in FPS, followed by the individual preparation and recording of discovered defects using the provided SRS.
4. *Group Inspection Meeting.* Again a period of one hour in which groups carried out defect aggregation, discovery and recording accompanied by group discussion.
5. *Post Experimental Survey.* A period of twenty minutes was used to discuss inspection issues and to complete the survey which provided details of experimental confidence and value on the part of the subjects.

All subjects were given instruction in inspections and, where relevant, in FPS. They were instructed to inspect the SRS individually (inspection preparation) and then an interacting group inspection was performed. If teams in a session were to use SSS they were instructed to meet as a team before inspection preparation began and to develop a decomposition strategy to use in inspection preparation.

4.3 Defect types

Instead of just counting the number of defects found by the reviewers, we examined the type of defects found. We constructed the defect classification scheme shown in Table 1.

Defect variables	*Defect type*	*Description*
	Individual defect types	
F	False positive	Report of a defect which is not a defect.
P	Overly sensitive	Report of a defect arising because of differences in personal preference to the style of wording and presentation used in the SRS.
R	Repeated finding	Report of a defect which had previously been identified by the same reviewer or team.
U	Unclear description	Description of a defect on the report form that cannot be understood by the authors.
D	Design related	Suggestion of ways the system design can sensibly be enhanced.
G	Syntactical	Typographical or other syntactical mistakes.
T	Consequential	Defect which could lead to future system faults if it is not corrected in the SRS.
	Major defect classes	
FPRU	False positive class	Defects which were not correctly being identified. Comprises of the elements F, P, R and U.
DGT	True defect class	Defects which were correctly identified. Comprises of the elements D, G and T.

Table 1 Defect classification scheme

[2] The SRS we used therefore is in no way representative of the quality of the document or any other documents or products produced by the organisation.

The focus of this paper is on the major defect classes (FPRU, DGT) when comparing between the two inspection methods (SSS and FPS). An issue arise on where a 'P' defect (overly-sensitive) should be grouped. We perceive this controversy and decided to carry it with the false positive for there is a chance that a reviewer would favour the style and hence regard the item as correct.

All our data are ratio data. However due to the non-normality of the data (tested using the Kolmogorov-Smirnov test) we decided to use the non-parametric Mann-Wittney U test[3].

Our null hypothesis is modified to accommodate the various defect classes that there is no difference between the two approaches:

H_0: $U_{SSS(FPRU)} = U_{FPS(FPRU)}$,
$U_{SSS(DGT)} = U_{FPS(DGT)}$

5. Results

Before examining issues about inspection strategy, we need to check if the two groups of SSS reviewers (Session A and C) can be combined as mentioned above.

Grouped t-test results reveals that, apart from being slightly over-sensitive, the result indicates that the SSS1 (Session A) reviewers do not differ in performance from the SSS2 (Session B) reviewers. The effect of P diminishes as it is merged with the False Positive class (FPRU). We conclude that the influence of P on differentiating the two groups of SSS reviewers is negligible. Our analysis therefore combines the two groups and focuses on the inspection strategy (SSS v's FPS).

All inspection data collected at both the Individual Preparation stage and the Group Inspection Meeting stage were classified according to the scheme in Table 1. This was done solely by the authors to ensure consistency. The scope of this paper is to investigate only the performance of the team as whole and therefore only the group data is presented. A comparison of individual versus group performance will be reported elsewhere.

For the group data, we have recorded the number of defects reported by individual reviewers in the group meeting and those which were found during the Group Inspection Meeting. There were 9 teams for each inspection strategy. The variables are abbreviated in the following way:

I_*: Defects identified by individual reviewers and reported in the group documents.

G_*: New defects identified by the group during the group inspection meeting.

ALL_*: All defected identified by the group (I_* + G_*).

Figure 1 shows the distribution of reported defects during the Group Inspection Meeting. The difference of the variables (vertical line) being the extra defects that were found by the team at meeting time. For a large number of groups, additional defects were identified at the group meeting.

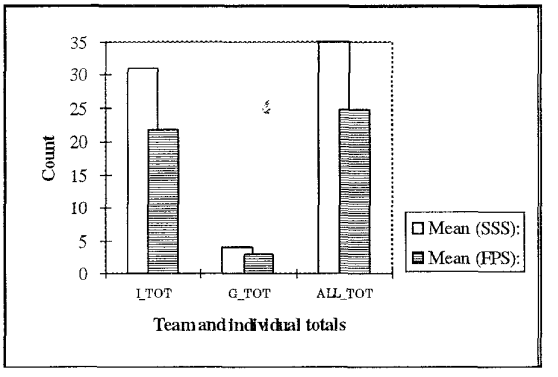

Figure 1 Average number of defects reported.

The averages across the methods are shown in Figure 2.

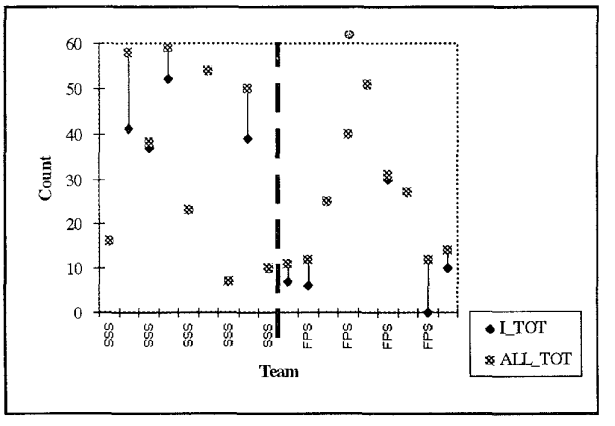

Figure 2 Defects reported by all the teams.

These result indicate that overall the SSS group (Mean of 35.00) had identified more defects than the FPS (Scenario) group (Mean of 24.78). In addition, it also shows that group meeting is contributing in identifying

[3] We have also analysed the data using group t-tests and found the results to be identical.

additional defects. The average gain for SSS is 11.4% (4/35) and 12.5% (3/24) for the FPS group.

The averages of the classified defect types are illustrated in Figure 3.

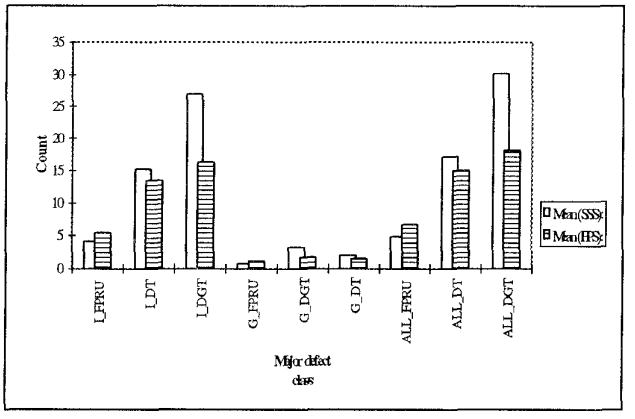

Figure 3 The averages of the classified defect types.

The result indicates a trend that the SSS group (both individual reviewers and teams) are superior to the FPS group (reviewers and teams) in identifying:

- more true defects (D,G,T),
- less false positives (F,P,R,U)

The difference is not statistically significant, however. The Mann-Wittney U test was performed on the defect type variables[4]. The variables that indicate significant difference[5] are shown in Table 2.

Defect	SSS (mean rank)	FPS (mean rank)	U	2-tailed P
I_P	12.33	6.67	15	.0244
I_G	12.72	6.28	11.5	.0078
ALL_F	6.33	12.67	12	.0106
ALL_P	12.44	6.56	14	.0188
ALL_G	12.78	6.22	11	.0078

Table 2 Results showing variables that are significant different.

The SSS reviewers, individually, appeared to be overly sensitive (p(I_P) less than 0.05) and picked up more syntactical defects (p((I_G) less than 0.05) than the FPS group. This also accounts for the differences within the overall variables (ALL_P, ALL_G). The FPS teams, on the other hand, found more false positives than the SSS teams (p(ALL_F) is less than 0.05).

Nonetheless, there is no significant difference between two groups with respect to the major defect classes. The null hypothesis is therefore accepted.

6. Discussion

Our experiment revealed no significant difference between the SSS and FPS groups in their inspection performance, either in identified true defects or overall false positives. Comparison on major defect categories showed no significant difference between the two treatments. The FPS decomposition Scenario strategy does not appear to be superior to the use of self-set strategies. This result adds to the findings of PVB, suggesting that the critical element may not be use of scenarios but rather the use of a decomposition strategy. Furthermore, it suggests that experienced individuals may be able to set their own strategies so as to achieve an overall performance level for the group at least as good as that achieved by the use of a scenario.

We examined the descriptions provided by the subjects in the post-experimental survey forms and found that various decomposition strategies were used. Examples of such were the partitioning the document into sections, logical processes, and areas of interest. Each reviewer was assigned an allotment. Given that the Scenario strategy is another mean of decomposition, it is not surprising that it behaved in a similar manner. This experiment has added some additional evidence to the proposition that decomposition strategies, in general, enhance inspection performance.

Although not statistically significant in this experiment, the SSS reviewers, on average, found more defects than the FPS group. Their performance was better in being able to identify more true defects and also less false positives, which is an observation in need of further study.

Further analysis has been carried out on each reviewer's performance relative to their background[6]. Results here indicate that reviewers with greater computing or information systems experience are able to identify more design related defects (D) when using SSS (p = 0.009). However no difference is distinguished for the FPS group and it is possible that this has occurred either because the scenario has the effect of improving the performance of

[4] We have also compiled the data using group t-test and found the results to be identical. However, we decided to use the non-parametric Mann-Whitney test because of the small sample size and non-normality of our data.

[5] 5% significance level is used throughout the experiment.

[6] The analysis is compiled in addition with the data obtained in the Reviewer's Background Survey.

the less experienced reviewers or reducing the effectiveness of the more experienced reviewers.

Syntactical defects (G), on average, account for 43% ($I\_G_{(SSS)}$ / $I\_DGT_{(SSS)}$ = 11.78/26.89) of the true defects (DGT) for the SSS group while it is only 17.7% ($I\_G_{(FPS)}$ / $I\_DGT_{(FPS)}$ = 2.89/16.33) for the FPS group. Similarly, the SSS reviewers were more overly-sensitive as they are accountable for generating more P type defects. This also may point to subtle influences in the use of a provided decomposition strategy over the SSS. These differences. however, may be important in the industrial setting.

When the reviewers were asked to rate their confidence in their inspection achievement[7], it was found that SSS reviewers who had least confidence had identified significantly more syntactical defects than the FPS reviewers. This could be an effect of undirected work for the SSS group. The issue of syntactical defects is not explicitly addressed in our FPS Scenarios and hence it is not surprising that fewer were identified. This result agrees with PVB's finding that Scenarios (or coordinated responsibilities) help reviewers to focus on specific defect class.

Excessive concentration on trivial defects is time consuming and not cost effective. The benefit of the neutralisation effect of Scenarios is that reviewers are directed to concentrate on more consequential issues. We suggest further research be conducted in this regard to investigate ways of improving support to the reviewers.

PVB reported that the group meeting had no net benefit when the gain and loss were combined in their experiments. Our result indicates the group meeting is effective in identifying more true defects. The net meeting gain (G_DGT/I_DGT) is found to be 12% (3.22/26.9) for the SSS group and 11% (1.7778/16.3) for the FPS group. Net meeting loss (G_FPRU/I_DGT) is 2.9% (0.7778)/26.9) for SSS group and 7.5% (1.222/16.3) for the FPS group. Thus there was a net benefit of 9.1% and 3.5% for the SSS and FPS groups respectively. This calculation is simplistic with no consideration of defect severity or identification value in the combination. There are many other ways in which these could be combined.

Given the gain in performance, these results suggest that group meetings should be conducted. The data however indicates a lower meeting gain for the FPS Scenario group. This may indicate that group meetings may not be as effective in an environment in which the reviewer's responsibilities are highly divergent.

Although there were only 18 groups in this experiment, we believe the result to be representative. However, we are currently planning for replication of our experiment to be performed to confirm the results.

[7] The rating collected in the post-experiment questionnaire, is a Likert scale of 1 to 7 from least confident to most confident.

7. Conclusion

Research on inspection has focused largely on code. However, recent literature and experience has suggested inspections should be carried out in earlier stages of the SDLC.

Our experiment has provided consensus with research and industrial experience that inspections are effective in discovering defects in an SRS before coding and implementation. In addition we have investigated the effectiveness of different inspection strategies and supported the prior research findings that 'decomposition' in general is an appropriate approach. Scenarios, representing one type of decomposition, can be as effective as some other well-planned partitioning scheme. However, it appears that experienced personnel may be able to quickly decide on a decomposition strategy that is as effective as a provided detailed decomposition strategy. The provided strategy, however, has benefits in directing the reviewers to particular defect types (or discouraged them from concentrating on trivial defects), thus providing a mechanism to focus the review according to organisational objectives.

Qualitative issues in inspections were also addressed in the experiment. Trivial defects, such as syntactical mistakes in the SRS, received probably too much attention by many of the reviewers. As inspections are relatively expensive, the results suggest that further research should look at defect prioritisation as well as the support that has to be provided to the reviewers in order to match the defect process to the defect type priorities of the organisation. The provision of a guided decomposition strategy can be beneficial in directing attention of the reviewers to more critical defects.

Since requirement specifications are more semantically oriented than code, additional research is needed to provide better guidelines and the required process granularity for such an environment.

References

[1] A. F. Ackerman, L. S. Buchwalk, and F. H. Lewsld, Software Inspections: An Effective Verification Process, *IEEE Software*, May, 1989.

[2] A. J. Albercht and J. E. Gaffney, Software Function, Source Line of Code, and Development Effort Prediction: A Software Science Validation, *IEEE Transaction on Software Engineering*, vol.9 no.6, 1983, 639-648.

[3] A. J. Albercht, Measuring Application Development Productivity, *Proceeding of the joint SHARE/GUIDE/IBM*

Application Development Symposium, October, 1979, 83-92.

[4] B. Boehm, Industrial software metric top ten list, *IEEE Software*, September, 1987.

[5] R. G. Ebenau and S. H. Strauss, *Software Inspection Process*, McGraw-Hill, 1994.

[6] M. Fagan, Design and code inspections to reduce errors in program development, *IBM System Journal*, vol. 15, no.3, 1976.

[7] D. P. Freedman and G. M. Weinburg , *Handbook of walkthroughs, inspections and technical reviews: evaluating programs, projects, and products*, 3rd ed., 1982.

[8] T. Gilb and D. Graham, *Software Inspection*, Addison-Wesley, 1994.

[9] R. B. Grady, *Practical Software Metrics for Project Management and Process Improvement*, 1992.

[10] W. Humphrey, *Managing the Software Process* , Addison-Wesley, 1989.

[11] IEEE, ANSI/IEEE Std 830-1984: IEEE Guide to Software Requirements Specifications, 1984.

[12] IFPUG 1994 International Function Point User Group, Function Point Counting Practice Manual (Version 4), IFPUG, 1994.

[13] D. R. Jeffery, G. C. Low, and M. Barnes , A Comparison of Function Point Counting Techniques, *IEEE Transactions on Software Engineering*, vol.19 no.5 May, 1993, 529-532.

[14] J. C. Kelly, J. S. Sherif, and J. Hops, An Analysis of Defect Densities Found During Software Inspections, *Journal of Systems Software*, vol. 17, 1992, pp. 111-117.

[15] C. F. Kemerer and B. S. Porter, Improving the Reliability of Function Point Measurement: An Empirical Study, *IEEE Transactions on Software Engineering*, vol.18 no.11 November, 1992, 1011-1024.

[16] C. F. Kemerer, Reliability of Functions Points Measurement - A Field Experiment, *Communication of the ACM*, vol.36 no.2, 1993, 85-97.

[17] L. W. P. Kim, C. Sauer, and R. Jeffery, A Framework for software Development Technical Reviews, *Proceeding IFIP Conference on Software Quality*, Hong Kong, December, 1994.

[18] R. Madachy, L. Little, and S. Fan, Analysis of a successful Inspection Program, *SEW Proceedings*, 1993, 176-188.

[19] D. Parnas and D. Weiss, Active design reviews: principles and practices, *Proceeding of the 8th International Conference on Software Engineering*, August, 1985, 215-222.

[20] A. Porter and L. Votta, An experiment to access different defect detection methods for software requirements inspections, Proceeding of the 16th *International Conference on Software Engineering*, Italy, May, 1994.

[21] A. Porter, L. Votta, and V. Basili, Comparing detection methods for software requirements inspections: A replicated experiment, *IEEE Transaction on Software Engineering*, June, 1995.

[22] G. M. Schneider, J. Martin, and W. T. Tsai, An Experimental Study of Fault Detection in User Requirements Documents, *ACM Transaction on Software Engineering and Methodology*, vol. 1, no. 2, 1992, 188-204.

[23] G. C. Shirley, How Inspections fail, *The Software Practitioner*, May, 1993, 12-17.

[24] E. Yourdon, *Structured Walkthroughs*, 4th ed., Yourdon Press, 1989.