An Empirical Analysis of Team Review Approaches for Teaching Quality Software Development

Amna Humayun
Department of Computer Science
National University of Computer and Emerging Sciences
Pakistan
amna.humayun@nu.edu.pk

Wafa Basit
Department of Computer Science
National University of Computer and Emerging Sciences
Pakistan
wafatahir@yahoo.com

Ghulam A Farrukh
Department of Computer Science
University of Agriculture Faisalabad
Pakistan
gafarrukh@gmail.com

Fakhar Lodhi
Department of Computer Science
National University of Computer and Emerging Sciences
Pakistan
fakhar.lodhi@nu.edu.pk

Rabea Aden
Department of Computer Science
National University of Computer and Emerging Sciences
Pakistan
rabea.aden@nu.edu.pk

ABSTRACT

Reviews are an integral part of the software development process. They are one of the key methodologies that undergraduates study in order to develop quality software. Despite their importance, reviews are rarely used in software engineering projects at the baccalaureate level. This paper demonstrates results from a study conducted on students at baccalaureate level enrolled in a one-semester software engineering course at the National University of Computer and Emerging Sciences – Foundation for Advancement of Science and Technology (NUCES-FAST) in Pakistan. The objectives of the study are: to determine how the various team review techniques help to educate students about the importance of the review process and find which technique is more suitable for teaching reviews to undergraduates. Two variations on team review are proposed: Similar Domain Review (SDR) and Cross-Domain Review (CDR) without author. The paper presents a comparison of the proposed and existing team review techniques and measures their effectiveness in terms of defect detection. The results show that the proposed variation SDR is more effective in defect detection than CDR (with/without author). Another interesting result is that the proposed CDR-without author is better than CDR with author (the existing team review approach). Also, early defect detection enabled students to incorporate changes and improve the software quality.

1. INTRODUCTION

Reviews are one of the most important techniques of the software development process. They are used for validation and defect detection in the different phases of the software development life cycle [16]. Software reviews promise high quality software by early elimination of software defects [15]. They also help in reducing the cost and time involved in developing software [7].

ICSE '10, May 2-8, 2010, Cape Town, South Africa

Reviews play one of the major roles in teaching students the development of quality software. Even though students study review approaches in software engineering courses, they seldom get the opportunity to practice these concepts. Existing literature has not fully addressed all aspects of software reviews at the undergraduate level. There is little empirical evidence available on the integration of review activity in the software engineering curriculum. The integration of the current review approaches into the curriculum is also not straightforward. They do not adequately address the constraints and limitations enforced by the academic environment.

To address the limitations of current review methodologies, we propose two variations of team reviews: Similar Domain Review and Cross-Domain Review without author. The paper compares the proposed techniques with the existing team review approach and measures their effectiveness in terms of defect identification and quality of work product.

The purpose of this study is to find a review approach that is better suited for teaching reviews to undergraduates and determine how the various review techniques help students to fully appreciate the review process.

The study was conducted in a series of experiments at the National University of Computer and Emerging Sciences – Foundation for Advancement of Science and Technology (NUCES-FAST) in Pakistan. The subjects involved were undergraduates taking a one-semester software engineering course. Reviews were carried out on artifacts produced at the end of two phases of software development: requirements and design.

The rest of the paper is organized as follows. The next section discusses the background work. Section 3 gives an overview on the prevalent forms of peer review. Section 4 presents the proposed team review techniques. Section 5 details the research methodology for software requirements and design. The hypotheses for the empirical analysis are stated in Section 6. Section 7 presents data collection and the results of the study. Student feedbacks are presented in Section 8. Finally, conclusions and directions for future work are given in Section 9.

2. LITERATURE REVIEW

This section discusses the work done on teaching and conducting peer reviews.

Collofello presented a guide for teaching and conducting software reviews in one-semester software engineering course [3]. He suggests that two to three lectures on reviews should be given to students. The paper lists the contents (related to reviews) that should be delivered to students. The author highlights the importance of review activity, suitable review technique selection for a project, behavioral factors involved in a review process, review techniques (inspection, walkthroughs and audits), participant roles in review process and planning and reporting of review activity. He suggests some review exercises and different phases of software project where reviews can be incorporated.

Collofello also proposes an SEI curriculum module in [4]. The module aims to provide a detailed discussion on different aspects of the software review process, to enable students to conduct effective software reviews. Cross incorporates additional material, including guidelines and notes for teaching and conducting the review process, to the SEI module in [5].

Four variations of walkthrough reviews: demonstration, team, transparency, and presentations are given by Sullivan for teaching students in [19]. Similar to Collofello [3, 4] and Cross [5], no empirical evidence is provided; making it difficult to determine which approach is actually better and should be applied. Sullivan incorporates peer reviews in five courses and observes in the case of team reviews that students produce their work products ahead of time due to "peer pressure". Moreover, sharing of work gives students a chance to learn in a far better manner.

Clark reports his experiences of six years in teaching and conducting peer testing in the Software Engineering Project course [2]. Peer testing is performed in two different ways: paired-peer code reviews and group peer inspections. In the former, students are paired with members from other groups and conduct reviews of work products of both members. Group peer inspections are conducted by a testing team consisting of members from other teams. He concludes that both approaches are useful in terms of defect detection.

To conduct inspection process effectively, *"Scenes of Software Inspections"* have been discussed in [6]. The scenarios discuss group settings for correctly and incorrectly performed inspection processes. These can be used for delivering lectures or self-study.

Attitude measurement techniques developed by psychologists are used to measure changes in student's attitude regarding code inspection are given in [1]. They suggest that students show *"intellectual acceptance"* towards code inspection subsequent to watching a video on inspection [6], and *emotional acceptance* only when they had participated in the code inspection process themselves.

An automated, online system Praktomat [22] is used to improve the quality of programs developed by students enrolled in an introductory programming course. Praktomat handles allocation and submission of programming assignments. It performs automated testing and only accepts programs that can be compiled. Students can review assignments of their colleagues. Praktomat also allows program resubmission based on the reviews. Final submissions are graded by instructors. To avoid plagiarism, the system allows reviews can be performed only after successful submission of assignment, and the author and reviewer are not given the same assignment. More than two-thirds of the students are of the view that this process has enhanced their programming skills. .

3. FORMS OF PEER REVIEW

A quality-driven organization practices a variety of peer review methods. The types of peer reviews ranges from formal to informal according to Karl Wiegers [20]. These are discussed as follows:

Inspection: A well-defined process including several stages: planning, overview, preparation, inspection, rework and follow-up [7, 8, 20]. It assigns designated roles (moderator, reader, author, recorder and inspector) to the review participants [12]. Inspection requires rigorous preparation and adequately trained Inspectors [10, 12 and 20].

Team Review: Similar in structure to inspection but is not as rigid and formal as the inspection process as certain review steps can be shortened or even skipped and roles can be merged [20]. According to Weiger team review and structured walkthrough [21] are similar in nature.

Walkthrough: An informal review process with no entry or exit condition. The defects found are not conveyed to the managing body. Thus, there are no metrics that can be analyzed to improve the review process [20].

Pair Programming: An informal review process where two people work on one module at the same time. One develops and the other reviews. It is a non-procedural approach. It doesn't enforce any planning, or documentation to carry out the review. Reviews can be biased as there is no external source of feedback [20].

Passaround: The artifact or work product is passaround to be reviewed by a number of persons. It does not guarantee prompt response due to lack of controlled procedure [20].

Peer Deskcheck: The most inexpensive review technique as the review is carried out by two people only: author and another person [20]. It doesn't enforce any formalism in the review process.

Ad Hoc: Least formal of all review techniques, no process is followed, no results are recorded.

4. PROPOSED REVIEW APPROACHES

It can be seen from the literature review that prior research on software reviews has focused on providing material on teaching and conducting software reviews for the academia. The review material is in form of curriculum designs, guidelines, tutorials and videos to be incorporated into the various phases of software development. However, very little empirical evidence is available to evaluate which techniques are suitable for teaching reviews at the undergraduate level. In this paper, we present an empirical study to find out the review technique better suited to the constraints of the academic environment.

Another constraint not addressed in the literature is the level of expertise of students. The peer review techniques stated in the previous section range from formal to informal. Inspections on the extreme side of formal reviews assign unique roles to each participant and require inspectors to be domain experts. Undergraduates cannot have the necessary proficiency and domain knowledge. Also, it is not always possible to assign unique roles in a project group. On the other side of scale, we have informal reviews: walkthroughs, pair programming, passaround and ad hoc reviews. They are unstructured and do not enforce any formalism. The middle ground between the two is team reviews. We propose two variations of team review techniques: Similar Domain Review and Cross-Domain Review. It is also proposed to design the team reviews to ensure that all students play all roles in parallel, especially that of the reviewer.

One other issue to be addressed is checklist preparation. Collofello [3] suggests that students carry out the planning and scheduling of the review process themselves. He further suggests that checklist preparation also needs to be done by the students. We propose that tailored checklist based on quality attributes ought to be provided to the students by the instructor as students cannot determine all necessary aspects of quality at this stage. We also propose the use of checklist in both the existing team review technique and proposed variations: SDR and CDR.

The proposed variations on team review techniques are listed below:

4.1 Similar Domain Review

In this approach, each group of students reviews the artifact produced by other groups and vice versa. The involved groups work on the same problem domain independently. It doesn't matter whether or not the author is present in the review process as each group has knowledge of the domain. We have evaluated this technique for groups of two members and same sub-domains. Our proposed approach can also be extended to groups of more than two members and similar sub-domains.

4.2 Cross-Domain Review without Author (Dissimilar Sub-domains)

In this review approach, students from different groups review the artifact produced by other groups (from different sub-domains) and vice versa. The author is not present in the review process. The advantage of Cross-Domain Review is that it assists students to get the picture of the entire system instead of concentrating only on their own module. This insight facilitates and smoothes the interfacing and integration process.

SDR and CDR without author are compared with the existing team review approach which we will refer to as *CDR with author* from now on. These techniques are evaluated for their effectiveness in terms of defect identification in the requirement and design phases of software development life cycle to find out what works better for students.

Self Review: We also suggest self review, a preliminary training approach. Self review is an individual activity, where each group member individually reviews work product produced by the group. This activity serves dual purposes: a) it familiarizes students with review technique and checklist, b) it gives them an opportunity to learn from their own mistakes.

5. RESEARCH METHODOLOGY

This section describes the setting, subjects and process needed to carry out the research. The subjects involved in the study were students at the baccalaureate level enrolled in a one-semester third-year software engineering course at the National University of Computer and Emerging Sciences – Foundation for Advancement of Science and Technology (NUCES-FAST) in Pakistan.

The study was conducted on the term project for the course which was a compulsory part of the course. The class strength was divided into groups of size two each. It was ensured that average CGPA of all groups was approximately equal. Students worked on five modules of an ERP system. Each group was assigned one module. Multiple groups worked on one module independently. The study was carried out in two major phases: the requirements phase and the design phase. The requirements phase was further divided into two modules: an initial requirement gathering process and a detailed software requirement specification (SRS) document.

The educational objective of the study was to teach students the importance of software reviews. From a research point of view, we want to evaluate the effectiveness of SDR and CDR. We will show that the proposed SDR is more effective in detecting defects than CDR (with/without author). Another note-worthy result is that the proposed CDR-without author is better than CDR with author (the existing team review approach). We will also see that early defect detection makes it possible for students to integrate changes and enhance the quality of software developed.

5.1 Initial Software Requirement Specification Phase

To ease the development process, students were asked to develop the requirement specification document in two sub-phases. In the initial phase students delivered high level functional and non-functional requirements, use case diagram, external interfaces and context level data flow diagram (DFD). In the second phase, students were required to design detailed use cases and high-level DFDs. To ensure consistency in format and contents of SRS document, tailored IEEE Standard templates [11] for each phase augmented by examples related to the ERP system were provided to the students.

5.1.1 Checklist Preparation

Checklist-based reading is regarded as a best practice [14]. In checklist-based reading, a list of questions is used to find defects in the product to be reviewed.

We propose to use checklist-based reading because it is a) easy to use, b) amenable to customization and c) a good reading technique to begin with from an educational perspective. A tailored version of the defect checklist [9, 18] was developed. Questions were simplified and advanced concepts were removed to increase understandability and performance. The checklist focused on the basic quality attributes of requirements and the SRS document: organization, completeness, correctness,

consistency, lack of ambiguity, non-functional aspects and traceability.

5.1.2 Training on Reviews

Submission of initial SRS was followed by a tutorial on software reviews with an emphasis on requirements. The tutorial includes:

1. Definition and importance of software reviews
2. Types of software reviews
3. Importance of requirements in software development life cycle
4. Quality attributes of requirements
5. How to write the specification document
6. How to follow the checklist in order to detect defects
7. Ethics of conducting software reviews

5.1.3 Self Review

The self review activity was conducted in a ninety minute session. In this activity students performed self review independently without any discussion with their group members. The objective was to ensure that all students participated in the exercise and realized their mistakes. Defect report forms were collected at the end of the activity. As expected, most of the students identified defects related to format and structure of document. This training acted as a brain-storming session for the upcoming review activities.

5.1.4 Similar Domain Review

After giving an overview and an exercise session on performing reviews, SDR was conducted for each module in a two-hour session. Each group reviewed the artifact of another group working on the same module. The author was not present during the review activity. Students submitted defects reported by group, but in case of non-consensus they could also submit defects identified by an individual.

5.1.5 SRS Integration

In this activity, the SRS documents for each module were integrated. Groups working on the same module were combined into one large group. A group lead was assigned to act as a moderator. This activity aims to enhance the following capabilities in students:

1. Group work
2. Consistent and error free integration of modules
3. Defect resolution

Students were able to come up with a consolidated list of defects based on a) defects reported in SDR, b) defects identified during integration and c) feedback from the instructors. These defects were then removed from the integrated SRS document. The instructors rechecked the integrated document and approved it for the next phase.

5.2 Detailed Software Requirement Specification Phase

The integrated version of SRS for each module serves as a basis for development of detailed SRS. Integration ensured to a great extent that no identified defects are carried forward from the initial SRS. In this phase, students were asked to develop use case model, use case analysis, level-1 and level-2 DFD, architecture diagram, object model, data dictionary, entity relationship model and graphical user interfaces. Students again worked in groups of two.

5.2.1 Checklist Preparation

The checklist is revised to include items related to basic quality attributes of use case analysis and new items for the detailed SRS. The checklist included questions related to completeness, organization, correctness, consistency, lack of ambiguity, modifiability and traceability. To enhance understanding and performance, the questions taken from [9, 18] were simplified and advanced concepts were removed. A short lecture was delivered to explain new items from the checklist and the experiences from previous reviews.

5.2.2 Similar Domain Review

SDR was conducted for each module as those conducted in the initial requirement phase. Each group reviewed the artifacts produced by another group working on the same module. The author was not present in the review process. Students submitted defects report forms after a two-hour review activity.

5.2.3 Cross-Domain Review (with and without author)

After SDR was conducted, CDR of the detailed SRS was conducted. Half of the total artifacts produced were reviewed by persons belonging to different groups working on different sub-domains. The other half of documents were reviewed by groups formed on the same pattern but with a difference, that the review group also included one author of the work product. It took two and half hours to successfully complete this review activity (Due to lack of sub-domain knowledge with reviewers, more time was assigned for this activity). Defect report forms were submitted at the end of the review exercise.

5.3 Design Phase

For the design phase a procedure similar to that for the detailed SRS phase was followed. On the basis of reviewed and corrected work products from the detailed SRS, students developed class diagram and sequence diagrams. They were provided a defect checklist with questions related to software design. The questions related to completeness were taken from [9, 18]. Questions relating to inheritance, aggregation, cohesion, traceability and coupling were taken from heuristics given in [17]. Defect report forms from SDR and CDR (with and without author) were submitted by the students at the end of the activity.

6. HYPOTHESES

This section states the basis for conducting the empirical analysis of the review approaches proposed in section 4. To determine the effectiveness of the review process in assisting students to detect defects in the requirements and design phases of software development, we state the following propositions.

P1: There is no difference in defects identified by students through SDR and CDR approach.

P1′: Defects identified by students through SDR approach are significantly higher than with CDR.

Table 1 a. Alternate Hypotheses for Defect Identification in Software Requirements for Proposition P1

$H10_1'$: Total defects identified through SDR are higher than defects identified through CDR.
$H10_2'$: Major defects identified through SDR are higher than defects identified through CDR.
$H10_3'$: Minor defects identified through SDR are higher than defects identified through CDR.
$H10_4'$: Wrong defects identified through CDR are higher than defects identified through SDR.
$H10_5'$: Completeness defects identified through SDR are higher than defects identified through CDR.
$H10_6'$: Correctness defects identified through SDR are higher than defects identified through CDR.
$H10_7'$: Modifiability defects identified through SDR are higher than defects identified through CDR.

Table 1 b. Alternate Hypotheses for Defect Identification in Software Requirements for Proposition P2.

$H20_1'$: Total defects identified through CDR without author are higher than defects identified through CDR with author.
$H20_2'$: Major defects identified through CDR without author are higher than defects identified through CDR with author.
$H20_3'$: Minor defects identified through CDR without author are higher than defects identified through CDR with author
$H20_4'$: Wrong defects identified through CDR without author are higher than defects identified through CDR with author.
$H20_5'$: Completeness defects identified through CDR without author are higher than defects identified through CDR with author.
$H20_6'$: Correctness defects identified through CDR without author are higher than defects identified through CDR with author.
$H20_7'$: Modifiability defects identified through CDR without author are higher than defects identified through CDR with author.

Table 2 a. Alternate Hypotheses for Defect Identification in Software Design for Proposition P1

$H11_1'$: Total defects identified through SDR are higher than defects identified through CDR.
$H11_2'$: Major defects identified through SDR are higher than defects identified through CDR.
$H11_3'$: Minor defects identified through SDR are higher than defects identified through CDR.
$H11_4'$: Wrong defects identified through CDR are higher than defects identified through SDR.
$H11_5'$: Completeness defects identified through SDR are higher than defects identified through CDR.
$H11_6'$: Inheritance defects identified through SDR are higher than defects identified through CDR.
$H11_7'$: Aggregation defects identified through SDR are higher than defects identified through CDR.
$H11_8'$: Cohesion defects identified through SDR are higher than defects identified through CDR.
$H11_9'$: Traceability defects identified through SDR are higher than defects identified through CDR.
$H11_{10}'$: Coupling defects identified through SDR are higher than defects identified through CDR.

Table 2 b. Alternate Hypotheses Defect Identification in Software Design for Proposition P2

$H21_1'$: Total defects identified through CDR without author are higher than defects identified through CDR with author.
$H21_2'$: Major defects identified through CDR without author are higher than defects identified through CDR with author.
$H21_3'$: Minor defects identified through CDR without author are higher than defects identified through CDR with author
$H21_4'$: Wrong defects identified through CDR without author are higher than defects identified through CDR with author.
$H21_5'$: Completeness defects identified through CDR without author are higher than defects identified through CDR with author.
$H21_6'$: Inheritance defects identified through CDR without author are higher than defects identified through CDR with author.
$H21_7'$: Aggregation defects identified through CDR without author are higher than defects identified through CDR with author.
$H21_8'$: Cohesion defects identified through CDR without author are higher than defects identified through CDR with author.
$H21_9'$: Traceability defects identified through CDR without author are higher than defects identified through CDR with author.
$H21_{10}'$: Coupling defects identified through CDR without author are higher than defects identified through CDR with author.

P2: There is no difference in defects identified by students through CDR approach both with and without author.

P2′: Defects identified by students through CDR approach without author are significantly higher than through CDR approach with author.

We also propose alternate hypotheses based on propositions P1 and P2 for defect identification in both software requirements and design. The alternate hypotheses for requirements are stated in Table 1a and Table 1b. Those for design are given in Table 2a and Table 2b.

7. DATA COLLECTION AND ANALYSIS

The data for the study is collected from defect report forms submitted by students at the end of the SRS and design reviews. Defect Rate Metric [13] is used to make the data consistent and comparable within reviews and phases. The metric is defined as:

$$\text{Defect Rate} = \frac{\text{Number of Valid defects Identified}}{\text{Total Review time}}$$

As the time assigned for reviews varied in each of the phases, all data points are converted and standardized using the above metric. The hypotheses stated in Section 5 are tested at 5% level of significance. Normality of each data set is checked using the Kolmogorov-Smirnov (non-parametric) test. For data lying on the normal curve, t-test for independent samples is used. Mann-Whitney U test is applied on the rest of the data.

7.1 Analysis of Detailed SRS

In the detailed SRS phase in Section 5.2, three types of reviews are conducted. Table 3a summarizes the results of hypotheses testing for SDR versus CDR. The results show that for total, major and completeness defects p-value is less than 0.05 accepting alternate hypothesis. That is, more defects in these categories were identified by groups working on SDR as compared to groups working on CDR.

CDR performs worst in wrong defect identification. This result is intuitive. Due to deficient domain knowledge, students take assumptions, leading to higher defect rates as compared to similar domain review. The results are justifiable because the domain knowledge owned by groups working on similar sub-domains results in identification of more major defects related to completeness.

The descriptive statistics presented in Table 4 also highlight the above mentioned results. On the contrary, for minor defects including defects related to correctness and modifiability CDR performed almost equal to SDR.

Next we present the hypothesis results of CDR with author and CDR without author (see Table 3b). Interestingly, as opposed to the common team review practice (where author is present in the review process), we observe that the author's contribution in the review done by undergraduate students leads to lower defect rate. This is because the reviewer gets easily convinced by the author's arguments due to insufficient domain knowledge of the author's module.

The mean defect rate for other attributes of SRS (excluding minor defects) is also slightly higher for CDR without author reviews (see descriptive statistics Table 4).

The hypothesis results show that for major defects (completeness related defects) p-value is less than 0.05 so we accept alternate hypothesis which states that our proposed approach CDR without author is better than the existing CDR with author.

Moreover, p-values also show that review groups having no author as a guide identified more total and wrongly detected defects. On the other hand, for minor defects (correctness and modifiability) p-values are greater than 0.05 causing acceptance of null hypotheses. Thus, for minor categories CDR with author performed almost equal to CDR without author.

In the previous sections we have specified that the defect checklist for initial and detailed SRS phases had items related to traceability of the SRS document. Traceability is an important attribute of an SRS; it helps in determining the completeness and consistency of a work product. Identified defects related to this category are next to none. As a result, traceability is not included in the analysis.

7.2 Analysis of Design

The analysis of the software design developed by the undergraduate students is performed in the similar manner as for the SRS reviews. The results of hypotheses testing for SDR versus CDR are summarized in Table 5a. It can be observed that for major, total and completeness defects, p-value is less than 0.05 rejecting null hypothesis. This allows us to conclude that more defects are identified through SDR as compared to CDR.

Table 3 a. Test Statistics of Detailed SRS: SDR vs. CDR

Hypot-heses	Defect Type	SDR versus CDR		
		Equality of Variance	P-value	Alternate Hypothesis Accepted
$H10_1'$	Total	Yes	0.03	Yes
$H10_2'$	Major	Yes	0.035	Yes
$H10_3'$	Minor	Yes	0.135	No
$H10_4'$	Wrong	Yes	0.045	Yes
$H10_5'$	Completeness	Yes	0.00	Yes
$H10_6'$	Correctness	Yes	0.238	No
$H10_7'$	Modifiability	Yes	0.5	No

Table 3 b. Test Statistic of Detailed SRS: CDR with Author vs. CDR without Author

Hypot-heses	Defect Type	CDR without author versus CDR with author		
		Equality of Variance	P-value	Alternate Hypothesis Accepted
$H20_1'$	Total	No	0.025	Yes
$H20_2'$	Major	Yes	0.0005	Yes
$H20_3'$	Minor	Yes	0.33	No
$H20_4'$	Wrong	Yes	0.04	Yes
$H20_5'$	Completeness	Yes	0.019	Yes
$H20_6'$	Correctness	Yes	0.185	No
$H20_7'$	Modifiability	Yes	0.36	No

Table 4. Descriptive Statistic of Detailed SRS

Hypo-theses	Defect Type	SDR versus CDR					Hypo-theses	Defect Type	CDR without author versus CDR with Author				
		Review Approach	Min	Max	Mean	S.D			Review Approach	Min	Max	Mean	S. D
$H10_1'$	Total	SDR	9.33	27.33	15.33	4.94	$H20_1'$	Total	With Author	4.67	15.33	9.52	4.51
		CDR	2	16	11.69	4.46			Without Author	11.33	18	16.19	2.42
$H10_2'$	Major	SDR	6.67	24.67	11.07	4.66	$H20_2'$	Major	With Author	2.67	10.67	5.9	3.2
		CDR	2	12.67	8.1	3.23			Without Author	8	14.67	12.57	2.35
$H10_3'$	Minor	SDR	2	8	4.46	2.18	$H20_3'$	Minor	With Author	2	5.33	3.61	1.48
		CDR	0	6.67	3.53	1.96			Without Author	1.33	6.67	4	1.72
$H10_4'$	Wrong	SDR	0.67	5.33	2.5	1.41	$H20_4'$	Wrong	With Author	0	3.33	1.2	1.28
		CDR	2	7.33	4.16	2.16			Without Author	2	7.33	3.46	2.23
$H10_5'$	Completeness	SDR	4.67	12	7.48	1.98	$H20_5'$	Completeness	With Author	1.33	7.33	2.95	2.03
		CDR	0	7.33	3.69	2.64			Without Author	3.33	12	6.38	3.33
$H10_6'$	Correctness	SDR	2	12	6.41	3.1	$H20_6'$	Correctness	With Author	2.67	10.67	6	3.48
		CDR	1.33	15.33	7.38	3.72			Without Author	1.33	15.33	8	4.5
$H10_7'$	Modifiability	SDR	0	2.67	0.82	0.94	$H20_7'$	Modifiability	With Author	0	1.33	0.57	0.59
		CDR	0	3.33	0.82	1.12			Without Author	0	3.33	0.76	1.24

Table 5 a. Test Statistic of Design: SDR versus CDR

Hypo-thesis	Defect Type	SDR versus CDR			
		Test Statistic	Equality of Variance	P-value	Acceptance of Alternate Hypothesis
$H11_1'$	Total	T-test	Yes	0.0365	Yes
$H11_2'$	Major	T-test	Yes	0.003	Yes
$H11_3'$	Minor	T-test	No	0.202	No
$H11_4'$	Wrong	T-test	Yes	0.0365	Yes
$H11_5'$	Completeness	T-test	Yes	0.015	Yes
$H11_6'$	Inheritance	T-test	Yes	0.28	No
$H11_7'$	Aggregation	Mann-Whitney U	-	0.12	No
$H11_8'$	Cohesion	T-test	Yes	0.42	No
$H11_9'$	Traceability	T-test	Yes	0.35	No
$H11_{10}'$	Coupling	Mann-Whitney U	-	0.34	No

Table 5 b. Test Statistic of Design: CDR with Author vs. CDR without Author

Hypo-thesis	Defect Type	CDR without author versus CDR with author			
		Test Statistic	Equality of Variance	P-value	Acceptance of Alternate Hypothesis
$H21_1'$	Total	T-test	Yes	0.0005	Yes
$H21_2'$	Major	T-test	Yes	0.002	Yes
$H21_3'$	Minor	T-test	Yes	0.02	Yes
$H21_4'$	Wrong	T-test	Yes	0.043	Yes
$H21_5'$	Completeness	T-test	Yes	0.001	Yes
$H21_6'$	Inheritance	T-test	Yes	0.277	No
$H21_7'$	Aggregation	Mann-Whitney U	-	0.32	No
$H21_8'$	Cohesion	T-test	Yes	0.1	No
$H21_9'$	Traceability	T-test	Yes	0.245	No
$H21_{10}'$	Coupling	T-test	Yes	0.346	No

The results are perceptive, as SDR reviewers have domain knowledge. On the other hand, for defect rates related to inheritance, aggregation, cohesion and coupling CDR performed almost equal to SDR. We believe that these defects can be identified equally well by any type of review (given the reviewers know and understand object oriented design principles). The statistics in Table 6 support our argument.

In the later stage, analysis of CDR with and without author is carried out. Table 6 displays statistics of these reviews. Hypotheses results given in Table 5b also verify our findings that the author's contribution in review done by undergraduate students leads to lower defect rate.

Moreover, review groups with no author in the review process performed much better in identifying major defects related to completeness. The mean defect rate for other attributes of SRS (excluding minor defects) is also higher for CDR without author reviews (see Table 6). Interestingly, in the design phase more wrong defects are identified in cross-domain reviews when the author is not present. These results are similar to the detailed SRS. But the ratio of these defects is so small that they do not affect the overall effectiveness of cross-domain reviews without author.

8. STUDENT FEEDBACKS

Student feedbacks were collected at the end of the review activity by circulating a questionnaire. Following results were recorded:

- 76% students thought SDR was better; 24% considered CDR as a better review approach.
- 60% students thought they identified more errors in SDR; 40% thought CDR helped identify more defects.
- 72% students said they faced no problems in SDR; 28% were of the view that difference in development approaches acted as a hindrance in conducting SDR.
- 12% students said they faced no problems in CDR; 88% were of the view that difference in domain was an obstacle in conducting CDR.
- 50% students were of the view that the presence of author in CDR without author could have reduced review time; the other half thought the presence of author in CDR with author made the review difficult, as he always justified himself and was seldom convinced.

Some general comments were:

- 90% found the review activity a useful learning experience and it improved the quality of their deliverables.

Table 6. Descriptive Statistic of Design

Hypo-theses	Defect Type	SDR versus CDR					Hypo-theses	Defect Type	CDR without author versus CDR with Author				
		Review Approach	Min	Max	Mean	S.D			Review Approach	Min	Max	Mean	S. D
$H11_1'$	Total	SDR	5.33	16.67	10.92	3.52	$H21_1'$	Total	With Author	2.67	8.67	5.33	1.95
		CDR	2.67	14.00	8.35	3.46			Without Author	6.67	14.00	10.16	2.70
$H11_2'$	Major	SDR	5.33	13.33	9.17	2.92	$H21_2'$	Major	With Author	2.00	8.00	4.08	1.76
		CDR	2.00	10.00	6.00	2.40			Without Author	4.67	10.00	7.08	1.84
$H11_3'$	Minor	SDR	0.00	4.00	1.79	1.34	$H21_3'$	Minor	With Author	0.00	4.00	1.25	1.25
		CDR	0.00	6.00	2.35	1.97			Without Author	0.67	6.00	3.08	1.91
$H11_4'$	Wrong	SDR	0.67	3.33	1.48	1.09	$H21_4'$	Wrong	With Author	0.00	2.67	0.80	1.09
		CDR	0.67	4.67	2.59	1.35			Without Author	0.67	4.00	2.53	1.44
$H11_5'$	Completeness	SDR	4.00	12.67	7.12	2.23	$H21_5'$	Completeness	With Author	0.67	5.33	2.16	1.45
		CDR	0.67	10.00	4.71	3.02			Without Author	2.67	10.00	6.25	2.59
$H11_6'$	Inheritance	SDR	0.00	4.00	0.61	1.10	$H21_6'$	Inheritance	With Author	0.00	0.67	0.33	0.35
		CDR	0.00	2.00	0.41	0.57			Without Author	0.00	2.00	0.50	0.69
$H11_7'$	Aggregation	SDR	0.00	2.00	0.41	0.57	$H21_7'$	Aggregation	With Author	0.00	0.67	8.33	0.23
		CDR	0.00	2.67	0.25	0.74			Without Author	0.00	2.67	0.41	0.93
$H11_8'$	Cohesion	SDR	0.00	4.00	0.82	1.33	$H21_8'$	Cohesion	With Author	0.00	1.33	0.50	0.47
		CDR	0.00	4.00	0.92	1.10			Without Author	0.00	4.00	1.16	1.32
$H11_9'$	Traceability	SDR	0.00	5.33	1.64	1.62	$H21_9'$	Traceability	With Author	1.33	3.33	2.00	0.71
		CDR	0.00	3.33	1.84	1.02			Without Author	0.00	3.33	1.66	1.12
$H11_{10}'$	Coupling	SDR	0.00	2.00	0.35	0.64	$H21_{10}'$	Coupling	With Author	0.00	0.67	0.16	0.30
		CDR	0.00	1.33	0.20	0.42			Without Author	0.00	1.33	0.25	0.49

- 60% said they gained useful knowledge for their own project by testing for another project in SDR.
- 50% said that CDR helped in the integration.
- 80% wanted the activity to continue in other courses and were willing to use it for their final year projects.

9. CONCLUSION AND FUTURE WORK

Reviews are one of the main techniques for developing quality software. The existing literature, however, contains very little empirical evidence on suitable techniques for teaching and conducting reviews at the undergraduate level. In this paper, we presented an empirical comparison of CDR with author (the existing team review technique) with our proposed approaches SDR and CDR without author.

Our main findings are that CDR without author is better that CDR with author in terms of major defect identification in software requirements and design at the undergraduate level. Secondly, SDR is more effective in defect detection process than CDR.

Our experience and results lead us to suggest that the whole class of undergraduate students taking software engineering course should work as a group. This allows them to work on a reasonably large modular solution, where multiple groups develop same sub-modules. This approach has two-fold benefits. Firstly, reviewing the same module developed by other groups helps students to look at the artifact from diverse angles. They can appreciate and criticize someone else's approach and gain useful knowledge for their module. Secondly, CDR helps students in the interfacing and integration of the software system.

Additional benefits of teaching and conducting reviews to undergraduates are that the students are able to identify defects at an early stage allowing them to improve software quality and appreciate the importance of reviews.
Future research directions include repeating the study over different institutes and students over the years to validate the results obtained. We also plan to carry out reviews on the entire software life cycle especially the implementation phase.

10. REFERENCES

[1] Bailey, D., Conn, T., Hanks, B. and Werner, L. 2003. Can we influence students' attitudes about inspections? Can we measure a change in attitude? In Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET). IEEE, (Mar. 2003), 260- 267.

[2] Clark, N. 2004. Peer testing in software engineering projects. In Proceedings of the 6th Conference on Australasian Computing Education - Volume 30 (Dunedin, New Zealand). R. Lister and A. Young, Eds. ACM International Conference Proceeding Series, vol. 57. Australian Computer Society, Darlinghurst, Australia, 41-48.

[3] Collofello, J. S. 1987. Teaching technical reviews in a one-semester software engineering course. In Proceedings of the 18th SIGCSE Technical Symposium on Computer Science Education (Missouri, United States, February 19 - 20, 1987). SIGCSE '87. ACM, New York, NY, 222-227.

[4] Collofello, J.S. 1988. The Software Technical Review Process. Curriculum Module SEI-CM-3-1.5, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa.

[5] Cross, J. A. 1988. Support Materials for the Software Technical Review Process. Support Materials SEI-SM-3-1.0, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa.

[6] Deimel, L. 1991. Scenes of Software Inspection Video Dramatizations for the Classroom. Software Engineering Institute, Carnegie Mellon University.

[7] Fagan, M.E. 1976. Design and Code Inspections to Reduce Errors in Program Development. IBM Systems Journal, 15, 182-211.

[8] Fagan, M.E. 1986. Advances in Software Inspections. IEEE Software SE-12, 744-751.

[9] http://www.tol.oulu.fi/users/ilkka.tervonen/WChecklist.pdf

[10] Humphery, W.S. 1989. Managing the Software Process. Addison-Wesley Publishing.

[11] Institute of Electrical and Electronic Engineers. Standard for Software Reviews and Audits. (Jun 1989) ANSI/IEEE Std 1028-1988.

[12] Institute of Electrical and Electronic Engineers. Recommended Practice for Software Requirements Specifications, (Dec 1993) IEEE Std 830-1993.

[13] Kan, S. 2002. Metrics and Models in Software Quality Engineering. Addison-Wesley Publishing.

[14] Laitenberger, O. and DeBaud, J. 2000. An encompassing life cycle centric survey of software inspection. J. Syst. Softw. 50, 1 (Jan. 2000), 5-31.

[15] Neil, D.O. 2001. Peer Reviews, Encyclopedia of Software Engineering, Wiley, New York.

[16] Pressman, R.S. 2001. Software Engineering a Practitioner's Approach. McGraw Hill International.

[17] Reil, A. J. 2002. Object-Oriented Design Heuristics, Addison Wesley.

[18] R.S. Pressman & Associates, Inc

http://www.rspa.com/checklists/index.html

[19] Sullivan, S.L. 1994. Reciprocal Peer Reviews. . In ACM SIGCSE Bulletin 26, 1, (Mar. 1994), 314 -318.

[20] Wiegers, K.E. 2002. Peer Reviews in Software: A Practical Guide. Addison-Wesley Professional.

[21] Yourdon, E. 1989. Structured Walkthroughs. 4th Ed. Englewood Cliffs, N.J.: Yourdon Press.

[22] Zeller, A. 2000. Making students read and review code. In Proceedings of the 5th Annual SIGCSE/SIGCUE Iticseconference on innovation and Technology in Computer Science Education (Helsinki, Finland, July 11-13, 2000). ITiCSE '00. ACM, New York, NY, 89-92.