View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by CiteSeerX

Using Learning Styles of Software Professionals to Improve their Inspection Team Performance

Anurag Goswami[1], Gursimran Walia[2], Abhinav Singh[3]
Department of Computer Science, Office of International Services
North Dakota State University[1, 2], Indiana University[3]
anurag.goswami@ndsu.edu[1], gursimran.walia@ndsu.edu[2], singhab@iu.edu[3]

Abstract— **Inspections of software artifacts during early software development aids managers to detect early faults that may be hard to find and fix later. While inspections are effective, evidence suggests that inspection abilities of individuals vary widely which affect overall inspection effectiveness. Cognitive psychologists have used Learning Styles (LS) to measure an individual's characteristic strength and ability to acquire and process information. This concept of LS is being utilized in software engineering domain as a means to improve inspection performance. This paper presents the results from an industrial empirical study, wherein the LS's of individual inspectors were manipulated to measure its impact on the fault detection effectiveness of inspection teams. Using inspection data from nineteen professional developers, we developed virtual teams with varying LS's of individual inspectors and analyzed the team performance. The results from the current study show that, teams of inspectors with diverse LS's are significantly more effective at detecting faults as compared to teams of inspectors with similar LS's. Therefore, LS's can aid software managers to create high performance inspection team(s) and manage software quality.**

Keywords-software inspection; learning style; requirements.

I. INTRODUCTION

Inspecting early lifecycle artifacts (e.g., requirements and design) can improve software quality by helping developers detect faults early in the *Software Development Life Cycle* (SDLC). Empirical evidence showed that, finding and fixing faults earlier rather than later is easier, less expensive and saves significant rework costs [1]. To have most impact on software quality, researchers and practitioners have focused efforts on finding and fixing faults committed during the requirements development [2]. Requirements development is the first and a critical phase, wherein requirements are gathered from different technical (developers, designers, testers) and non-technical (managers, end-users) stakeholders. These requirements are recorded using *Natural Language* (NL) in a *Software Requirements Specification* (SRS) document. SRS is a means of communications amongst stakeholders but is prone to *mistakes* and *faults* due to inherently ambiguity, imprecision and vagueness in NL [3].

Among different approaches used for detecting NL requirement faults (e.g., NL to State transitions [4], checklist based inspections [5], scenario based reading [6], ad hoc inspections [7]), software inspections are widely recognized as most effective technique. Inspection process includes reviewing a software work-product by a group of skilled individuals to identify faults. Empirical evidence demonstrate the benefits of inspection on artifacts developed at all phases of development (e.g., requirement, design, code, interfaces) [8].

The phases in the inspection process defined by Fagan [9] are: 1) selecting skilled individuals/inspectors; 2) individual review to find faults; 3) team meeting to consolidate faults; 4) follow-up and repair. Fagan [9] emphasized different parts of the process (e.g., more emphasis on an individual preparation phase rather than team meeting phase). Regardless of the team meetings, evidence shows that the effectiveness (# of faults found) of an inspector during the individual review significantly impacts the overall effectiveness of an inspection team [10].

To improve the performance of inspectors during the individual review, researchers have tried to understand whether individual factors (e.g., educational background; level of technical degree) are correlated to their inspection effectiveness [11]. Contrary to the expectations, results at major software organizations showed that software engineers with a non-technical degree found significantly more number of requirement faults as compared to the technical degree holders [11]. Even when inspectors use same technique, and receive same training, their effectiveness varies significantly. These results led us to hypothesize that inspector's ability of detecting faults in a software artifact are affected by the ways with which they psychologically acquires, process and retains information (as opposed to their technical expertise and level of education).

On that end, cognitive psychology research [12] affirms that individuals vary in their abilities to perceiving and process information, i.e. they have varying *Learning Style* (LS) preferences and strengths. For example, some people like to think and work alone; some are more comfortable learning through concrete evidence and examples. Research results of LS in psychology prove that an individual perceive and process information better if it is presented in their preferred LS [13]. Our research extends the idea of individual LS's to evaluate its impact on the software inspection process.

While the concept of using LS in software engineering domain is novel, academia have experimented with creating heterogeneous teams to improve team performance [14]. Software Engineering researchers have also borrowed psychology research to improve inspection team performance [15]. As an example, researchers' used *Myers-Briggs Type Indicator* (MBTI) instrument (that measures psychological preference of individuals) to create heterogeneous inspection teams to maximizing disparity between team members. Despite these novel efforts, they have met with limited success because unlike LS instruments (that measure the learning preferences), MBTI is a personality inventory [16]. The only research linking LS's in software engineering domain [17] have been at studying the communication aspects of the stakeholders during requirements elicitation. Their research has shown that LS's of non-technical stakeholders should be considered when selecting

DOI reference number: 10.18293/SEKE2015-228

the requirements elicitation methods. The results also showed that software engineers (like other human beings) have different learning preferences. This result motivated us to evaluate if LS can aid in planning and performing the inspection.

We hypothesize that inspector's *Learning Styles* (LS) can be used to create heterogeneous inspection teams which in turn, would increase their team performance by detecting more unique faults (i.e. less fault overlap) during the inspection. To evaluate this hypothesis, this paper presents results of an industrial study on the effect of LS preferences of nineteen professional software engineers on their inspection team performance. The participants reported their LS's and individually inspected a requirements document using the fault-checklist technique and recorded faults. We analyzed the impact of LS's of inspectors by creating virtual inspection teams (by combining individual data) for different team sizes. Next, all virtual teams for each team size were sorted from most dissimilar to most similar in terms of the LS's of individual inspectors followed by an evaluation of their team performances. The results show that team of inspectors with dissimilar LS's performed significantly better than the teams of inspectors with similar LS's. Software managers can use these results to plan and manage inspections in their organizations.

II. BACKGROUND - MEASURING LEARNING STYLES

Kolb [18] introduced the concept of LS's, and developed the first LS instrument. Over the years, psychologists have developed different versions of LS models [19] and validated the use of LS's in engineering education [12]. Previous researches revealed that the *Felder and Silverman's Learning Style Model* (FSLSM) is the most advanced and widely used to measure the LS's preference of individuals [20]. The instrument used to measure LS is known as the *Index of Learning Styles* (ILS) [12] and is used in this research as below.

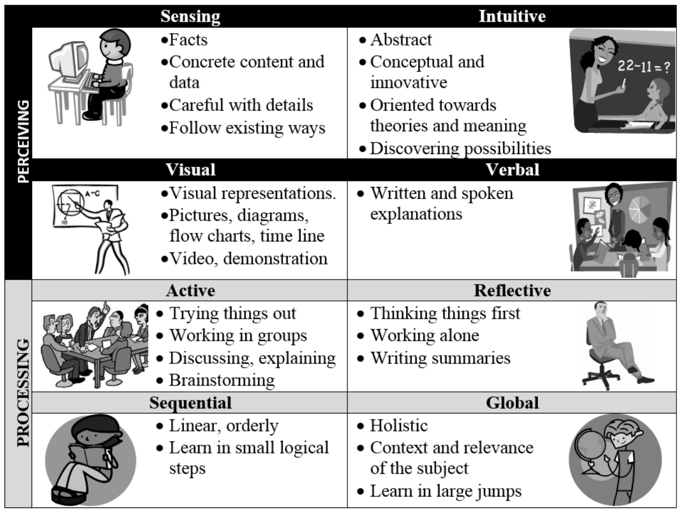
A. Felder and Silverman Learning Style Model (FSLSM)

The FSLSM model (shown in Fig. 1) capture most important LS preferences among individuals and then classifies characteristic strength and preference across four LS dimensions. These dimensions related to the way individuals "*perceive*" and "*process*" information. The two dimensions which relates to perceiving information includes: a) *Sensing*/*Intuitive*; and b) *Visual*/*Verbal*. The remaining two dimensions (i.e. *Active*/*Reflective* and *Sequential*/*Global*) relate to information processing. Brief description of LS model is described in Fig. 1.

We have used FSLSM and its accompanying instrument, *Index of Learning Style* (ILS), to measure the LS of inspectors.

B. Index of Learning Styles (ILS)

The ILS instrument has been empirically validated for its reliability and construct validity [21]. A sample ILS output is shown in Fig. 2. The ILS instrument is an online questionnaire with 44 questions. Each LS dimension has 11 questions. For example, in Visual/Verbal dimension, if a person selects 10 answers that favors visual category and 1 towards verbal category then the LS score will be 9 (i.e. 10-1) with a '*strong*' preference towards the visual category represented by a symbol 'X' on the top of the score (see Fig. 2). The symbol 'X' represents the preference towards a category in a LS dimension.



PERCEIVING	**Sensing**	**Intuitive**
	•Facts •Concrete content and data •Careful with details •Follow existing ways	• Abstract • Conceptual and innovative • Oriented towards theories and meaning • Discovering possibilities
	Visual	**Verbal**
	•Visual representations. •Pictures, diagrams, flow charts, time line •Video, demonstration	• Written and spoken explanations
PROCESSING	**Active**	**Reflective**
	• Trying things out • Working in groups • Discussing, explaining • Brainstorming	• Thinking things first • Working alone • Writing summaries
	Sequential	**Global**
	• Linear, orderly • Learn in small logical steps	• Holistic • Context and relevance of the subject • Learn in large jumps

Figure 1. Felder Silverman Learning Style Model

ILS score ranging from 1 to 3 represents that a person is balanced towards both the categories in an LS dimension. A score between 5-7 and 9-11 states that the person has a *moderate* and *strong* preference towards a category in a LS dimension.

III. RESEARCH APPROACH – USING LS TO FORM VIRTUAL INSPECTION TEAMS

The goal of the study was to be evaluate the impact of LS's on the inspection performance by creating multiple virtual inspection teams for varying number of inspectors (e.g., ranging from N=2 to N=10 inspectors) and then sorted from most dissimilar to most similar w.r.t the LS's of team members. This was followed by an evaluation of their inspection performances (*effectiveness*). To achieve this objective, a software tool was developed to automate this process by utilizing different multivariate statistical approaches. These approaches are briefly described here with more details in [22].

A. Principal Component Analysis (PCA)

FSLSM classifies each LS dimension into two categories (sensing/intuitive, visual/verbal, active/reflective and sequential/global). The relationship between two categories of each dimension is negatively correlated. That is, as score in one category increases, score in the other decreases. The **first** step was to transform the original correlated variables (i.e. LS scores across categories in each LS dimension) into uncorrelated variables. PCA is a multivariate technique that is being used to convert a set of observations of possibly correlated variables into set of values of uncorrelated variables called principal

```
ACT                                            X                 REF
       11   9   7   5   3   1   1   3   5   7   9   11
                            <-- -->

SEN                     X                                        INT
       11   9   7   5   3   1   1   3   5   7   9   11
                            <-- -->

VIS         X                                                    VRB
       11   9   7   5   3   1   1   3   5   7   9   11
                            <-- -->

SEQ                             X                                GLO
       11   9   7   5   3   1   1   3   5   7   9   11
                            <-- -->
```

Figure 2. Example result of the questionnaire on the ILS

components (PCs) [23]. PCA is used in this research to better understand the interrelationships between two categories (e.g., visual/verbal) of each of the four LS dimensions and between all the four LS dimensions for each individual. PCA transforms the original correlated data (i.e., FSLSM output, Fig. 2) into a new set of uncorrelated variables called principal components (PCs) [23]. ***Note***, for each individual in our research, the numbers of possible PC's are always equal to or less than the number of original variables (i.e., 8 categories across 4 LS dimensions)[24].

B. *Cluster Analysis (CA)*

During the ***second*** step, CA was used to group similar participants into different clusters based on their LS's. CA [23] is being used in our research to form groups/clusters of individuals based on their LS data. The resulting clusters of CA explain high similarity of LS's within each cluster and high dissimilarity of LS's between different clusters [25]. Among different types of clustering techniques (e.g. Hierarchical, Non-Hierarchical, Agglomerative, Divisive clustering); we have used k-means clustering algorithm [26]. CA groups the participants into clusters of similar LS, which helped us to study the relation between LS of members on the scale ranging from dissimilar to similar LS preferences. A team formed with different cluster members will lead to dissimilar LS group and a team formed from same cluster members leads to a similar LS group. More details of CA can be found here [22].

C. *Discriminant Analysis (DA)*

During the ***third*** step, DA was used to find out the probability of a participant belonging to a cluster. Using DA, LS variations are partitioned into a "*between group*" and a "*within-group*". This result of the DA is used to maximize the LS variations across different clusters, and minimize the LS variations within each cluster [27]. While CA explained that there is more dissimilarity among different clusters, there is a lack of dissimilarity in the LS preferences of the individuals belonging to the same cluster. DA provides *Group Membership* (GM) to determine the dissimilarities between individual LS's within the same cluster and with respect to the individuals in other clusters. So, DA provides GM values for each individual w.r.t each cluster. GM was used in our study to sort the teams ranging from most dissimilar LS to most similar LS preferences and strengths.

This process of extracting software inspection teams with varying levels of LS preferences was automated. Details of evaluating the performance of these teams appears in Section V.

IV. EMPIRICAL STUDY DESIGN

To evaluate the impact of LS variability on the inspection performance, LS's of nineteen professionals (working in IT Company) were gathered via online survey questionnaire. The participating subjects were trained on the inspection process and on using the fault checklist to record faults found during the inspection. Next, each subject individually inspected an industrial strength requirements document (that was seeded with faults) and reported faults. Study details are provided below.

Research Question*:* Whether the variation in the LS's of individual inspectors is positively correlated to their team performance during an inspection of requirements document?

Variables*:* The study manipulated the LS's of individual inspectors (*independent variable*) and measured its effect on the team effectiveness (*dependent variable*) during the inspection.

Participating Subjects*:* Nineteen software professionals working in a software company participated in the study. Some of them have worked on multiple projects in industry. The subjects' reported to have an average of three years of experience in interacting with user to writing and inspecting requirements and use cases.

Artifacts*:* The document inspected in the study described the requirements for the Loan Arranger system (LAS). LAS is responsible for grouping loans into bundles based on user-specified characteristics and then sell to other financial institutions. For use in previous studies, the document was written in plain English, was 10 page long, and seeded with thirty realistic faults. The fault seeding was done by Microsoft researchers prior to the study. The document is publicly available[1] and have been used in several inspection studies [28, 29].

Experiment Procedure*:* Study steps as described below:

Step 1 – Pre study survey: participants were asked to fill pre-study survey questionnaire to provide feedback about their experience of working in software industry. The survey elicited information about their experience in interacting with end-users to write requirements, writing use cases, inspecting requirements, and changing requirements for maintenance.

Step 2 – Learning Style Questionnaire Survey: participants were given Felder Silverman's LS questionnaire. Participants answered 44 multiple choice questions[2] and, the LS results are generated for each participant on ILS scale (Fig. 2). For each dimension on ILS (Active/Reflective, Sensing/Intuitive, Visual/Verbal, and Sequential/Global), the participant has score towards one category. Hence, only four LS categories (from each dimension) form LS of an individual with a score of either 1 or 3 or 5 or 7 or 9 or 11. These scores are then converted into actual scores which has scores in both the categories (i.e. number of answers supported for each category in a dimension) as shown for a subset of 10 (out of 19) subjects in Table I. For example, in Active/Reflective dimension, subject ID 9 answered 8 questions in favor of Active and 3 in favor of Reflective.

TABLE I. EXAMPLE OF ACTUAL SCORES OF 10 PARTICIPANTS

ID	ACT	REF	SEN	INT	VIS	VER	SEQ	GLO
1	5	6	8	3	9	2	6	5
2	5	6	7	4	6	5	6	5
3	9	2	9	2	10	1	5	6
4	3	8	7	4	4	7	7	4
5	4	7	10	1	11	0	6	5
6	4	7	5	6	9	2	4	7
7	5	6	8	3	11	0	9	2
8	3	8	5	6	4	7	10	1
9	8	3	6	5	8	3	3	8
10	4	7	10	1	6	5	6	5

[1] *http://steel.cs.ua.edu/~carver/BackgroundReplication*
[2] *https://www.engr.ncsu.edu/learningstyles/ilsweb.html*

Step 3 – Training and Inspecting LAS Requirements: The subjects were trained (by the same researcher in a single session) on basic concepts in an inspection and how to detect faults in a requirements document using fault checklist technique. The subjects were instructed on different fault types and how to use the fault form to record faults during the inspection using example requirements. Then, the subjects were asked to work alone and performed an inspection of LAS to identify and record faults. To normalize the results, the subjects were provided sixty minutes to perform the inspection (read the document and record faults). At the end of inspection process, nineteen fault lists were collected (one per subject). One of the researchers read through the faults reported by each participant (and compared against the seeded fault list) to remove any false-positives before analyzing the data. In addition, the fault reporting forms required the subjects to classify the faults identified during the inspection into one of the following fault types: *Omission* (O), *Ambiguous Information* (A), *Inconsistent Information* (II), *Incorrect Fact* (IF), *Extraneous* (E), and *Miscellaneous* (M).

V. EVALUATION CRITERIA

This section describes the process used to form virtual inspection teams (using individual LS data), sorting teams (w.r.t LS dissimilarity) and evaluating their team performance (using individual fault data). Our previous results showed that, for cost-effective inspections, team sizes should be limited to ten inspectors due to high cost of inspections and diminishing return beyond a size of 10 inspectors [30]. Therefore; the tool generated all possible virtual inspection teams for inspection team size from N=2 to N=10 inspectors. For each inspection team size, the tool sorts the virtual teams in the decreasing order of LS dissimilarity of the inspectors. The tool then outputs the total unique faults found by each team and their fault detection rate. The evaluation steps are described in subsections V.1-4.

1) Creating Virtual inspections: We created virtual inspection teams (i.e. teams that did not actually meet, we just combined their data) with team size ranging from 2 to 10 inspectors and each team size has all the possible combinations of virtual teams. For example, to create virtual inspection team of size 4 (from a pool of 19 inspectors), we created 3876 virtual inspection teams (i.e. $19C_4$).

2) Grouping of similar inspectors in clusters: The correlated LS of inspectors in each LS dimension were converted into uncorrelated variables by the tool using PCA (Section III.A). Next, inspectors of similar LS's were grouped together in the same cluster (number of cluster is same as the team size being analyzed) using CA (Section III.B).

3) Sorting teams based on the LS of inspectors: In this step, the tool calculates the group membership (GM) of each inspector in a cluster using DA. Next, tool sorts all inspection teams (i.e. $32C_4$, from step 1) in the order of decreasing level of dissimilarity (i.e. most dissimilar to similar) in the LS's of the individual inspectors.

4) Evaluating inspection performance of teams: During this step, the tool combines the individual inspection data and outputs the total unique faults and average time taken by each virtual inspection team of all sizes. To summarize, all possible inspection team were formed for each team size and their virtual inspection results were organized from teams with dissimilar to similar LS's along with their performance.

VI. DATA ANALYSIS AND RESULTS

This section presents the results on the; 1) effect of variation in the LS's on the inspection team performance; 2) distribution of fault types (mentioned in Section IV) across different LS dimension and categories.

As stated earlier, for each team size (e.g., N=4), all possible virtual teams were generated and then sorted with dissimilar LS's (highest number of cluster involved in team formation) to similar (least number of cluster involved) LS's. Inspection data (i.e. faults found by each participant) was individual data; so fault detection *effectiveness* for virtual teams was calculated by combining the unique faults detected by each participant in LAS requirements document. This analysis was performed on all possible virtual inspection teams for all sizes.

Fig. 3 compares the average number of unique faults found by virtual inspection teams ranging from N= 2 to 10 inspectors. Each line represents a particular team size (e.g., N=4) and maps the average number of faults found by the virtual inspection teams (formed with a certain # of clusters). Results are organized by the increasing number of clusters (or increasing dissimilarity) involved in the team formation (i.e., the higher the cluster number, the more dissimilarity the team members). Also mentioned earlier, the *# of clusters* that could participate in the team formation is always less than or equal to the team size (e.g., 1 or 2 or 3 clusters for team size 3).

Based on the results in Fig. 3, a general observation is that, for each team size, teams with highest number of clusters involved (i.e. most dissimilar inspectors) found maximum average number of faults as compared to the same team size with less number of clusters. The results show a consistent increase in the inspection effectiveness with an increase in the number of clusters used to form teams. For example, in team size 5, teams created with only one cluster (i.e. most similar teams) found an average of 13.42 faults; whereas teams created from five different clusters (i.e. most dissimilar team) found an average of 17.45 faults. This effectiveness trend is consistent across all team sizes.

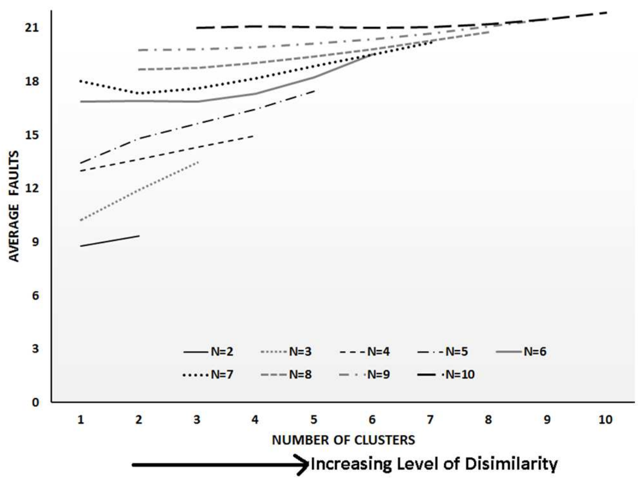


Figure 3. Team effectiveness organized by increasing # of clusters

As an exception (in Fig. 3), for some larger team size (e.g., team size 10), virtual inspection teams could not be formed from a single cluster. This is because for team size 10, the tool creates 10 different clusters via k-means algorithm. All 19 participants were distributed across these 10 clusters and there was no cluster that contained all 10 participants. Therefore, as the team size increases, the number of participants that belong to the same cluster decreases which reduces the probability that a team will be formed from less number of clusters

Based on the above results, teams of inspectors with dissimilar LS's had less fault overlap and consequently, their inspection effectiveness is higher.

To evaluate this effect, we performed a linear regression test to see whether the dissimilarity in the LS's of inspectors is positively correlated with the average number of unique faults found by inspection teams of different sizes. The results (Table II) show that, dissimilarity in the LS of inspectors had a strong and significant positive correlation with the team effectiveness for team size 3 to 10 (shaded rows). We anticipate that increasing the team size beyond a certain number of inspectors would not significantly diversify the LS's of inspectors in a team (due to 8 possible LS categories). Generally, software companies do not employ a large inspection teams (which is the reason we had analyzed up to team of 10 inspectors). Overall, creating inspection teams based on the dissimilarity in their LS strengths (guided by the number of clusters involved in their formation) appears to increase the fault detection effectiveness during an inspection of requirements document.

TABLE II. LINEAR REGRESSION RESULTS

Team size	Effectiveness Correlation
2	*P=0.14; Correl. Coeff = 0.112; r^2=0.013*
3	*P<0.001; Correl. Coeff = 0.387; r^2=0.015*
4	*P<0.001; Correl. Coeff = 0.185; r^2=0.034*
5	*P<0.001; Correl. Coeff = 0.268; r^2=0.072*
6	*P<0.001; Correl. Coeff = 0.228; r^2=0.052*
7	*P<0.001; Correl. Coeff = 0.292; r^2=0.085*
8	*P<0.001; Correl. Coeff = 0.211; r^2=0.044*
9	*P<0.001; Correl. Coeff = 0.166; r^2=0.028*
10	*P<0.001; Correl. Coeff = 0.060; r^2=0.004*

We analyzed the impact LS dimensions (made up of combination of categories) had, on inspection effectiveness and nature of fault found. The was done to gain insights about whether certain LS are responsible for higher inspection effectiveness and detection of particular fault type or distributed across different LS dimension? To perform this analysis, we captured the classification of faults according to their fault type (Section IV) found by the participants belonging to each cluster. From raw data it was found, none of the inspectors had a preference towards *Verbal*-VER. The remaining LS categories (*Active*–ACT, *Reflective*–REF, *Sensing*–SEN, *Intuitive*–INT, *Sequential*–SEQ, and *Global*–GLO) were analyzed.

To analyze the effect of each LS (i.e. combination of categories across each dimension) on inspection effectiveness and fault types, we created groups of each LS using six LS categories across three LS dimensions. Therefore, eight clusters with their respective number of members were: *REF-SEN-GLO (five), ACT-SEN-SEQ (five), ACT-INT-SEQ (one), REF-SEN-SEQ (one), REF-INT-SEQ (two), ACT-SEN-GLO (one), ACT-INT-GLO (two),* and *REF-INT-GLO (one).* Fig. 4 shows the

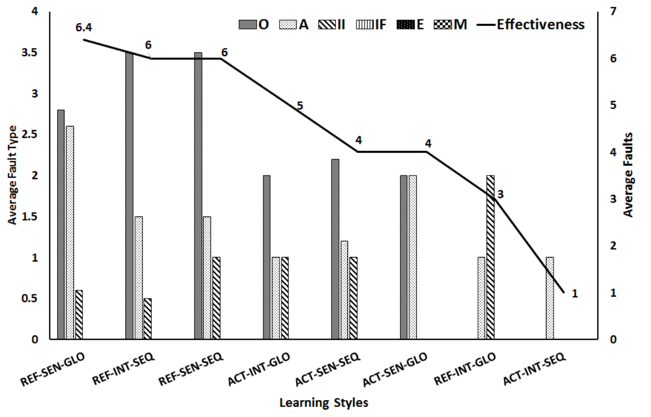


Figure 4. Effectiveness and Fault Type by each LS

average inspection effectiveness (shown by solid line) and *average fault type* (shown by bars) correlation for each LS cluster during inspection. Results are ordered from most effective to least effective cluster. Left y-axis shows the average number of different fault type detected and secondary y-axis on the right shows the average inspection effectiveness.

Based on results in Fig 4, following observations were made:

a) Inspectors with REF-SEN-GLO LS's had the maximum inspection effectiveness (6.4) and close to that, REF-INT-SEQ LS cluster found the next maximum effectiveness (6). Upon analyzing the pre-study survey data, it was found that inspectors with REF-SEN-GLO LS preference had high experience of working with requirements analysis as compared to inspectors with REF-INT-SEQ LS preference.

b) Inspectors belonging to the REF-SEN-GLO (i.e. most effective cluster) found the maximum number of Ambiguous Information fault (A). This result suggests that inspection performance rely on a combination of categories along each LS dimension and a single LS category cannot help detect all types of faults. Hence, there should be an inspection team with inspectors of diverse LS's.

c) Another observation is that REF-INT-SEQ and REF-SEN-SEQ cluster found the maximum number of Omission (O) faults. Also, REF-INT-GLO cluster found maximum number of Inconsistent Information (II) faults.

These results reinforce that, ***a combination of different LS's enabled inspectors to find faults of different types*** and that ***difference in the LS of inspectors enable a higher coverage of faults present in an artifact***. The result however revealed that inspectors belonging to the five LS clusters (i.e., *REF-SEN-GLO, REF-INT-SEQ*, *REF-SEN-SEQ*, *ACT-INT-GLO, ACT-SEN-SEQ*) out of 8 clusters were able to uncover all the three types (O, A, and II) of faults present in requirements document.

VII. THREATS TO VALIDITY

In this experiment, we were able to address some of the validity threats. Participating subjects were software professionals working in real industry settings. Heterogeneity of document was handled by providing participants with the same LAS document to inspect. Our experiment consists of inspectors who has different levels of work experience due to which group composition effect was not addressed. Training was provided by one trainer to all the participating subjects which addressed the training bias. We were also able to address fatigue effect by providing enough time to participants to take surveys (i.e. LS questionnaire, pre-study, and post-study survey) and perform requirements inspection in their comfortable environment where they can take break(s). However, the LAS document was developed externally by Microsoft and we do not have access of LS of authors which did not led us to control the effect of LS of authors of SRS on inspection output. Also, for larger team size (e.g., team size 10), there was not enough data to form virtual inspection teams with small number of clusters. These issues regarding the generalization of results will be addressed in future studies.

VIII. Discussion of Results

The focus of our study was to investigate the impact of individual inspector on performance of inspection team during the requirements inspection. The results from Section VI (Fig. 3) showed that dissimilarity in LS of inspectors had a direct and positive relationship with inspection team effectiveness. This means, higher the dissimilarity in the LS of inspectors in a team, the more number of faults are detected in requirements document during the inspection (i.e. higher inspection output). While testing this results statistically (Table II), there was a strong significant correlation between the LS dissimilarity and inspection effectiveness. Therefore, using LS's as an input to guide staffing/formation of inspection teams is beneficial for software managers. Results also revealed that some LS do favor inspection positively (i.e. high effectiveness and on detection of different fault type) as compared to other LS's. Based on the results provided in this paper, the concept of LS is applicable in software inspections domain and can help to manage the quality of software by creating high performance inspection team(s).

IX. Conclusion and Future Work

While the data size used in this study was small, these results showed that if inspection teams are created by taking different LS of inspectors into account, they would read requirements document with different perspectives. This results in less fault overlap among inspectors in a team and leads to high inspection output. These results are interesting and provide us with initial evidence to continue on this path. We plan to analyze the pre/post study data to gain more insights into the LS's of professional developers and its impact on their daily activities. Our future works includes replicating this analysis for larger data sets. Another future work includes analyzing the data to evaluate the correlation (positive or negative) inspectors with certain LS preferences (e.g., *Active vs. Reflective)* may have on their performance during the requirements inspections.

References

[1] Perry, W.E.: 'Effective Methods for Software Testing: Includes Complete Guidelines, Checklists, and Templates' (John Wiley & Sons, 2006. 2006)

[2] Ackerman, A.F., Buchwald, L.S., and Lewski, F.H.: 'Software inspections: an effective verification process', Software, IEEE, 1989, 6, (3), pp. 31-36

[3] Berry, D.M., and Kamsties, E.: 'Ambiguity in requirements specification': 'Perspectives on software requirements' (Springer, 2004), pp. 7-44

[4] Aceituna, D., Do, H., Walia, G.S., and Lee, S.-W.: 'Evaluating the use of model-based requirements verification method: A feasibility study'. Empirical Requirements Engineering (EmpiRE), 2011 First International Workshop on2011 pp. 13-20

[5] Parnas, D.L., and Lawford, M.: 'The role of inspection in software quality assurance', Software Engineering, IEEE Transactions on, 2003, 29, (8), pp. 674-676

[6] Shull, F., Rus, I., and Basili, V.: 'How perspective-based reading can improve requirements inspections', Computer, 2000, 33, (7), pp. 73-79

[7] Porter, A.A., Votta Jr, L.G., and Basili, V.R.: 'Comparing detection methods for software requirements inspections: A replicated experiment', Software Engineering, IEEE Transactions on, 1995, 21, (6), pp. 563-575

[8] Fagan, M.E.: 'Design and code inspections to reduce errors in program development': 'Pioneers and Their Contributions to Software Engineering' (Springer, 2001), pp. 301-334

[9] Fagan, M.E.: 'Advances in software inspections': 'Pioneers and Their Contributions to Software Engineering' (Springer, 2001), pp. 335-360

[10] Porter, A., Siy, H., Mockus, A., and Votta, L.: 'Understanding the sources of variation in software inspections', ACM Transactions on Software Engineering and Methodology (TOSEM), 1998, 7, (1), pp. 41-79

[11] Carver, J.: 'The impact of background and experience on software inspections', Empirical Software Engineering, 2004, 9, (3), pp. 259-262

[12] Felder, R.M., and Silverman, L.K.: 'Learning and teaching styles in engineering education', Engineering education, 1988, 78, (7), pp. 674-681

[13] Allert, J.: 'Learning style and factors contributing to success in an introductory computer science course', (IEEE, 2004), pp. 385-389

[14] Rutherfoord, R.H.: 'Using personality inventories to help form teams for software engineering class projects', SIGCSE Bull.,2001, 33,(3),pp.73-76

[15] Miller, J., and Yin, Z.: 'A cognitive-based mechanism for constructing software inspection teams', Software Engineering, IEEE Transactions on, 2004, 30, (11), pp. 811-825

[16] Montgomery, S.M.: 'Addressing diverse learning styles through the use of multimedia'. Frontiers in Education Conference, 1995. Proceedings., 19951995 pp. 3a2. 13-13a12. 21 vol. 11

[17] Aranda, G.N., Vizcaíno, A., Cechich, A., and Piattini, M.: 'A cognitive-based approach to improve distributed requirements elicitation processes'. Cognitive Informatics, 2005.(ICCI 2005). Fourth IEEE Conference on, Irvine, USA, 8-10 Aug. 2005 pp. 322-330

[18] Kolb, D.A.: 'Experiential learning: Experience as the source of learning and development' (Prentice-Hall Englewood Cliffs, NJ, 1984. 1984)

[19] Charkins, R., O'Toole, D.M., and Wetzel, J.N.: 'Linking teacher and student learning styles with student achievement and attitudes', Journal of Economic Education, 1985, pp. 111-120

[20] Felder, R.M.: 'Are learning styles invalid?(Hint: No!)', On-Course Newsletter, 2010, pp. 1-7

[21] Felder, R.M., and Spurlin, J.: 'Applications, reliability and validity of the index of learning styles', International Journal of Engineering Education, 2005, 21, (1), pp. 103-112

[22] Goswami, A., and Walia, G.:'Using Learning Styles to Create Virtual Inspection Teams: A Technical Report', http://www.goswamianurag.com/techRep/LSTeamsTech.pdf, The Department of Computer Science, North Dakota State University, 2015

[23] Anderson, T.W.: 'An introduction to multivariate statistical analysis' (Wiley New York, 1958.)

[24] 24 Jolliffe, I.T.: 'Principal component analysis' (Springer verlag, 2002. 2002)

[25] Steinbach, M., Ertöz, L., and Kumar, V.: 'The challenges of clustering high dimensional data': 'New Directions in Statistical Physics' (Springer, 2004), pp. 273-309

[26] Hartigan, J.A., and Wong, M.A.: 'Algorithm AS 136: A k-means clustering algorithm', Journal of the Royal Statistical Society. Series C (Applied Statistics), 1979, 28, (1), pp. 100-108

[27] Tatsuoka, M.M., and Tiedeman, D.V.: 'Chapter IV: Discriminant Analysis', Review of Educational Research, 1954, 24, (5), pp. 402-420

[28] Carver, J., Shull, F., and Basili, V.: 'Observational studies to accelerate process experience in classroom studies: an evaluation'. Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on2003 pp. 72-79

[29] Shull, F., Carver, J., and Travassos, G.H.: 'An empirical methodology for introducing software processes', ACM SIGSOFT Software Engineering Notes, 2001, 26, (5), pp. 288-296

[30] Mandala, N.R., Walia, G.S., Carver, J.C., and Nagappan, N.: 'Application of kusumoto cost-metric to evaluate the cost effectiveness of software inspections'. Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, Lund, Sweden, 17-22 Sep. 2012 pp. 221-230