SOFTWARE INSPECTION TEAM FORMATION BASED ON THE LEARNING STYLE OF INDIVIDUAL INSPECTORS

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Haribabu Bavanari

In Partial Fulfillment of the Requirements
For the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

October 2012

Fargo, North Dakota

North Dakota State University

Graduate School

Title

SOFTWARE INSPECTION TEAM FORMATION BASED ON THE

LEARNING STYLE OF INDIVIDUAL INSPECTORS

By

HARIBABU BAVANARI

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Gursimran Walia
Chair (typed)

Dr. Simone Ludwig

Dr. Kendall Nygard

Dr. Luis Del Rio

Approved by Department Chair:

10/29/12	Dr. Kenneth Magel (Associate Head)
Date	Signature

ABSTRACT

To improve the software quality, researchers have focused their effort on developing and validating effective methods of finding and fixing defects early in the development process. *Software inspections* are most widely used defect detection method. Also, researchers showed that the overall effectiveness of an inspection team is affected by the effectiveness of individual inspectors. But researchers have not been able to completely understand the inherent characteristic that makes an individual inspector effective. This paper investigates this problem by analyzing the learning style (LS) preferences of individuals who make up inspection team. Also presents a tool that provides ability to researchers to study the relationship between inspectors' LS and his/her effectiveness in uncovering defects in software requirement document. Cluster and Discriminant analysis techniques were used to sort inspection teams based on their LS preferences. Researchers can use this tool to study further correlations between inspector's LS and their performance in team.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my advisor Dr. Gursimran Singh Walia for his continued support throughout this paper. I appreciate his time, assistance and continuous guidance. I would also like to thank Dr. Kendall Nygard, Dr. Simone Ludwig, and Dr. Luis Del Rio for being a part of my graduate supervisory committee. Special thanks to the faculty and staff of the Computer Science Department for their unconditional support throughout my master's program. Finally, I am grateful to my parents who are the reason for all my achievements and have supported and motivated me throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

In today's world of modern computing, organizations depend on the software systems for automation of their tasks to reduce time and cost. Successful software organizations thrive on delivering quality software products within allocated time and budget [1]. The software development process involves the translation of information from one form into another (i.e., from customer's needs to requirements to design to a working product). Defects are introduced at various points during the development, if left undetected are shipped along with the product. Most of the defects are introduced at the early stages of the project, which propagate to the later phases where they are harder to find and fix. Software artifacts produced during the early phases of software development (requirement and design documents) help to describe the functionality and design of software to be developed. Therefore, successful software organizations focus their attention on finding and correcting defects in requirement and design phases to avoid their propagation into later phases.

Software requirement analysis and determination is a tedious process that involves stakeholders who are technical (e.g., requirement engineers, programmers) and non-technical (e.g., end users, sponsors) personnel. Therefore, while the formal modeling languages (e.g., programming languages) can be used to develop software solution, the natural language (NL) is used to describe the customer needs and problems. The output of this stage is a *Software Requirements Specification* (SRS) artifact that includes requirements for a software product to be developed, written in NL. The purpose of SRS is to communicate customer's requirements, to further phases of software lifecycle which transforms the NL into conceptual modeling to build the software that meet the requirements.

While NL is used to describe the needs or problems, the inherent nature of English language causes the requirements to be complex, imprecise, vague, and ambiguous. Furthermore, writing requirements in NL often leads to defects in the document, because different people have abilities to interpret NL in different ways due to its ambiguous nature and lack of common understanding among different stakeholders. To make the matters worse, large amount of cost in software development is involved in eliminating defects during the later stages of the software process that can be traced back to the mistakes or misunderstandings during the requirements development [38, 39]. In addition, finding and fixing faults earlier rather than later is easier, less expensive and reduces avoidable rework.

Among various methods used for early detection and removal of faults, software inspections have been empirically validated [2, 3, 4, 5]. Inspections are a process whereby software artifacts are examined by a group of inspectors to ensure that they meet a set of quality constraints by uncovering faults in the artifact. The main idea of the inspection as defined by Fagan [2] is as follows. Once the author completes a software artifact, which could be a requirements document, design, or code, it is submitted for an inspection. The inspection consists of multiple steps. The inspection leader first chooses a team of skilled individuals who will perform the inspection. Then, the document to be inspected is distributed to these team members. The team-members individually review a software work-product to identify faults and meet together to consolidate the faults into a list, which is returned to the document author who then will fix the reported faults. Since the initial definition of the inspection process, many variations have been made on it (e.g., placing more emphasis on individual review). Still, the main goal of the software inspection is to remove the faults in the software work product and enable; 1) saving of cost and time, which needs to be expended if the faults pass to later stages of software development; and 2) improving the quality of software product by enhancing its reliability, maintainability and availability [2, 3, 39, 40].

While inspections are an effective verification method, its success is heavily dependent on the individuals overall success in uncovering defects. Previous research in software inspection process has shown that the variations among inspection team members can have a greater effect on the outcome of the inspection than the variations in the actual inspection process [6]. Previous research has also tried to evaluate if the level of the technical background or education of the inspectors can impact their ability to find more number of defects during an inspection [6, 41]. The result shows that the technical computer science knowledge or their degree (PhD vs. Master vs. Bachelor) did not have any effect on the fault detection abilities of software inspectors.

This result has led us to hypothesize that, the inspector's ability of finding defects in a requirements document are affected by their individual strengths and preferences in the ways they comprehend and process information – i.e., their individual *learning styles* (LS). The ways in which an individual characteristically acquires, retains, and retrieves information are collectively termed the individual's LS by the psychologists [42]. An appropriate definition of LS preference was provided by Keefe [43], and is consistent with the use of LS in this paper:

> "*Learning styles are characteristic cognitive, affective, and psychological behaviors that serve as relatively stable indicators of how learners perceive, interact with, and respond to the learning environment*".

On that note, previous research [7, 8, 9] in psychology has shown that individual students have different LS's. The psychologists mention that, some individuals need concrete information (e.g., facts, experiment data) while others understand abstractions better. Some individuals prefer if information is presented to them visually, while other prefers verbal explanations. Some are more active learners (i.e., by trying and analyzing) whereas others are more reflective learners (i.e., understanding the information completely before attempting to analyze it) [7].

Psychologists have developed and empirically evaluated different learning styles models over a period of time. Some of the most relevant learning style models include: T*he Myers-Briggs Type Indicator (MBTI)*, *Kolb's Learning Style Model*, and the *Felder-Silverman Learning Style Model (FSLSM)*. Of these three models, the FSLSM describes the learning style of a learner in most detail and includes the preference dimensions parallel to other learning style models, although the distinction between preferences on four dimensions is unique [10]. According to this model a learner is classified in four dimensions as follows, with more details appearing later:

- *Sensing learners* (Oriented towards facts, concrete content, data, hands-on work, practical, apply theory in practice) or *Intuitive learners* (abstract, conceptual, innovative, oriented toward theories and meanings, discovering possibilities);
- *Visual learners* (prefer visual representations of presented material - pictures, diagrams, flow charts, time line, video, demonstration) *or Verbal learners* (prefer written and spoken explanations);
- *Active learners* (learn by trying things out, working in groups, discussing, explaining, brainstorming) or *Reflective learners* (learn by thinking things through, working alone, writing summaries);
- *Sequential learners* (linear, orderly, learn in small logical steps) or G*lobal learners* (holistic, context and relevance of the subject, learn in large jumps).

Empirical studies conducted by educational psychologists have also shown that a student understands and retain information better if the information is presented to them in his/her preferred modes of learning style (LS). Conversely, students are tend to be bored, inattentive, and perform poorly on tests if there is mismatch between the learning styles of students and the teaching style of an instructor [7, 42, 44]. This paper attempts to utilize the Learning Style (LS) preferences of individual inspectors in order to form inspection teams, that will improve the defect detection effectiveness of individual inspectors and the team's overall effectiveness.

Like any other individual, software developers (and software inspectors) have different learning styles that affect their ability to process and comprehend information presented in the software artifact under review. Individual inspectors, depending on their learning style strength, can find different type of defects present in a software artifact. For example, if individual inspectors participating in an inspection process have similar learning style preferences, then they are more likely to find same type of defects present in an artifact. On the other hand, teaming inspectors based on a diverse set of learning style preferences can help reduce the overlap of fault among different inspectors and increase the total number of faults found by all the inspectors.

Additionally, the personnel inspecting a software artifact are often different from the ones who were involved in the development of the software artifact. Therefore, a mismatch of the learning styles between the developers of an artifact and inspectors of the same artifact can negatively impact the inspection effectiveness (i.e., the number of unique defects found by all the inspectors). This scenario is especially true in the requirements phase, where the requirements are elicited, analyzed and documented in NL by software engineers working within an organization, while the software inspectors assigned to review the requirements document are a mix of technical and non-technical personnel from either within or outside the organization.

Therefore, this paper presents a tool that intends to help researchers and practitioner to investigate the impact of learning style preferences on the effectiveness of requirements inspection. To accomplish that, the software tool presented in this paper enables grouping inspectors according to their learning preferences (ranging from most similar to most dissimilar for each inspection team size) so that its effect on the number of defects uncovered during an inspection can be analyzed.

The paper is outlined as follows. Section 2 summarizes and discusses Learning Styles, including the Felder-Silverman Learning Style Model we have adopted for study. It also discuss about the questionnaire through which subjects Learning Styles were derived. Section 3 presents research approach which includes principal component analysis, clustering analysis and discriminant analysis that is used to develop teams with varying level of LS differences in individual team members for varying inspection team sizes. In Section 4, we described the research design followed by the actual tool description in Section 5, that would allow the researchers and other users to form software inspection teams based on the LS's of individual inspectors; and produces the inspection output of the resulting teams. Sections 6 discuss the impact of the tool for the researchers and practitioners. Finally, in Section 7, we provide conclusions and future enhancements on the tool.

2. BACKGROUND RESEARCH

This section provides background information regarding the LS models (Section 2.1), the detailed description of the LS categories used in the *FLSLS model* (Section 2.1.1), and the details of the survey instrument used to assess the individual learning preference (Section 2.1.2). Section 2.2 provided background information on the use of LS's in the context of Software Engineering.

2.1. Learning Style Models

Kolb [11] introduced the concept of learning styles, and is credited with the development of the first learning styles instrument. Over the years, educational psychologists have developed different variations of the learning style models [42, 43, 45, 46, 47, 48, 49] and validated the use of learning styles in engineering education [7]. The literature review of the publications referring to the learning styles revealed that the Felder and Silverman's learning style model is most advanced and widely used to assess the learning styles [12, 13]. It is measured through an instrument called the Index of Learning Styles (ILS) by Felder and Soloman [14]. The following subsections explain the learning style model developed by the Felder and Silverman and the ILS instrument used for measuring the learning style preference.

2.1.1. Felder-Silverman learning style model (FLSSM)

Richard Felder and Lisa Silverman developed a learning style model entitled "*Felder-Silverman Learning Style Model*" that captures the most important learning style differences among the engineering students [7]. The model classified individuals as having strengths and preferences for one category or the other on the basis of four dimensions that related to the way individuals "perceive information" and the way they "process information". The two dimensions that 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. This is illustrated in Figure 1 and is discussed below.

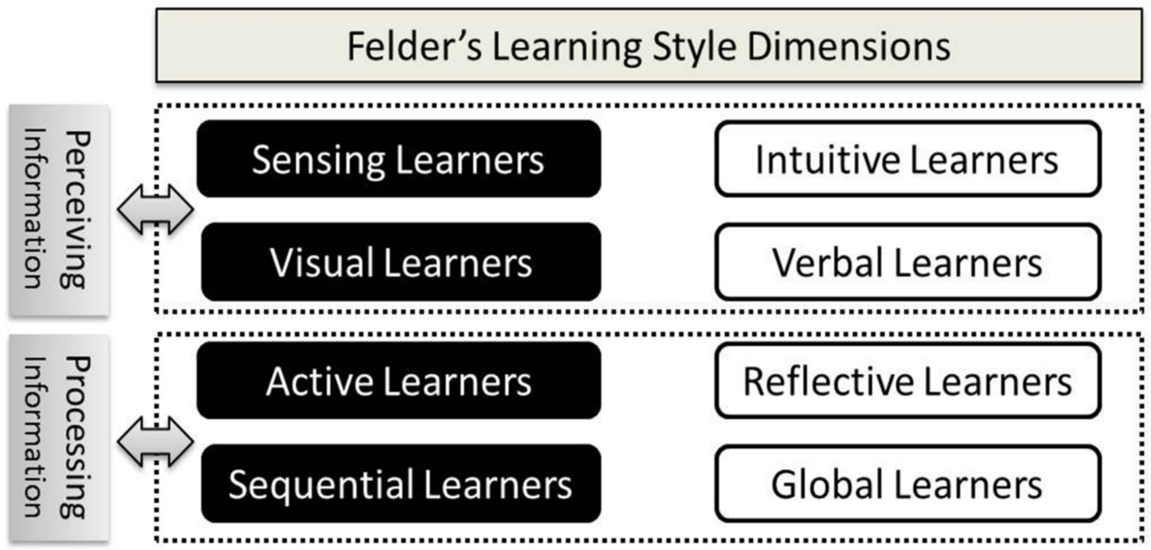


Figure 1. Felder-Siverman learning style model

The first dimension of the learning style model classified subjects as "*Sensing*" or "*Intuitive*" learners as discussed follows:

a) *Sensing* people prefer learning facts. They like solving problems by well-established methods and dislike complications and surprises. Sensors tend to be patient with details and good at memorizing facts and doing hands on (laboratory) work. Furthermore, sensing learners are considered as more realistic and sensible; they tend to be more practical than intuitive learners and like to relate the learned material to the real world;

b) *Intuitive* people often prefer discovering possibilities and relationships. They like innovation and dislike repetition. They tend to work faster and to be more innovative than sensors. Intuitive learners do not like work that involves a lot of memorization and routine calculations.

The second dimension of the learning style model classified subjects as “*Visual*” or “*Verbal*” learner as discussed follows:

a) *Visual* people remember best what they see (such as pictures, diagrams, flow charts, time lines, films, and demonstrations). They prefer visually presented information;
b) *Verbal* people get more out of words, and written and spoken explanations. They prefer verbally presented information

The third dimension of the learning style model classified subjects as “*Active*” or “*Reflective*” learner as discussed follows:

a) *Active* people tend to retain and understand information by doing something active with it (discussing or applying it or explaining it to others). "Let's try it out and see how it works" is an Active's phrase. Furthermore, they tend to be more interested in communication with others and prefer to learn by working in groups where they can discuss about the learned material;
b) *Reflective* people prefer to think about information quietly first. "Let's think it through first" is the Reflective’s response. Regarding communication, they prefer to work alone or maybe in a small group together with one good friend.

The last dimension of the learning style model classified subjects as “*Sequential*” or “*Global*” learner as discussed follows:

a) *Sequential* people tend to gain understanding in linear steps, with each step following logically from the previous one. They tend to follow logical stepwise paths in finding solutions. They may not fully understand the material but they can nevertheless do something with it (like solve homework problems or pass a test) since the pieces are logically connected;

b) *Global* people tend to work in large jumps, absorbing material almost randomly without seeing connections, and then suddenly "getting it". They may be able to solve complex problems quickly or put things together in novel ways once they have grasped the big picture, but they may have difficulty explaining how they did it.

The index of learning styles (ILS) is an online questionnaire designed to assess preferences on the above dimensions of FSLSM model. The ILS instrument has been empirically validated for its reliability and for its construct validity [7].

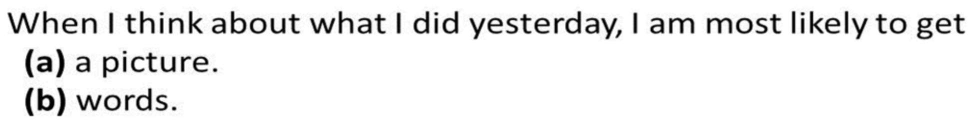
2.1.2. Index of learning styles (ILS)

The ILS is a 44-question instrument (11 questions for each of the four dimensions) intended to measure the learning style preference on each of the four dimensions of the FSLSM model. This instrument was initially created in 1994, and has been revised as a result of rigorous empirical validation. The latest ILS instrument is accessible online at no cost. 'Appendix.ILS Questionnaire' provides a list of 44 questions used in the ILS instrument.

An individual, who wants to know their learning style preference, can use this online link to record their responses on all the 44 questions. Then, the results are immediately reported on the same link that shows the individual's scores on all the four dimensions and how to correctly interpret the results.

Each learning style dimension has 11 associated questions that require an individual to select one of the two choices, with each choice corresponding to one or the other category of the dimension (e.g., active or reflective). For example, for *Visual/Verbal* dimension, of all the 11 questions that has to be answered, each question has an answer that supports either visual or verbal category. An example of the question belonging to Visual/Verbal dimension is shown in

Figure 2. From Figure 2, it could be readily understood that if a subject chooses option ‘a’ then one point will go towards visual learning otherwise one point will go towards verbal learning. Based on the responses on all the 11 questions for each LS dimension, the final LS score report is shown in Figure 3 and is explain follows.



When I think about what I did yesterday, I am most likely to get
(a) a picture.
(b) words.

Figure 2. An example question for the Visual/Verbal dimension of FSLSM model

As shown in Figure 3, each LS dimension is calculated on a scale of 1 to 11 (which is count of the number of responses belonging to each category of the LS dimension). The ‘X’ on the numbers denotes the final score that a person has received in different categories on ILA questionnaire. That is, a score of 9 on the “Visual” category of the Visual/Verbal LS dimension means that out of 11 answers, 10 were in the support of Visual learning preference and 1 was in support of Verbal learning preference. Therefore, 10 – 1 equates to a Visual score of 9. This result shows that this particular individual prefers more visual information as compared to the verbal information. Similar pattern of scales is followed by the other categories (i.e. Sensitive-Intuitive, Visual-Verbal and Sequential-Global) as shown in Figure 3. Therefore, every subject has some *actual score* (varying from 0 - 11) for each LS dimension.

Additionally, an individual with scores of 1 or 3 are considered to be balanced on the two categories of a LS dimension, because they have almost same number of responses in both the categories (e.g., 4-3 =1 or e.g., 7-4 = 3).

```
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 3. An example score report

2.2. Use of Learning Style Models in Software Engineering

While FSLSM has been validated [12, 13] and, used widely for capturing human learning preferences mostly in academia [7, 8, 9, 15], it has not been investigated in the context of software development (to the best of our knowledge).

The concept of matching student-teacher LS's have paid dividends in improving student learning. We believe that extending the use of LS preferences to software engineering context can be very effective in improving the quality of software products. Software engineers (similar to students in a class) differ in the way they "perceive" and "process" information that is documented in software artifacts (e.g., during the early software documents which involves collaboration among technical and non-technical stakeholders). This is especially true during the "requirement stage" of the software development life-cycle, which produces a list of NL requirements that are documented by requirement engineers (who may have different LS's). To produce these requirements, the requirement engineers talk to their customers (who are non-technical end users), in order to elicit and clarify the actual needs.

Evidence suggests that communication errors often arise due to the inability of the end-users to understand the technical jargon used by the developers during the requirements elicitation process which causes ambiguities (i.e., multiple interpretations of same requirements) and incompleteness in the requirement document. Additionally, the requirements document after being developed is passed on to the group of inspectors (either from within or outside the organization) in order to detect defects that were made during the development. These software inspectors (who are different from the requirement developers) may also differ in the way they "perceive" and "process" information. An acute mismatch between the developers and inspectors LS can inhibit the defect detection effectiveness of individual inspectors.

Furthermore, evidence into software inspections have also suggested that reviewing the software artifact from same point of view leads to detection of overlap of faults found by different reviewers, causing a decrease in the overall team effectiveness. Therefore, using the LS preferences and strengths of individual inspectors can be used to reduce the overlap of faults among different inspectors and increase the team's defect detection effectiveness.

The concept of using LS to form more effective software inspection teams is not just limited to the requirements phase, but rather applicable to the inspection of software deliverables produced at the other stages of the software development lifecycle (e.g., code, test plans etc.).

3. RESEARCH APPROACH

To classify the inspectors based on their LS's and to form the inspection team based on the learning styles, we introduced two most important statistical data classification techniques, *Cluster Analysis* (CA) and *Discriminant Analysis* (DA).

CA was used to classify individuals into different groups (clusters) in a way that the individuals belonging to the same cluster are more closely related to one other (i.e., they have similar LS preferences) than the individuals that are assigned to other groups. However there is a variation (dissimilarity) between each individual LS's within the same cluster. In other words CA attempts to form clusters that include individuals with most similar LS preferences, but there will be a variation (dissimilarities) between each individual LS preference within the same cluster. DA helps to further categorize the individuals within each cluster. In sub sections 3.2 and 3.3, the CA and DA were further explained in detail with examples.

As mentioned in Section 2, FSLSM classifies individual's LS into one or the other category in each of the four LS dimensions (sensing/intuitive, visual/verbal, active/reflective; and sequential/global). Within each LS dimension, the relationship between two categories is negatively correlated. That is, as one category increases, the other decreases, and vice versa. Using *active-reflective* dimension as an example, an individual having more number of responses belonging to the active category equates to a lesser preference to the reflective LS. Therefore, the FSLSM questionnaire gives correlated data as a result. This correlation (dependency) within the categories has a negative effect on the statistical classification techniques used by DA.

To transform correlated LS's into uncorrelated LS's, we utilized the Principal Component Analysis (PCA) technique. In this research, PCA was applied to transform the correlated LS variables into a set of uncorrelated variables, entitled *Principal Components* (PC) [16, 17]. More details about PCA and PC's are explained in Subsection 3.1.

In our research approach, we developed a conceptual framework (for inspection team formation) to form inspection teams based on LS with an underlying goal of improving the inspection team effectiveness. This framework uses CA of principal components (PC) to segment the subjects into clusters of similar LS preferences [16, 18, 19]. Next, the DA method was used to analyze the further similarity/dissimilarity information between each individuals LS within a cluster that are not covered by CA [20, 21].

To summarize, this framework employs three modes of statistical analysis:

a) PCA – to create PC's based on LS data; which eliminates any correlation of data;
b) CA – that uses PC's to create clusters that include individuals with similar LS preferences;
c) DA – that uses the results from a) and b) to evaluate the classification of CA, and to determine, how the individuals LS preferences are dissimilar within that cluster.

This result is then used by the software tool to analyze the defect detection effectiveness of the teams based on similar and dissimilar LS's for different inspection team sizes. The working and the design of the tool is presented in Section 4. This section details the principles and detailed description of the PCA, CA and DA statistical techniques.

3.1. Principal Component Analysis (PCA)

PCA is a multivariate technique that is used to convert a set of observations of possibly correlated variables into set of values of uncorrelated variables called principal components (PC) [16].

As mentioned in Section 2, FSLSM classifies individual's LS into one or the other category in each of the four LS dimensions (sensing/intuitive, visual/verbal, active/reflective; and sequential/global). The relationship between two categories of each dimension is negatively correlated. That is, as one category increases, the other decreases, and vice versa. PCA is used in this research to gain a better understanding of 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 questionnaire output) into a new set of uncorrelated variables called principal components (PC) [16, 17].

For each individual, 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) [17]. Each PC accounts for certain variance between the categories in each dimension; and between the dimensions. The PCA will try to account maximum possible variance with the first PC that exists in LS preferences of individual subjects. The second PC will try to account maximum possible variance that could not be explained by first PC and so on. However, it takes all possible number of PC's to explain 100% variance of original data (sometimes the 100% variance may cover with less than total possible number of PC's [22]. The end result of PCA (PC's) is always listed with their respective decreasing variance. Each PC is independent to other PC's and each PC reveals different properties of the original data. So, the total variation in the original data can be covered with all the possible number of PC's. The total variation covered in the original data increases as the numbers of PCs are increasing.

The scree plot (Figure 4) is a simple line segment plot that shows the total variance in the original data which is covered by respective PC's. The marks on the solid line shows the variance that is covered by respective PC. The marks on the dotted line shows the cumulative variance that is covered by respective PC.

As shown in Figure 4, the first PC explains the most variance (~38%) and second explains approximately 22% of variance which not covered in first PC. That is, using both first and second PC's together cover around 60% of variance (cumulative). As the number of PC's increases, the amount of variance described in original data decreases (indicated with solid line). For example, after the last PC (13th one), the variance that is not covered in the original data is close to zero.

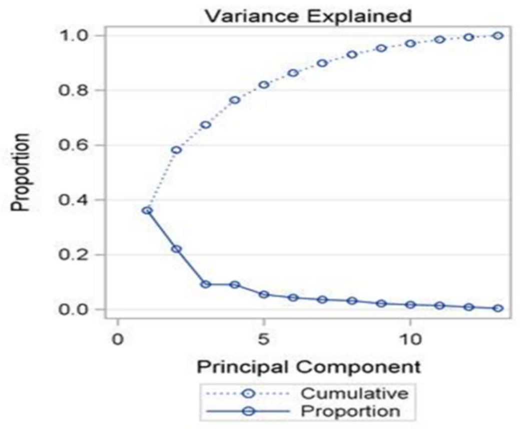


Figure 4. Scree plot of PC's [35]

3.2. Cluster Analysis (CA)

Cluster analysis (CA) is another multivariate technique, which form groups (also called clusters) with the objects that are relatively homogeneous within themselves and heterogeneous between each other [16, 23]. The main goal of performing CA in our research is to form 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 [24, 25, 26, 27]. The resulting clusters could be understood more from the Figure 5, which shows three different clusters denoted by different colors. The square blocks represent individual subjects, which are then grouped into three different clusters (shown by three different colors). The individuals belonging to each cluster (represented by the blocks of same color) are more similar in their characteristics as compared to other clusters.

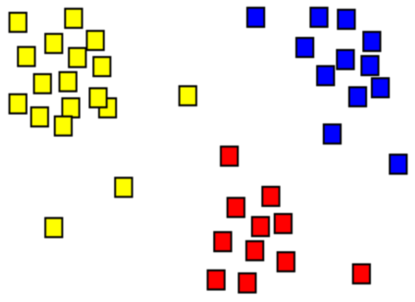


Figure 5. Cluster analysis [36]

There are broadly two types of clustering techniques - Hierarchical and Non-Hierarchical. Hierarchical clustering is a method of CA which builds a hierarchy of clusters and is further classified as two types, a) Agglomerative, a bottom up approach where each observation starts in its own cluster, and pairs of clusters are merged according to their similarities. As the similarity decreases all the sub groups are fused into single cluster. b) Divisive, a top down approach where all observations start in one cluster, and further divided into dissimilar groups; the process continues until there are as many subgroups as observations [28].

On the other hand, k-means clustering is a method of non-hierarchical CA which partitions observations into *k* clusters. The working of k-means algorithm is illustrated in Figure 6 and explained as follows.

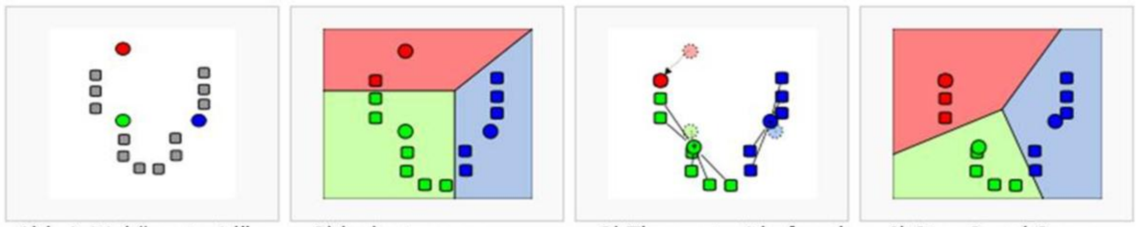

1) k initial "centroid" are randomly generated within the given observations (shown in red, blue, green)
2) k clusters are created by associating every observation with its nearest centroid. The partitions here differentiated by colors
3) The centroid of each k cluster becomes the new mean of cluster.
4) Step 2 and 3 are repeated until convergence has been reached

Figure 6. Demonstration of K-Means algorithm [37]

In *k*-means, the user inputs desired number of clusters (*k*), then the k-means algorithm chooses *k* initial centroids at random and each subject will be assigned to nearest centroid, then the centroids are reset to the average of their assigned cluster. All individuals are then reassigned to the new closest centroid and the process is repeated until there are no more changes in cluster [29].

K-means clustering was implemented with SAS using PROC FASTCLUS that uses a method that calls nearest centroid sorting [23]. The PROC FASTCLUS was directly inspired by k-means clustering algorithm. As illustrated in Figure 6, the k-means algorithm works in four basic steps. 1) Initially it selects k random set of observations as centroids (mean) of expected k clusters. 2) Generates *k* clusters by associating every observation to its nearest centroid 3) Generates new centroid by calculating mean of each observation of the k clusters. 4) Repeat step 2 and 3 until the convergence has been reached.

The PROC FASTCLUS procedure differs from other nearest centroid sorting methods in the way the initial centroids are selected. The initial centroids are the centroids which are randomly selected by *k*-means algorithm at first step (Figure 6). In our study, CA was used to segment the students into clusters of similar LS. These clusters 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 similar LS group.

3.3. Discriminant Analysis (DA)

DA is a statistical method in which LS variations are partitioned into a "*between group*" and a "*within-group*" component. This result of the DA is used to maximize the LS variations across different clusters, and minimize the LS variations within each cluster [16, 30, 31, 35].

As discussed in the previous section, the k-means algorithm (CA) will attempt to form clusters that contain individuals with similar LS preferences based on centroid (Figure 6). So, 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 LSs within the same cluster and with respect to the individuals in other clusters.

To accomplish this goal, DA provides GM values for each individual for all clusters and the maximum GM value indicates that the individual has most similar LS when compared to the particular cluster. So, an individual is classified into a cluster that has the maximum GM value. The dissimilarities between each individual LSs within the same cluster could be evaluated by using the difference between the GM values of individuals. Therefore, DA delivers Group Membership (GM) values for each individual with respect to each cluster.

This could be better explained with example in Table 1. In Table 1, the individual 1 (ID =1) has GM value of 0.98 with respect to Cluster 1 and has GM value of 0.02 with respect to cluster 2. Therefore the individual 1 has more similar LS preferences with Cluster 1. So, individual 1 has been classified into cluster 1. Next, the individual 3 (ID =3) has GM value of 0.95 with respect to cluster 1 and GM value of 0.05 with respect to cluster 2. Therefore the individual 3 has more similar LS preferences with cluster 1. So, Individual 3 has been classified into cluster 1.

Also, DA is used to assess the adequacy of CA result (clusters information of each individual that is generated by CA) [33]. According to theories of DA, there is very little chance of DA classification to be different from the CA classification. In rare cases, when the DA classification is different from CA classification, then the CA classification is replaced with DA classification as evidence from literature which suggests that the DA classification can help remove the misclassification that often plagues the CA output [30, 31, 34].

To assess the adequacy of CA result, DA uses Group Membership (GM) value of each individual and cluster combination (generated by CA). Then, DA considers the highest GM value of each individual and cluster combination; and assigns the individual into a cluster for which that individual has maximum GM value. In that way, DA classifies all the individuals into known clusters (that were generated by CA). This scenario is illustrated in Table 1 and explained below.

As shown in Table 1, the 5 students were classified into two different clusters using the CA technique. The second column (cluster ID) denotes the clusters that the subjects were classified into, based on their LS preferences. That is, subjects 1 and 3 had similar LS preferences and belong to cluster ID 1; whereas the remaining subjects were similar in their LS's and were grouped into cluster ID 2.

Table 1. DA output for 2 clusters

Student ID	Cluster ID	Cluster1 GM value	Cluster2 GM value	Maximum of Cluster 1&2
1	1	0.98	0.02	0.98
2	2	0.30	0.70	0.70
3	1	0.95	0.05	0.95
4	2	0.01	0.99	0.99
5	2	0.01	0.99	0.99

Then, DA generates GM values for each individual and cluster combination (as shown in columns 3 and 4 of Table 1). Then, based on the highest GM value for each individual for all the clusters, DA classifies an individual into a particular cluster. As shown in Table 1, DA classifies an individual 1 into cluster 1 as the individual 1 has the highest GM value 0.98 with cluster 1 when compared to cluster 2 GM value 0.02. This process is repeated until each individual has been classified into different clusters, and the resulting classification is compared with the CA result. Therefore, the GM values helps us to determine an individual with most dissimilar LS (of all the individuals within that cluster) when compared to the individuals belonging to other clusters. For example, in Table 1, individuals 1 and 3 belonging to clusterID1 have dissimilar LS preferences with respect to the individuals 2, 4, and 5 belonging to cluster ID 2. Further, the individual 1 is more dissimilar (when compared with individual 3) to the individuals belonging to the other cluster. Because the individual 1 GM value (0.98) is greater than individual 3 GM value(0.95).

In this research, GM was used to sort the teams ranging from most dissimilar LS to teams with most similar LS preferences and strengths. This process of extracting software inspection teams with varying levels of LS preferences was automated using the software tool, and is described in Section 4.

4. RESEARCH DESIGN –TEAM FORMATION

In this section, we will discuss, the process of using together the three statistical analyses methods (i.e., PCA, CA, DA) in order to form inspection teams based on the individuals LS preferences. The underlying criteria for selecting inspection teams were to select individuals that would help uncover more number of defects present in the software artifact (e.g., SRS document). As mentioned in Sections 1 and 2 (guided by the literature review findings), if individual inspectors have diverse set of LS preferences or strengths (as opposed to similar LS's), they are more likely to detect different types of defects present in the artifact under inspection. This would result in the detection of larger number total defects, and would reduce the overlap of defects found by multiple inspectors, resulting in an increase in software inspection effectiveness.

Therefore, this research design is threefold;

1) Form all the possible teams (using all possible non-repeated combinations of individual inspectors);

2) Sort the teams based on LS's of individual inspectors- For each inspection team size, this step requires formation of teams, ranging from individuals with most dissimilar LS's to most similar LS's (as the two extremes);

3) Count of the unique defects and common defects - The common defects can be categorized as the defects identified by more than one team member and the unique defects are the total number (non-repeated) of defects that are identified by all team members that were unique.

In order to form software inspection teams based on their similar and dissimilar LS's, We developed a tool called Learning Style Based Inspection System (LSBIS) and is described in

Section 5. To provide an overview, LSBIS is a Silverlight application which uses SQL Database to store all the inspectors LS data. To incorporate the PCA, CA, and DA statistical classification techniques into our LSBIS, we used SAS which runs on server side and also can communicate with our front end Silverlight application. There are two phases in the tool of which the first phase aims to store the individual (LS data and defect data) data in database, whereas the second phase aims to form teams based on team members LS data and maps it to the unique and common defect count for those teams, for varying inspection team size.

In the following sections, we describe these phases that comprise the data processing, and inspection team forming processes. The details of the tool are discussed later.

4.1. Team Formation

The team formation process with LSBIS includes the below steps in the listed order :

1. *Form all possible teams*: Find all possible combinations of teams of inspectors for a given team size;
2. *Transform LS into PC's*: The LS variables are transformed into PC's using PCA; this will eliminate any correlation in the data.
3. *Form clusters*: PC's are then used by CA that classifies individuals into clusters;
4. *Find out the Group membership*: DA generates GM value for each individuals to assess the validity of the CA classification;
5. Categorize the teams *based on the LS's of individuals:* The result from above steps are used to form teams based on the LS similarity (and dissimilarity) of individual members

To demonstrate this scenario, Table 2 shows sample LS data from 11 inspectors. These values were taken as input to the next step in order to form teams in terms of the LS of individuals ranging from most dissimilar to similar. In the remaining sub-sections, we provide a summary of each step leading to the team formation to analyze the teams that can uncover more defects during an inspection of software artifact.

Table 2. LS data for each individual

ID	Learning Style Classification							
	Active	Reflective	Sensitive	Intuitive	Visual	Verbal	Sequential	Global
1	5	6	8	3	9	2	5	6
2	6	5	2	9	7	4	6	5
3	6	5	10	1	11	0	7	4
4	6	5	6	5	6	5	7	4
5	10	1	5	6	11	0	7	4
6	4	7	3	8	2	9	3	8
7	8	3	0	11	4	7	4	7
8	4	7	10	1	10	1	7	4
9	9	2	4	7	8	3	4	7
10	8	3	5	6	10	1	8	3
11	11	0	9	2	9	2	10	1

4.1.1. Possible teams

The first step is to find all possible combinations of teams for a given team size. To find all possible combinations of teams we used below combinations formula which gives all possible groups without repetition. Given ‘n’ individual inspectors, and an inspection team size of ‘r’, we obtain all possible combinations (teams) using the equation

$$\text{Number of combinations} = \frac{n!}{(n\text{-}r)!r!}$$

For example, the total combinations for all the inspection teams of 2 inspectors from a pool of 11 inspectors, is 55 combinations. All the 55 resulting combinations are shown in Table 3.

Table 3. All possible combinations for team size 2

Group	Group Members		Group	Group Members		Group	Group Members	
1	1	2	19	2	11	37	5	8
2	1	3	20	3	4	38	5	9
3	1	4	21	3	5	39	5	10
4	1	5	22	3	6	40	5	11
5	1	6	23	3	7	41	6	7
6	1	7	24	3	8	42	6	8
7	1	8	25	3	9	43	6	9
8	1	9	26	3	10	44	6	10
9	1	10	27	3	11	45	6	11
10	1	11	28	4	5	46	7	8
11	2	3	29	4	6	47	7	9
12	2	4	30	4	7	48	7	10
13	2	5	31	4	8	49	7	11
14	2	6	32	4	9	50	8	9
15	2	7	33	4	10	51	8	10
16	2	8	34	4	11	52	8	11
17	2	9	35	5	6	53	9	10
18	2	10	36	5	7	54	9	11
						55	10	11

4.1.2. Transform LS into PC's

Next step is to transform the LS variables of individuals (which give us the correlated data) into uncorrelated variables using PCA. Before the individuals can be classified into different clusters, we needed to normalize (using PCA) our variables (original LS data) because the variables with variances tend to have effect on the resulting clusters. As variance increase with variables, its effect on the resulting clusters increases. SAS procedure PROC PRINCOMP was used to compute uncorrelated PC's.

In our research, the original LS data has eight variables (corresponding to each of the eight LS categories) and was transformed into possible number of (less than or equal to 8) PC's. This was implemented in SAS by using PROC PRINCOMP procedure. We use FSLSM questionnaire result as input to this procedure and received PC's as output.

Following the same example of an inspection team size of 2 inspectors, the outcome of PCA is shown in Table 4. As shown in Table 4, we received only four PC's, that means we accounted the 100% variance in data with the first four PC's only (you can see in Table 4, there are only 4 PC for each subject). The number of PC's varies from different sizes and different input data.

Table 4. PCs of individuals after transform LS data into PCs

Student	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
1	0.03	-1.67	0.72	0.28	0	0	0	0
2	-1.28	0.31	-0.15	-1.10	0	0	0	0
3	1.98	-1.48	0.45	0.17	0	0	0	0
4	-0.29	-0.45	-1.13	-0.05	0	0	0	0
5	1.54	1.63	0.89	-0.19	0	0	0	0
7	-3.98	-1.04	-0.62	0.46	0	0	0	0
8	-3.04	1.66	-0.04	0.01	0	0	0	0
9	1.39	-2.46	-0.02	-0.18	0	0	0	0
10	-0.89	1.17	1.25	0.69	0	0	0	0
11	1.34	0.75	-0.05	-0.84	0	0	0	0
12	3.20	1.59	-1.32	0.74	0	0	0	0

4.1.3. Form clusters based on similarity

Next step is to form clusters based on students LS similarity. The uncorrelated data (outcome of PCA) was used as input for CA which classifies individuals into clusters. The number of clusters is equal to the given size of team.

CA takes PC's as input and provides the cluster numbers for each subject. K-means clustering algorithm (that was used in this research) requires two inputs; a) the required number of clusters and b) all the PC's.

This was implemented in SAS using PROC FASTCLUS, which is designed to find best possible clusters for the given data set. PROC FASTCLUS also designed to provide the exact number of clusters. Observations that are very close to each other are usually assigned to the same cluster, while observations that are far apart are in different clusters. In our research we find the exact number of clusters which equals to inspection team size. Such a way we can find most dissimilar by selecting one team member from each cluster.

Table 5 shows the output of cluster (team size of 2) that would be formed by the CA in our tool. In Table 5, first row shows individuals ID number and the second row shows the cluster number for that respective subject. In this example, for required team size 2, it forms two clusters: subjects with IDs 3, 5, 9, 11, 12 are classified into cluster 1 and rest of the students is classified into cluster 2.

Table 5. CA output

Student	1	2	3	4	5	6	7	8	9	10	11
Cluster No	2	2	1	2	1	2	2	1	2	1	1

4.1.4. Calculate group membership for each inspector using DA

Next the DA was conducted on PC's with known CA result, to predict group membership (GM) based on a linear combination of the variables. In this study, DA was used to determine the GM of each individual. The outcome of DA (GM) for each individual helps us to analyze each individual LS preference relative to other individuals within his/her cluster as well as relative to the individuals belonging to other clusters.

In our research, a SAS procedure named PROC DISCRIM was used to conduct discriminant analysis. After CA, the tool performs DA to find GM values for each subject in each cluster. The maximum GM value indicates that the subject is most dissimilar (when compared to his/her other members in their belonging cluster) to the members of other clusters.

For Example, as shown in Table 6, the subject 2 has the probability of 0.88 to belong in cluster 2 which is the higher than its probability to belong in cluster 1 (0.12). Hence, he/she is placed in cluster 2. Furthermore, as shown in Table 6, subject 7 and 8 has the highest probability (1.00) among all the other subjects belonging to cluster 2, which suggests that both subjects 7 and 8 has most dissimilar LS preference compared to the subjects belonging to cluster 1.

Table 6. DA output for 2 clusters

Subject ID	Cluster ID	Cluster1 GM value	Cluster2 GM value	Maximum of Cluster 1&2
1	2	0.35	0.65	0.65
2	2	0.12	0.88	0.88
3	1	0.97	0.03	0.97
4	2	0.30	0.70	0.70
5	1	0.96	0.04	0.96
7	2	0.00	1.00	1.00
8	2	0.00	1.00	1.00
9	1	0.94	0.06	0.94
10	2	0.05	0.95	0.95
11	1	0.96	0.04	0.96
12	1	1.00	0.00	1.00

4.1.5. Team categorization process

From the above steps, we produced all the possible teams (Section 4.1.1), derived the clusters information (Section 4.1.3), and the group membership (Section 4.1.4) of each individual. This section explains the process of incorporating all the above results to sort the teams based on the LS similarity of individual members.

Using the three values (i.e., non-repetitive possible teams, cluster number and GM value for each individual), we sorted the inspection teams ranging from most dissimilar to most similar.

The total team sorting process achieved in three simple steps. As a first step, we analyze the individual clusters and the GM values of each individual for all possible teams (shown in Table 3). The resulting data is shown in Table 7.

From CA, we know each individual's dissimilarity across different clusters and their similarity within the cluster. So, a team formed with the members, where no two members belong to same cluster is described as most dissimilar team. Also, a team formed with the members from single cluster (or subset of clusters) are described as more similar groups. So, we sorted the teams ranging from most dissimilar to similar as explained in remaining two steps.

As second step in sorting team's process, we counted the total number of clusters for each inspection team size, and sorted in descending order. The no of clusters for each of the 55 possible teams is showed in "# of clusters" column in Table 7.

Also, as shown in Table 7, multiple teams contained members that belonged to different clusters (i.e., had dissimilar LS's). To understand the relative dissimilarity in their LS preferences of such teams, we used GM values to help sort the teams in descending order of their LS dissimilarities. The GM values for each inspection team member (i.e., column 4 for inspection member 1 and column 7 for inspection member 2 as shown in Table 7) were added to produce "Total GM" values to sort the teams with decreasing level of dissimilarity in the LS's of inspection member 1 and member 2.

The following observations summarize the details presented in Table 7.

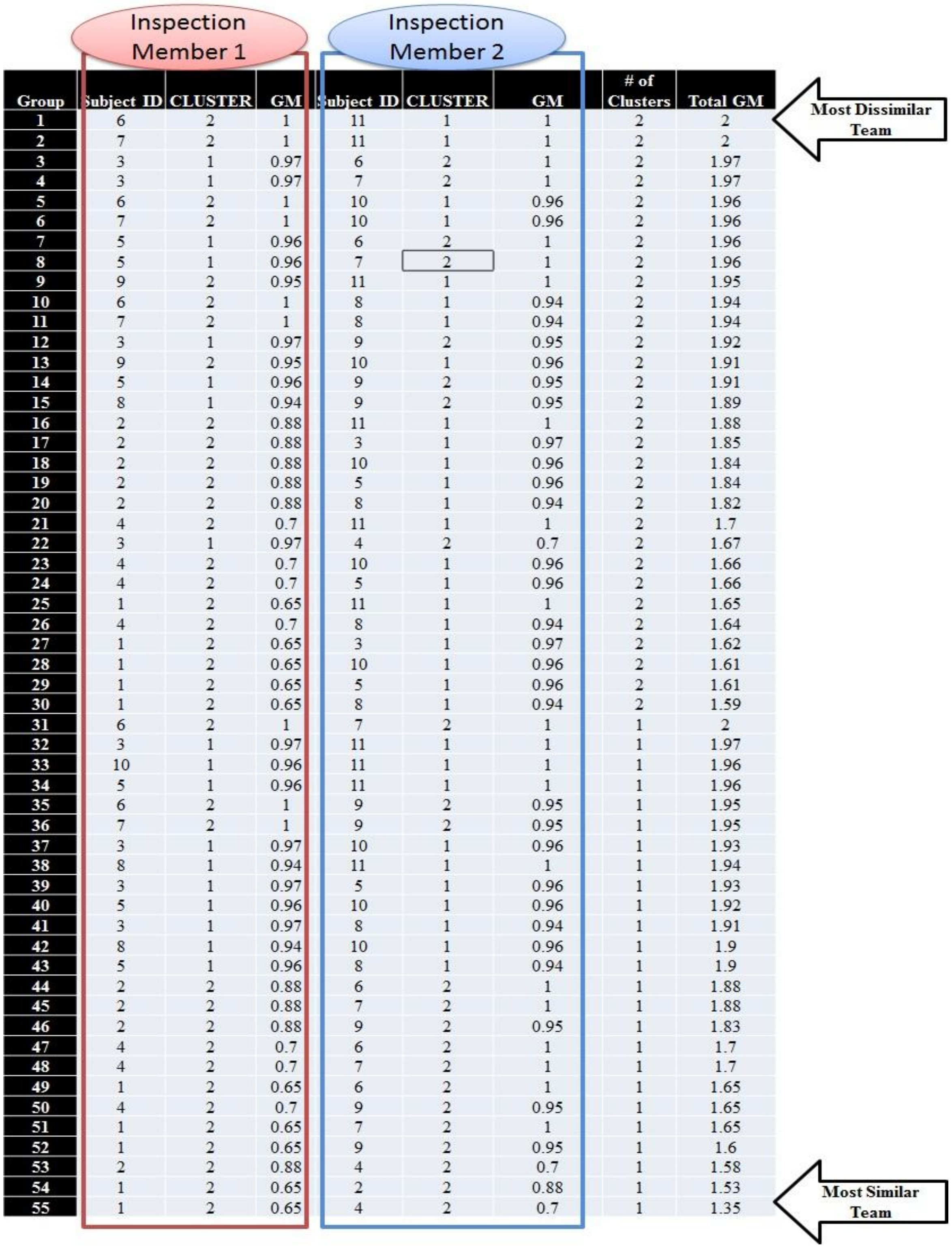
1. The most dissimilar team (of size 2) contains subjects 7 and 12; where each inspector belonging to a different cluster.

    a. As we move down the Table 7, the level of dissimilarity decreases (as evidenced by the decreasing value to Total GM) until we get to Group (Team)# 30 which is the least dissimilar team containing individual inspectors belonging to different clusters.

2. Moving further, teams 31 to team 55 represent teams that contain individuals belonging to same clusters – cluster 1 or cluster 2.
    a. Furthermore, team# 55 is the most similar team (of all the 55 teams) since it has least “Total GM” values for all the teams containing individuals belonging to same cluster.

This same analysis can be performed for any inspection team size. The tool described in the next section automates this process of sorting teams. The tool also provided additional features for researchers to analyze the defect detection effectiveness of teams with dissimilar and similar LS of individual members (for varying inspection team sizes).

Table 7. Sorting team from most dissimilar to most similar in terms of LS of individual


Inspection Member 1
Inspection Member 2
Most Dissimilar Team
Most Similar Team

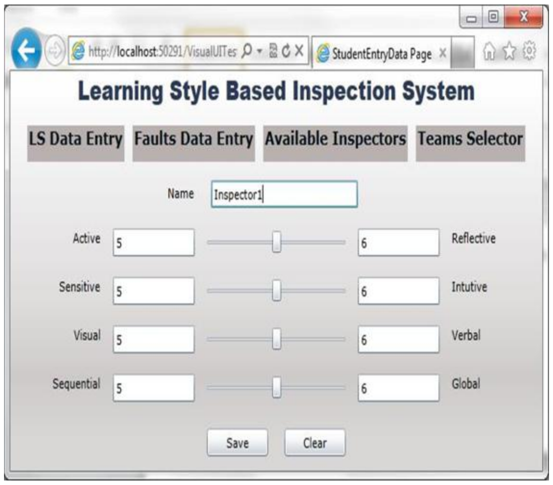
Group	Subject ID	CLUSTER	GM	Subject ID	CLUSTER	GM	# of Clusters	Total GM
1	6	2	1	11	1	1	2	2
2	7	2	1	11	1	1	2	2
3	3	1	0.97	6	2	1	2	1.97
4	3	1	0.97	7	2	1	2	1.97
5	6	2	1	10	1	0.96	2	1.96
6	7	2	1	10	1	0.96	2	1.96
7	5	1	0.96	6	2	1	2	1.96
8	5	1	0.96	7	2	1	2	1.96
9	9	2	0.95	11	1	1	2	1.95
10	6	2	1	8	1	0.94	2	1.94
11	7	2	1	8	1	0.94	2	1.94
12	3	1	0.97	9	2	0.95	2	1.92
13	9	2	0.95	10	1	0.96	2	1.91
14	5	1	0.96	9	2	0.95	2	1.91
15	8	1	0.94	9	2	0.95	2	1.89
16	2	2	0.88	11	1	1	2	1.88
17	2	2	0.88	3	1	0.97	2	1.85
18	2	2	0.88	10	1	0.96	2	1.84
19	2	2	0.88	5	1	0.96	2	1.84
20	2	2	0.88	8	1	0.94	2	1.82
21	4	2	0.7	11	1	1	2	1.7
22	3	1	0.97	4	2	0.7	2	1.67
23	4	2	0.7	10	1	0.96	2	1.66
24	4	2	0.7	5	1	0.96	2	1.66
25	1	2	0.65	11	1	1	2	1.65
26	4	2	0.7	8	1	0.94	2	1.64
27	1	2	0.65	3	1	0.97	2	1.62
28	1	2	0.65	10	1	0.96	2	1.61
29	1	2	0.65	5	1	0.96	2	1.61
30	1	2	0.65	8	1	0.94	2	1.59
31	6	2	1	7	2	1	1	2
32	3	1	0.97	11	1	1	1	1.97
33	10	1	0.96	11	1	1	1	1.96
34	5	1	0.96	11	1	1	1	1.96
35	6	2	1	9	2	0.95	1	1.95
36	7	2	1	9	2	0.95	1	1.95
37	3	1	0.97	10	1	0.96	1	1.93
38	8	1	0.94	11	1	1	1	1.94
39	3	1	0.97	5	1	0.96	1	1.93
40	5	1	0.96	10	1	0.96	1	1.92
41	3	1	0.97	8	1	0.94	1	1.91
42	8	1	0.94	10	1	0.96	1	1.9
43	5	1	0.96	8	1	0.94	1	1.9
44	2	2	0.88	6	2	1	1	1.88
45	2	2	0.88	7	2	1	1	1.88
46	2	2	0.88	9	2	0.95	1	1.83
47	4	2	0.7	6	2	1	1	1.7
48	4	2	0.7	7	2	1	1	1.7
49	1	2	0.65	6	2	1	1	1.65
50	4	2	0.7	9	2	0.95	1	1.65
51	1	2	0.65	7	2	1	1	1.65
52	1	2	0.65	9	2	0.95	1	1.6
53	2	2	0.88	4	2	0.7	1	1.58
54	1	2	0.65	2	2	0.88	1	1.53
55	1	2	0.65	4	2	0.7	1	1.35

5. APPLICATION OF RESEARCH TOOL

Having explained the team formation process in the earlier sections, this chapter presents an automated tool that automates the formation of team development based on varying LS preferences, and maps it to the defect data of individual inspectors belonging to an inspection team.

5.1. Adding Learning Style Data

Researchers or users can upload an each individual inspector LS data into tool database. Researchers will need to enter all eight learning styles data along with inspector's ID. For convenience, both scale and textboxes are provided, so that users can use either of the one which makes them more comfortable (as shown in Figure 7). Also, the fields in LS data entry page are mandatory; otherwise the user is prompted with an error message box as shown in Figure 8.

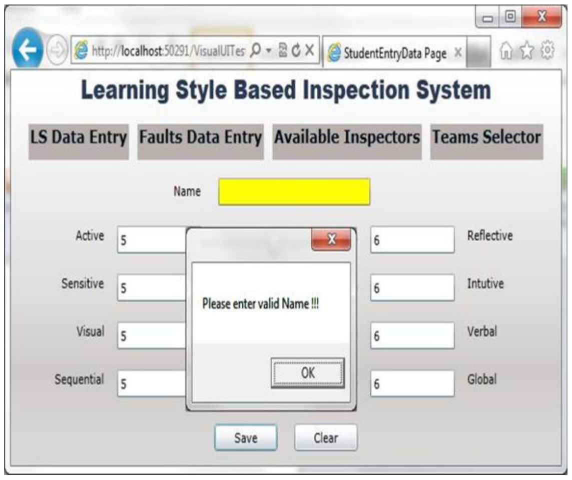


Learning Style Based Inspection System

Name: Inspector1

Active	5	6	Reflective
Sensitive	5	6	Intutive
Visual	5	6	Verbal
Sequential	5	6	Global

Figure 7. LS data entry



Learning Style Based Inspection System

Name:

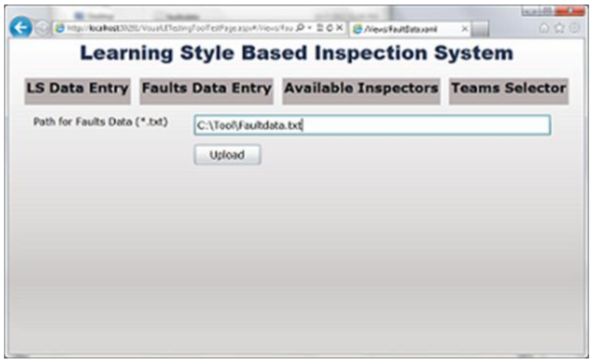
Active	5	6	Reflective
Sensitive	5	6	Intutive
Visual	5	6	Verbal
Sequential	5	6	Global

Please enter valid Name !!!

Figure 8. LS data entry error message

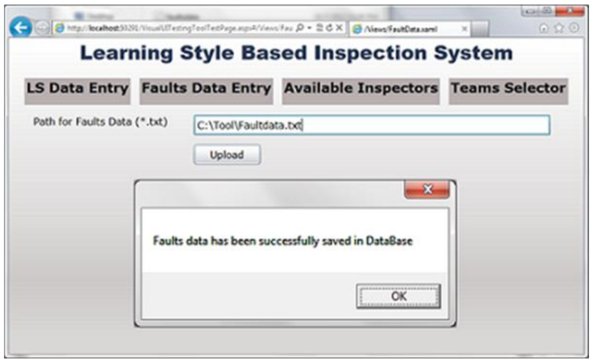
5.2. Uploading Faults Data

To upload inspection results, user would need to enter full path to faultdata file and then click upload button (as shown in Figure 9). Author will get notified the status of upload by a message box as shown in Figure 10. The faultdata file should be a text file with tab spaced values. It reads each row and update a record in database. In each row of faultdata file, the very first attribute should have inspectors identification number and the rest attributes should have the faults found by that inspector. This fault data will be used to evalute the total & redundant number of faults found by group of inspectors. Researchers can fetch the advantage of faults data to study the team performance while uncovering defects in a software artifact.



Path for Faults Data (*.txt) C:\Tool\Faultdata.txt

Figure 9. Inspector data entry



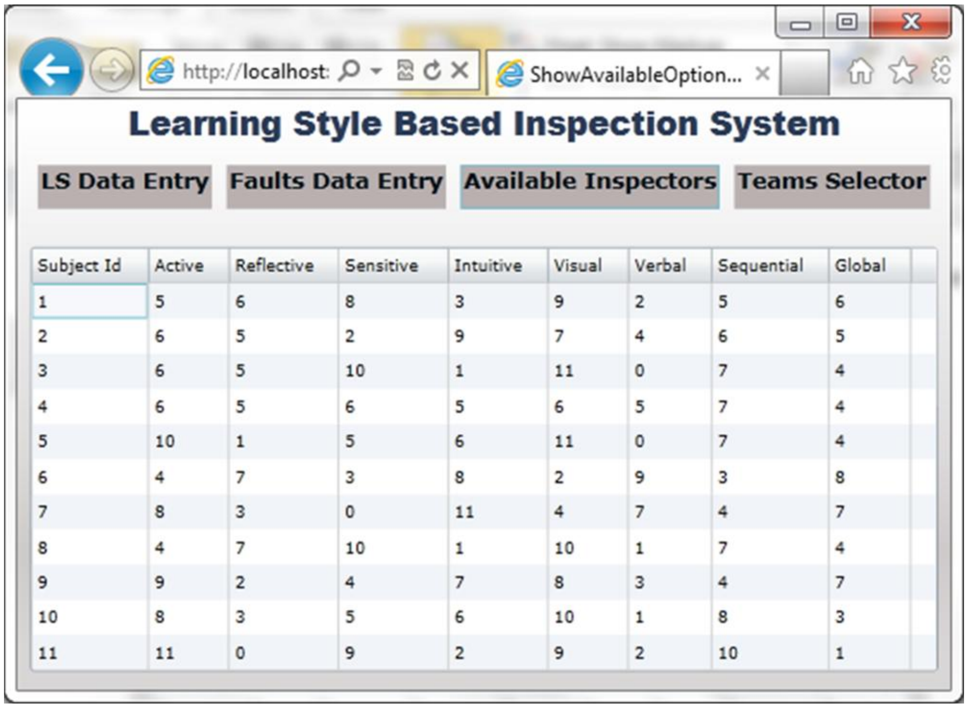
Path for Faults Data (*.txt) C:\Tool\Faultdata.txt

Faults data has been successfully saved in DataBase

Figure 10. Inspection data entry result

5.3. Available Inspectors

Upon the successful upload of the LS data and the fault data for each individual inspector, the tool provides ability for the researcher to see the list of inspectors with their LS data as shown in Figure 11.



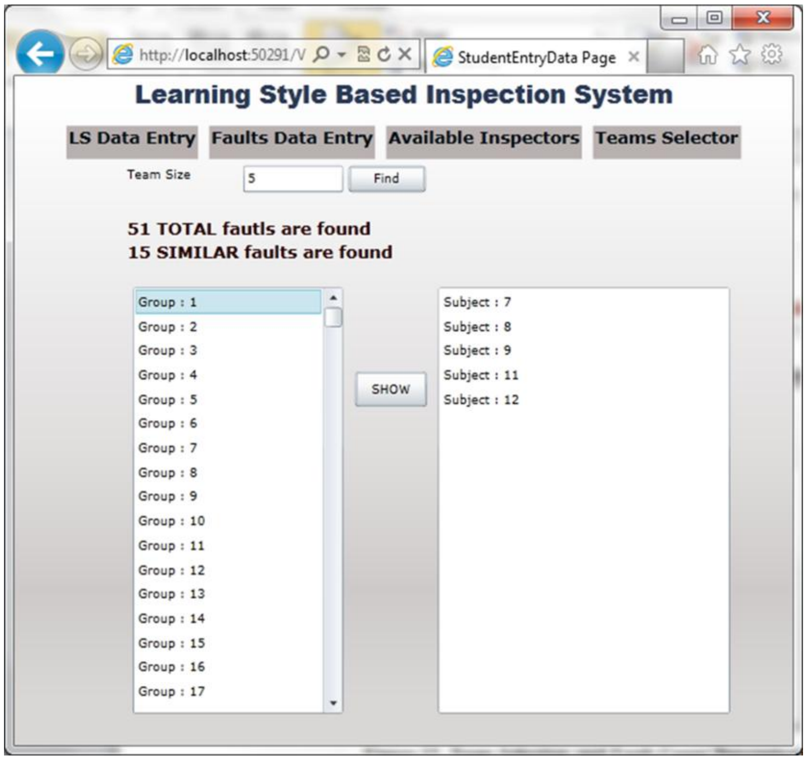
Subject Id	Active	Reflective	Sensitive	Intuitive	Visual	Verbal	Sequential	Global
1	5	6	8	3	9	2	5	6
2	6	5	2	9	7	4	6	5
3	6	5	10	1	11	0	7	4
4	6	5	6	5	6	5	7	4
5	10	1	5	6	11	0	7	4
6	4	7	3	8	2	9	3	8
7	8	3	0	11	4	7	4	7
8	4	7	10	1	10	1	7	4
9	9	2	4	7	8	3	4	7
10	8	3	5	6	10	1	8	3
11	11	0	9	2	9	2	10	1

Figure 11. List of inspectors and their LS preferences

5.4. Team Selector

This feature provides all the possible teams sorted from most dissimilar (in terms of the LS of individuals who make up the team) to the team with most similar LS preferences among individual members. The tool provides an inbuilt functionality that calculates and displays the count of *unique faults*, and the count of *similar faults* found by each team. This is shown in Figure 12 and explained below.

To receive this information, the user will need to input the team size (which should be more than one and less than the total number of available inspectors). Upon entering the required team size, user can then click "*Find*' option to see the list of all the teams. The list box on the left side shows the teams sorted from dissimilar to similar LS preferences. The most dissimilar team is on top of the list. To see the group members and the faults found by respective group, user will need to highlight the group listed in left side list box and click "*SHOW*" button. The result will be presented to the user on the right side in form of a list box that shows: a) the individuals from the highlighted group, b) the number of *unique (Total) faults* detected by the team; and c) the number of *redundant* (similar) faults found by the respective group.



51 TOTAL fautls are found
15 SIMILAR faults are found

Group : 1
Group : 2
Group : 3
Group : 4
Group : 5
Group : 6
Group : 7
Group : 8
Group : 9
Group : 10
Group : 11
Group : 12
Group : 13
Group : 14
Group : 15
Group : 16
Group : 17

Subject : 7
Subject : 8
Subject : 9
Subject : 11
Subject : 12

Figure 12. Team selection and fault count determination

6. RELEVANCE OF YOUR TOOL TO RESEARCHERS AND PRACTITIONERS

The tool will facilitate researchers to study the impact of inspectors LS preferences on the effectiveness and efficiency during the inspection of a software artifact. This tool will especially aid the researchers to help evaluate the promise of LS preferences as a basis for forming teams that would uncover larger number of software defects. To accomplish this research goal, the tool provides the functionality to help sort the teams consisting of individuals inspectors with most dissimilar LS's to teams consisting of individual inspectors with most similar LS's.

Additionally, this tool will allow evaluating the redundant and total number of faults found by various inspection teams (formed based on the LS preferences) to reduce the overhead of manually entering and combining the defect detection effectiveness of the individuals making up an inspection team. So, researchers could fetch the advantage of this tool to study how an inspection team with similar vs. dissimilar LS preferences uncover the defects present in in requirement document.

Also, this tool provides freedom for project managers to select the team with similar or dissimilar learning style preferences depending on the resources available to them or depending on the LS preferences of the developers of the software artifacts being inspected. Furthermore, this tool can be used to form development teams based on the developer's LS preferences that can aid collaboration and improve overall productivity.

7. FUTURE IMPROVEMENTS AND CONCLUSION

The paper presents research that was performed to develop an automated tool that helps researchers and practitioner to investigate the impact of learning style preferences on the effectiveness of requirements inspection. To accomplish that, the research tool enabled grouping inspectors according to their learning preferences (ranging from most similar to most dissimilar for each inspection team size) so that its effect on the number of defects uncovered during an inspection can be analyzed. One key advantage of the tool is the development of broad range of teams based on the individual member's preferences of the learning styles. Dr. Walia's Empirical Software Engineering (ESE) research group at North Dakota State University has already started using this tool to evaluate the effectiveness of teams based on LS preferences. Their research also plans on using this tool to study the impact of the inspector's learning styles on the other activities of software development process (e.g., requirement elicitation, global software development etc.).

This tool is not only limited to inspection of early software documents, but can be also used to form teams during the code review. Further enhancements of this tool will incorporate the FSLSM questionnaire into tool, which stores the result in a database. It can also have separate security profiles for both managers and individual inspectors. The individual inspector profiles will have ability to only save their LS and faults data, whereas manager's profiles can have access to team formation and the team effectiveness. It can also have a functionality to help form teams of inspectors with only specified learning preferences (e.g. only more Active & sensitive) or form teams in order to match the learning style preferences of the developers of the software artifacts.

8. REFERENCES

[1] Nalbant. Serkan, “An Evaluation of the Reinspection Decision Policies for Software Code Inspections”. In Thesis Submitted to The Graduate School of Natural and Applied Sciences of Middle East Technical University, Jan 2005.

[2] M.E. Fagan, “Design and Code Inspections to Reduce Errors in Program Development”. IBM Systems, vol.15, no. 3, pp. 182-211, 1976.

[3] M.E. Fagan, “Advances in Software Inspection”. IEEE Trans. Software Eng., vol. 12, no. 7, pp. 744-751, July 1986.

[4] E.P. Doolan, “Experience with Fagan’s Inspection Method”. Software Practice and Experience, vol. 22, no. 2, pp. 173-182, Feb. 1992.

[5] G.W. Russell, “Experience with Inspection in Ultra Large-Scale Developments”. IEEE Software, vol. 8, no. 1, pp. 25-31, Jan. 1991.

[6] J. Carver, “The Impact of Background and Experience on Software Inspections”. PhD Thesis, Department of Computer Science, University of Maryland College Park, MD, 200.

[7] R.M. Felder, L.K. Silverman, "Learning and Teaching Styles in Engineering Education". Engr. Education, 78(7), 674-681 (1988).

[8] R.M. Felder, E.R. Henriques, "Learning and Teaching Styles in Foreign and Second Language Education". Foreign Language Annals, 28(1), 21-31 (1995).

[9] R.M. Felder, "Reaching the Second Tier: Learning and Teaching Styles in College Science Education". J. College Science Teaching, 23(5), 286-290 (1993).

[10] S. Graf, S. R. Viola, T. Leo, “Representative Characteristics of Felder-Silverman LearningStyles: an Empirical Model”. Proceedings of the IADIS International Conference on Cognition andExploratory Learning in Digital Age (CELDA 2006), Barcelona, Spain, pp. 235-242.

[11] M. K. Smith, “David A. Kolb on experiential learning”. Retrieved January 20, 2012, from: http://www.infed.org/biblio/b-explrn.htm.2001

[12] R.M. Felder, J.E. Spurlin, "Applications, Reliability, and Validity of the Index of Learning Styles". Intl. Journal of Engineering Education, 21(1), 103-112 (2005).

[13] R.M. Felder, “Are Learning Styles Invalid? (Hint: No!)”. On-Course Newsletter, September 27, 2010.

[14] R.M. Felder, B.A. Soloman, “Index of Learning Styles, “http://www.ncsu.edu/felder-public/ILSpage.html”. 2004, accessed February 15, 2006.

[15] H. Hong, Kinshuk, “Adaptation to Student Learning Styles in Web Based Educational Systems”. In L. Cantoni & C. McLoughlin (Eds.), Proceedings of World Conference on Educational multimedia, Hypermedia & Telecommunications (Ed-Media), 2004, pp. 491-496.

[16] T. W. Anderson, “An Introduction to Multivariate Statistical Analysis”. Second Edition, New York: John Wiley & Sons, 1984.

[17] T. Nathan, B. Brian, “Evaluating Principal Component Analysis for Identifying Optimal Bands Using Wetland Hyperspectral Measurements From the Great Lakes, USA”. Remote Sensing, Aug 2009, ISSN 2072-4292

[18] Asa Ben-Hur, Isabelle Guyon, “Detecting Stable Clusters Using Principal Component Analysis”. Humana press, 2003 pp.159-182

[19] Chris Ding, Xiaofeng He, “K-means Clustering via Principal Component Analysis”. Computational Research Dividsion, 2004.

[20] SAS Institute Inc., “SAS/STAT user’s guide, release 6.03 edition”. SAS Institute, Inc, Cary, NC, 1988.

[21] BK. Williams, “Some observations on the use of discriminant analysis in ecology”. Ecology 64:1283-1291, 1983.

[22] I. T. Jolliffe, “Principal Component Analysis, Springer-Verlag”. pp. 487, ISBN 978-0-387-95442-4, 1986.

[23] Keh-Dong. Shiang, “101 Applications of Multivariate Statistical Procedures: PRINCOMP, CLUSTER, DSCRIM in SAS 9.2”.

[24] M. Aldenderfer, R. Blashfield, “Cluster Analysis”. London: Sage, 1984.

[25] M. R. Anderberg, "Cluster Analysis for Applications”. New York: Sage, 1973.

[26] M. Lorr, “Cluster Analysis for social scientists”. San Francisco: Jossey-Bass, 1983.

[27] M. Steinbach, L. Ertoz , V. Kumar, “The Challenges of Clustering High Dimensional Data”. DAAD 19-01-2-0014.

[28] Lin. Lin, “Bayesian Veriable Selection in Clustering and Hierarchial Mixture Modeling”. Department of Statistical Science Duke University, 2012.

[29] D. MacKay, “Chapter 20. An Example Inference Task: Clustering Information Theory, Inference and Learning Algorithms”. Cabrdge University Press, pp.284-292, ISBN 0-

521-64298-1, MR 2012999. [30]	W. Klecka, “Discriminant Analysis”. Beverly Hills: Sage, 1980.

[31] M .M. Tatsuoka, “Multivariate Analysis”. New York: Macmillan, 1988.

[32] T. Jombart, “A tutorial for Discriminant Analysis of Principal Components (DAPC) using adegenet”. pg 1.3-4, 2012.

[33] K. J. Galbraith, J. Lu, “Cluster And Discriminant Analysis on Time-Series as a Research Tool”. 1999.

[34] P. A. Lachenburch, M. Goldstein, “Discriminant Analysis”. Vol. 35, No. 1, 1979.

[35] Scree plot for PC’s, “http://support.sas.com/documentation/cdl/en/statug/63347/HTML /default/viewer .htm #statug_princomp _sect021.htm”

[36] Figure to explain Cluster Analysis, http://en.wikipedia.org/wiki/Cluster_analysis

[37] Demonstration of K-Means Algorithm, http://en.wikipedia.org/wiki/K-means_clustering

[38] B. Boehm, V.R. Basili, "Software Defect Reduction Top 10 List". IEEE Computer, 2001. 34(1): 135-13.

[39] A. Ackerman, L. Buchwald, F. Lewski, “Software Inspections: An Effective Verification Process”. IEEE Software, 1989. 6(3): 31-36.

[40] T. Gilb, D. Graham, “Software Inspection”. Addison-Wesley, 1993.

[41] J. Carver, N. Nagappan, A. Page, "The Impact of Educational Background on the Effectiveness of Requirements Inspections: An Empirical Study". IEEE Transactions on Software Engineering, 2006. 34(6):800-812.

[42] P. Friedman, R. Alley, “Learning/Teaching Styles: Applying the Principles”. Theory into Practice, 23, 1: 77-81, 1984.

[43] J. W. Keefe, “Learning Style: An Overview in J. W. Keefe (ed.), Student Learning Styles: Diagnosing and Prescribing Programs”. NASSP, 1979.

[44] M. Ehrman, R. Oxford, “Adult Language Learning Styles and Strategies in an Intensive Training Setting”. The Modern Language Journal, 74: 311-327, 1990.

[45] D. KoIb, “Experiential Learning: Experience as the Source Of Learning and Development”. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[46] B. McCarthy, “The 4MAT System: Teaching to Learning Styles with Right/Left Mode Techniques”. Barrington, IL: EXCEL, Inc., 1987.

[47] lB.Myers, M.H. McCaulley "Manual: A Guide to the Development and Use of the Myers Briggs Type Indicator". Palo Alto, CA: Consulting Psychologists Press, 1985.

[48] M.Kane, "Cognitive Styles of Thinking and Learning: Part One". Academic Therapy 19(5): 527, 1984.

[49] R.J. Charkins, D.M. OToole, J.N. Wetzel "Linking Teacher and Student Learning Styles with Student Achievement and Attitudes". J Economic Education, Spring 1985: 111-120, 1985.

[50] E. Kanninen, "Learning styles and e-learning". Tampere University of Technology.

APPENDIX. ILS QUESTIONNAIRE

I understand something better after I
(a) try it out.
(b) think it through.

I would rather be considered
(a) realistic.
(b) innovative.

When I think about what I did yesterday, I am most likely to get
(a) a picture.
(b) words.

I tend to
(a) understand details of a subject but may be fuzzy about its overall structure.
(b) understand the overall structure but may be fuzzy about details.

When I am learning something new, it helps me to
(a) talk about it.
(b) think about it.

If I were a teacher, I would rather teach a course
(a) that deals with facts and real life situations.
(b) that deals with ideas and theories.

I prefer to get new information in
(a) pictures, diagrams, graphs, or maps.
(b) written directions or verbal information.

Once I understand
(a) all the parts, I understand the whole thing.
(b) the whole thing, I see how the parts fit.

In a study group working on difficult material, I am more likely to
(a) jump in and contribute ideas.
(b) sit back and listen.

I find it easier
(a) to learn facts.
(b) to learn concepts.

In a book with lots of pictures and charts, I am likely to
(a) look over the pictures and charts carefully.
(b) focus on the written text.

When I solve math problems
(a) I usually work my way to the solutions one step at a time.
(b) I often just see the solutions but then have to struggle to figure out the steps to get to them.

In classes I have taken
(a) I have usually gotten to know many of the students.
(b) I have rarely gotten to know many of the students.

In reading nonfiction, I prefer
(a) something that teaches me new facts or tells me how to do something.
(b) something that gives me new ideas to think about.

I like teachers
(a) who put a lot of diagrams on the board.
(b) who spend a lot of time explaining.

When I'm analyzing a story or a novel
(a) I think of the incidents and try to put them together to figure out the themes.
(b) I just know what the themes are when I finish reading and then, I have to go back and find the incidents that demonstrate them.

When I start a homework problem, I am more likely to
(a) start working on the solution immediately.
(b) try to fully understand the problem first.

I prefer the idea of
(a) certainty.
(b) theory.

I remember best
(a) what I see.
(b) what I hear.

It is more important to me that an instructor
(a) lay out the material in clear sequential steps.
(b) give me an overall picture and relate the material to other subjects.

I prefer to study
(a) in a study group.
(b) alone.

I am more likely to be considered
(a) careful about the details of my work.
(b) creative about how to do my work.

When I get directions to a new place, I prefer
(a) a map.
(b) written instructions.

I learn
(a) at a fairly regular pace. If I study hard, I'll "get it."
(b) in fits and starts. I'll be totally confused and then suddenly it all "clicks."

I would rather first
(a) try things out.
(b) think about how I'm going to do it.

When I am reading for enjoyment, I like writers to
(a) clearly say what they mean.
(b) say things in creative, interesting ways.

When I see a diagram or sketch in class, I am most likely to remember
(a) the picture.
(b) what the instructor said about it.

When considering a body of information, I am more likely to
(a) focus on details and miss the big picture.
(b) try to understand the big picture before getting into the details.

I more easily remember
(a) something I have done.
(b) something I have thought a lot about.

When I have to perform a task, I prefer to
(a) master one way of doing it.
(b) come up with new ways of doing it.

When someone is showing me data, I prefer
(a) charts or graphs.
(b) text summarizing the results.

When writing a paper, I am more likely to
(a) work on (think about or write) the beginning of the paper and progress forward.
(b) work on (think about or write) different parts of the paper and then order them.

When I have to work on a group project, I first want to
(a) have "group brainstorming" where everyone contributes ideas.
(b) brainstorm individually and then come together as a group to compare ideas.

I consider it higher praise to call someone
(a) sensible.
(b) imaginative.

When I meet people at a party, I am more likely to remember
(a) what they looked like.
(b) what they said about themselves.

When I am learning a new subject, I prefer to
(a) stay focused on that subject, learning as much about it as I can.
(b) try to make connections between that subject and related subjects.

I am more likely to be considered
(a) outgoing.
(b) reserved.

I prefer courses that emphasize
(a) concrete material (facts, data).
(b) abstract material (concepts, theories).

For entertainment, I would rather
(a) watch television.
(b) read a book.

Some teachers start their lectures with an outline of what they will cover. Such outlines are
(a) somewhat helpful to me.
(b) very helpful to me.

The idea of doing homework in groups, with one grade for the entire group,
(a) appeals to me.
(b) does not appeal to me.

When I am doing long calculations,
(a) I tend to repeat all my steps and check my work carefully.
(b) I find checking my work tiresome and have to force myself to do it.

I tend to picture places I have been
(a) easily and fairly accurately.
(b) with difficulty and without much detail.

When solving problems in a group, I would be more likely to
(a) think of the steps in the solution process.
(b) think of possible consequences or applications of the solution in a wide range of areas.