# Cognitive Factors in Perspective-Based Reading (PBR):

# A Protocol Analysis Study

Bryan Robbins
*Department of Computer Science and Engineering*
*Mississippi State University*
*bryan AT btr3 DOT com*

Jeff Carver
*Department of Computer Science*
*University of Alabama*
*carver AT cs DOT ua DOT edu*

## Abstract

*The following study investigated cognitive factors involved in applying the Perspective-Based Reading (PBR) technique for defect detection in software inspections. Using the protocol analysis technique from cognitive science, the authors coded concurrent verbal reports from novice reviewers and used frequency-based analysis to consider existing research on cognition in software inspections from within a cognitive framework. The current coding scheme was able to describe over 98% of the cognitive activities reported during inspection at a level of detail capable of validating multiple hypotheses from literature. A number of threats to validity are identified for the protocol analysis method and the parameters of the current experiment. The authors conclude that protocol analysis is a useful tool for analyzing cognitively intense software engineering tasks such as software inspections.*

## 1. Introduction

Empirical software engineering (ESE) is not unfamiliar with the importance of cognitive factors such as memory and problem solving processes in various software development tasks. The vast majority of existing work, however, utilizes a "black-box" approach to experimentation with cognitive factors, providing limited insight regarding the cognition associated with tasks. The current study approaches cognition at a lower level by introducing the protocol analysis method from cognitive science.

Section 1.1 describes Perspective-Based Reading, the task we have chosen to analyze. Section 1.2 describes the protocol analysis method we have chosen to analyze the task. Given the task and method, Section 1.3 details two research questions we believe can be approached, and Section 1.4 outlines a research plan designed to answer these questions.

### 1.1 Perspective-based reading

Perspective-Based Reading (PBR) is a reading technique for software inspections from the scenario-based reading (SBR) family of reading techniques designed for defect detection in a requirements document [2]. With PBR, inspectors take on the role of important document stakeholders (e.g. user, tester, or designer) and use stakeholder-centric scenarios called perspectives to guide defect detection. The scenario includes the development of a stakeholder-specific alternative model to assist in identifying defects. For example, the user model may be use cases, the tester model may be equivalence class test cases, and the designer model may be ER diagrams. Each PBR scenario helps the inspector determine whether the document under review provides the information necessary for constructing the assigned model. This process leads to the detection of defects in the document. PBR has proven to be an effective quality assurance approach, which is tailorable to organizational concerns [14].

Along with the introductions of both SBR and PBR, researchers have conducted empirical analysis arguing for the techniques' utility [2, 12]. Comparison studies, such as those comparing PBR to Checklist-Based Reading (CBR) or another reading technique, are common in the literature because the end results of effectiveness and efficiency are most important to practicing software developers. PBR's effectiveness and efficiency have been evaluated by many independent studies, and replications of those studies [2, 6, 9, 10]. The majority of these comparisons do find that PBR's performance exceeds that of other approaches (i.e. CBR and ad-hoc) for defect detection.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

A separate category of PBR studies address performance characteristics of the technique rather than only comparing performance variables. These studies allow for a better understanding of the determining factors of PBR's effectiveness. Early studies addressed the claims of PBR, such as whether the perspectives are really different [13]. More recent work has addressed the same question at a lower level, determining when and why PBR sometimes does not uphold its unique performance characteristics [10]. Many factors have been identified at the more detailed level offered by performance studies that affect the overall effectiveness and efficiency of the technique.

### 1.2 Protocol analysis

Protocol analysis is a method from the field of cognitive science designed to describe the cognitive activities associated with problem solving tasks. The PBR task qualifies as a problem solving task, which is focused on the problem of finding defects. Ericsson and Simon argue that concurrent verbal reports, also known as "think-aloud" protocols, collected during task performance can rightfully indicate the task-relevant contents of short-term memory. Furthermore, knowledge of these memory contents should lead to an understanding of cognitive processes and memory usage associated with the concurrent task [5].

A coding scheme provides the link between subjects' concurrent verbal reports and a reliable description of cognitive processes that can be analyzed. Also important to the coding scheme is the underlying cognitive model supporting the codes. In our case, only a simple model of memory is necessary to derive a coding scheme, as more advanced cognitive functions are not yet seen as critical to the task. Any existing PBR work that has formalized the cognitive processes or memory elements involved in carrying out the PBR task should also be used in the design of the coding scheme.

### 1.3 Research questions

We propose the application of the protocol analysis technique to the investigation of PBR. Protocol analysis allows for the description of cognitive activity as supported by a coding scheme. The idea of coding observations is common to qualitative data analysis. The coded data of protocol analysis should be able to describe the cognition associated with PBR. Such a description should allow for a more complete understanding of the task. The description should also be useful in the testing of existing hypotheses regarding PBR performance. This study has two goals, stated as complementary research questions Q1 and Q2:

Q1: Can the task of applying Perspective-Based Reading to a software artifact be described by a coding scheme?

Q2: Can a coding scheme describing Perspective-Based Reading support the unification of existing research on PBR performance?

### 1.4 Research plan

To address Q1, the development of a coding scheme is described in Section 2. The coding scheme is applied to verbal reports collected according to the methodology described in Section 3. Section 4.1 describes the results of the study in response to Q1. Using a systematic literature review as described in Section 3.4, a number of hypotheses regarding PBR performance were collected. Three hypotheses with a clear mapping to the current coding scheme and data collection context are addressed statistically in response to Q2. These results are given in Section 4.2. Threats to the validity of results are described in Section 5, and in light of the results and their threats, the conclusions of the study are given in Section 6.

## 2. Coding scheme development

As mentioned, the protocol analysis method depends on the development of a psychologically valid coding scheme. The validity and reliability of the coding scheme underpins the validity of all analyses performed on the coded data. The coding scheme for this study is adapted from cognitive psychology theory on memory processes.

### 2.1 Psychological model

A traditional memory model, as originally proposed by Atkinson and Shiffrin in 1968, is shown in Figure 1 [1]. This model serves as the basis for the development of the coding scheme. The model includes sensory input from the world through sensory memory and knowledge retrieval through long-term memory (LTM). Short-term memory (STM) is located in a critical position between LTM and sensory memory. All problem solving tasks are performed in STM, thus contents of STM and the processes used to manipulate these contents are critical to the analysis of problem solving tasks such as PBR. In fact, the limitations of STM, such as limitations on the number of facts readily available for manipulation, serve as the primary limitations for any problem solving task.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

**Table 1. Coding Scheme Codes and Descriptions**

| Symbol | Code | Description |
|---|---|---|
| ***Stimulus Input*** | | |
| RD | Read-Document | The reviewer is reading the assigned artifact |
| RS | Read-Scenario | The reviewer is reading the perspective-based scenario |
| RW | Read-Worksheet | The reviewer is reading the scenario-specific worksheet |
| ***Retrieval from LTM*** | | |
| KD | Recall-Document-Knowledge | The reviewer is recalling knowledge from the document |
| KT | Recall-Training-Knowledge | The reviewer is recalling knowledge from PBR training |
| KP | Recall-Perspective-Knowledge | The reviewer is recalling knowledge from the given perspective |
| KW | Recall-Web-Knowledge | The reviewer is recalling knowledge of web applications |
| KC | Recall-Conference-Knowledge | The reviewer is recalling knowledge of academic conferences |
| KS | Recall-Software-Knowledge | The reviewer is recalling knowledge of software engineering |
| ***Manipulation in STM*** | | |
| CK | Combine-Knowledge | The reviewer is combining existing knowledge into a single fact |
| AI | Assume-Intent | The reviewer is making an assumption about the document |
| ***Response Output*** | | |
| WW | Write-on-Worksheet | The reviewer is writing on the perspective worksheet |
| WD | Write-Defect | The reviewer is recording a defect |

The current coding scheme builds on four types of codes evident in the Atkinson and Shiffrin model of Figure 1: stimulus input, LTM retrieval, STM manipulation, and response output. We considered the activities relevant to PBR for each type of code in the model. The resulting coding scheme is presented in Table 1.

## 2.2 Coding scheme

Stimulus input for the PBR task comes from three different written documents. The artifact under review (referred to as the "document" in the coding scheme) is a requirements document in the current study. PBR also provides two supporting documents to the reviewer during document inspection. The first supporting document, a written scenario, details the steps of the PBR technique, including the creation and evaluation of the underlying perspective-specific model. A worksheet is also given to reviewers to aid in the step-by-step creation of the perspective-specific model. The Read Document (RD), Read Scenario (RS), and Read Worksheet (RW) codes support the reading of these written documents during the PBR task.

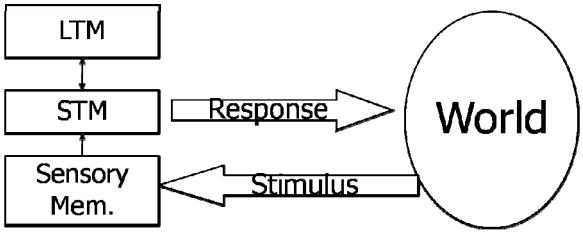


**Figure 1. Basic model of human memory**

LTM retrieval during PBR is perhaps the most interesting activity of the task. The recall of Document Knowledge (KD) is important to the reading of the document and identification of inconsistencies. Training Knowledge (KT) supports the overall review process as a reaction to learned processes. Carver's work on the role of background and experience in inspections identified two additional types of knowledge that we encode in our coding scheme [3]. Perspective-specific Knowledge (KP) should directly impact the review, as well as knowledge of the system domain. In the chosen requirements document, Knowledge of Web Applications (KW), Knowledge of the Standard Conference Paper Submission Process

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

(KC), and Knowledge of General Software Engineering Practices (KS).

The manipulation of facts in STM is supported by two rather generic codes in the current scheme. The Combination of Knowledge (CK) is essential to reasoning of any kind. The Assumption of Intent (AI) also appears to play a role in software inspections, as assumptions regarding the artifact can directly affect the remainder of the task. Finally, the applicable responses for the PBR task both involve writing. Subjects can Write on the Perspective-specific Worksheet (WW) or Record Defects (WD) as they are found. Both of the responses are native to the PBR task.

The resulting coding scheme as presented in Table 1 includes 13 codes, with 3 for stimulus input, 6 for retrieval from LTM, 2 for STM manipulation, and 2 for response output.

# 3. Methodology

This experiment makes use of the traditional PBR setting combined with the experimental setup required for protocol analysis. Section 3.1 describes the experimental procedure in detail. Section 3.2 characterizes the subject population. Finally, Section 3.3 briefly explains how the coding scheme was applied to the data in this study, and Section 3.4 describes the systematic review process used to identify hypotheses about PBR performance from software engineering literature.

## 3.1 Experimental procedure

The experiment was divided into two phases – a group training session on PBR and an individual data collection session.

During group training, all potential subjects were trained on the topics of requirements inspections, defect classification, defect reporting, and PBR. PBR training included practice with both the user and tester perspectives. The user perspective was based on the model of system sequence diagrams and the tester perspective based on equivalence class testing. Defect classes included ambiguous, extraneous, incomplete, inconsistent, and omitted information.

During individual data collection, subjects first submitted an experience survey similar to He and Carver's survey [6]. Then, subjects were acclimated to the verbal reporting activity by working 3-digit by 3-digit "long multiplication" problems. Finally, they performed the PBR task while providing concurrent verbal reports. The facilitator only intervened to maintain the appropriate depth and frequency of verbal reports. The document under review was a natural requirements document for a web-based conference paper submission and review system. The document had been used a year prior to the study to support the development of an actual system by a group of senior software engineering majors in a capstone course.

Data collection sessions lasted one hour, including all training activities. Therefore, subjects had data collection sessions of varying length and reached various points in the artifact. Because defect detection effectiveness was not a focus of the study, the variable length sessions did not affect analysis.

## 3.2 Subjects

The subjects for this study were graduate and upper-level undergraduate students enrolled in a software quality assurance course at Mississippi State University. In total, ten subjects participated in the training and inspection process. When each subject arrived for their inspection session, they were randomly assigned to either the user or the tester PBR perspective (ensuring that in total there were five subjects using each perspective). Unfortunately, three subjects had to be excluded due to inadequate process conformance and one had to be excluded due to technical issues with the recording equipment. As a result, only six subjects were left for analysis. Of those six subjects, four employed the user perspective and two employed the tester perspective.

## 3.3 Coding methodology

The individual data collection sessions produced four types of data per subject. First, the experience self-survey was submitted by each subject. The second set of data was an audio recording that captured the subjects' concurrent verbal reports during performance of the PBR task. Finally, two documents supporting the PBR task – the defect report and perspective-specific worksheet – were gathered.

The most important set of data for this study is the set of concurrent verbal reports. In order for the verbal data to be analyzed, the recordings (approximately 45 minutes of audio data for each subject) were transcribed into written form. Natural pauses in speech were used to separate verbal data into manageable individual "reports". The transcribed verbal data was coded independently by two coders. The individual coding process involved describing the activity in each report in terms of the codes in Table 1. The PBR-related data of the perspective worksheet and defect report were used to inform the coding process.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

For example, one verbal report contained, "Ok, well that's ambiguous because we don't know if the old paper will still be on the system or not." In this report, the subject is discovering a defect by combining (CK) knowledge just obtained from the document (RD) with knowledge of the web application domain (KW) that files behind a system are stored and can be retained or deleted. The subject then uses the combined knowledge to report the defect as ambiguous (WD). We can verify on the defect report obtained from the subject that a defect with this description and category was indeed reported. The resulting codes for this verbal report are CK(RD, KW), WD, which describe the process of combining new document knowledge with knowledge of web-based applications and reporting a defect. In like manner, the verbal reports from all 6 subjects were coded independently by the two coders.

After independent coding, the codes of each subject were compared. Matching codes were kept, and disagreements were resolved collectively in discussion between the two coders. One issue with the current coding scheme, discussed in detail in section 5.1 as a threat to the validity of this study, was that the descriptions of when to apply each code were very generic. This led to larger degrees of inference and subjectivity in the individual coding process. Ideally, codes should be very similar between independent coders, but our scheme often led to different interpretations that had to be resolved collectively. We feel that after the group coding, however, the resulting application of codes was consistent across all subjects, despite any subjectivity.

The resulting coded verbal reports serve as the basis for the analysis of Q1 and Q2, thus the coding process is very important to the validity of this and any protocol analysis study. We recognize a number of issues with the coding methodology utilized in this study. While we maintain the validity of the current codes, we certainly acknowledge that improvements will not only increase the validity of our results, but also reduce the overhead required during the coding process.

### 3.4 Systematic literature review

To identify any existing hypotheses about PBR performance, we performed a systematic literature review of the software engineering literature in the manner similar to that suggested by Kitchenham [8]. The identification of existing research is essential to the consideration of Q2, as we wish to evaluate the ability of the coding scheme to describe known trends in PBR performance.

For the literature review, we identified keywords of PBR, SBR, and software inspection research. Keywords were input into six literature databases of varying scope (some with a general focus, and some focused on computing literature). After filtering by title and abstract to remove irrelevant topics, data from the studies were extracted to answer two questions. First, we wanted to know the cognitive factors, if any, suggested by the research. Second, we wanted to see if the study verified the role of any identified cognitive factors empirically, and if so, under what conditions.

Very few studies directly investigated cognitive factors in PBR performance, but many suggested at some level their role in the PBR task. Three of the strongest hypotheses from a set of 20 in all studies were chosen for the analysis of Q2. Hypotheses from literature were mapped to testable hypotheses H1, H2, and H3 within the context of this study. These hypotheses are discussed in section 4.2.

## 4. Results

The results of the study correspond to the two original research questions, Q1 and Q2. The analysis of Q1, described in Section 4.1, is straightforward, addressing the ability of the coding scheme to describe the cognition associated with the PBR task. The analysis of Q2, discussed in Section 4.2, addresses the hypotheses from the systematic literature review.

### 4.1 Q1: Evaluation of coding scheme

We believe the coding scheme of Table 1 should sufficiently describe the cognition associated with PBR. Figure 2 shows a graphical breakdown of the coding scheme as applied to the verbal reports of the 6 subjects. Subjects U1, U2, U3, and U4 applied the user perspective. T1 and T2 applied the tester perspective. In Figure 2, CK codes are all combined as "CK(*, *)" and retrieval codes combined as "K".

Analysis of the frequencies presented in Figure 2 leads to an evaluation of the coding scheme. First, the WW and RD codes, which describe the simple tasks of reading the document and writing on the perspective worksheet, account for an average of nearly 60% of all codes (29.99% and 29.64% of all codes, respectively). The next most frequent code is the CK code, which signals the combination of facts in STM. CK accounts for 23.85% of all codes. After CK, the frequency of codes drops to 7.75% (for the WD code) and lower values for the remaining codes. Significantly, less that 2% of all verbal reports were unclassified by the current scheme (1.47%, UNDEF in Figure 2). Un-coded verbal reports were generally off-task

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

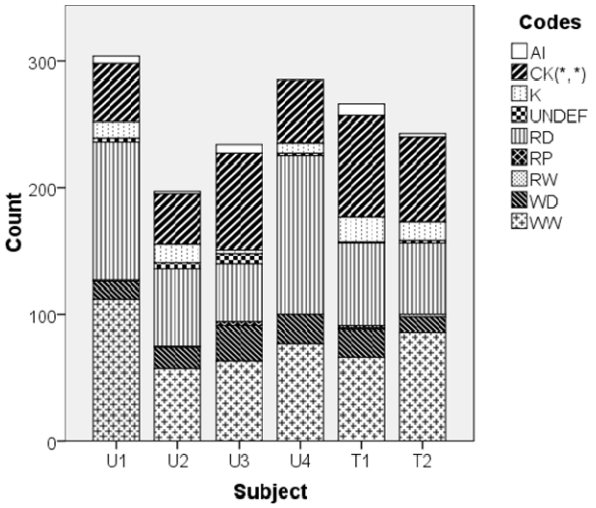


**Figure 2. Distribution of codes by subject**

commentary and higher-level commentary about the task, such as "I'm getting tired of using this worksheet," or "It seems redundant for me to keep writing this down, yet I continue."

One concern with the frequency distribution of the codes is that the RD and WW codes, which do not capture much interesting cognitive activity, dominate the verbal reports. This high frequency could negatively affect frequency-based analysis such as Pearson's Chi Square. However, the appearance of the RD and WW codes is certainly legitimate, as reviewers of any document must spend a great deal of cognitive effort simply reading the document and creating PBR's required alternative representation. Future studies, however, may not choose to utilize the perspective worksheets in order to obtain more appropriate data and support more rigid analysis.

We conclude that the coding scheme sufficiently described the cognitive activity associated with PBR as evidenced in concurrent verbal reports. However, the ability to simply describe cognitive activities with codes is certainly less interesting than using coded data to address existing hypotheses from within a cognitive framework. Testable hypotheses H1, H2, and H3, discussed in Section 4.3, address this second goal.

## 4.2 Q2: Existing hypotheses from literature

**4.2.1. H1: Defect triggers in software inspections.** The first literature-based hypothesis, identified by Chaar, et al. in a 1993 study, suggests that defect detection in software inspections is preceded by an activity labeled as a "defect trigger" [4]. We hypothesize that the CK code should correspond to the defect trigger of Chaar, et al. Specifically, we state testable hypothesis H1 as:

H1: Most defect identifications (i.e. sections of WD codes) will be preceded by combine knowledge actions (i.e. CK codes)

To test H1, we use Pearson's Chi Square Goodness of Fit Test. The Chi Square is valid for categorical data such as the coded verbal reports. The null hypothesis for this test is that there is no difference between the frequency of CK codes preceding defect reports and the average frequency of CK codes (i.e. 23.85%). In this analysis, a defect report is a section of sequential WD codes. WD codes often appear in sections to accommodate the identification, recording, and classification of defects, which may appear in separate verbal reports.

Of 52 separable WD sections across all 6 subjects, 38 sections were preceded by a CK code while only 14 were not. To test whether this frequency was significantly different than the expected frequency of 23.85%, a Pearson's Chi Square test was conducted. The results indicate that the instances of CK codes preceding WD sections are significantly more common than expected ($X2=69.383$, $p < .01$).

As mentioned in Section 4.1, there is concern that the analysis of frequencies in the coded data may be negatively affected by the high frequency of the RD and WW codes. For a more robust analysis of H1, we first exclude the RD and WW codes from the overall distribution. After making this assumption, the expected frequency of the CK code among the remaining codes is 59.2% (rather than 23.85% when all codes are considered). When the Pearson's Chi Square analysis is conducted with the new expected value, the results are still significant ($X2=4.146$, $p < .05$)).

In both the default and a more restricted coding scenario, the occurrence of CK codes preceding WD codes in verbal reports during the PBR task is significantly greater than expected based on the overall frequency of the code. This result leads to the statistical validation of H1 as indicated by the current coding scheme and collective set of verbal reports across all 6 subjects of the current study.

**H2: Nature of defects detected.** The multiple perspectives used during PBR should lead to less overlap in detected defects [2]. At least two studies have sought to verify this fundamental claim of PBR, but with mixed results [10, 11, 13]. With the granularity of the current study, the question of differences between perspectives can also be addressed. In fact, this analysis does not depend on the coding scheme, but the hypothesis was discovered via the systematic literature review of this study, so we included the analysis here. We do, however, propose a new type of analysis possible at this level. The following testable hypothesis H2 was developed:

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

H2: The type of defects found by subjects should be related to the assigned PBR perspective.

Across all 6 subjects of the current study, 27 unique potential defects were identified. Five of these potential defects were removed because they were not actually defects. Each of the 22 remaining defects was then categorized by the authors a posteriori as relating to the user perspective, tester perspective, or neither perspective. Note that this categorization did not take into account any information about which subjects identified the defect. The determination of a user defect compared with a tester defect was done solely on the content of the defect description. Table 2 shows the categorization of defects, in which user defects involved errors in the interaction sequence given by the requirements document and tester defects included issues with inputs and expected outputs for each function. In conducting this analysis, only 2 defects were related to neither perspective (these are excluded from Table 2 and the resulting analysis). Of the remaining, classifiable 20 defects, 13 defects were categorized as user defects and 7 as tester defects.

The a posteriori categorization of defects allows for a Chi Square Test for Independence to be conducted on the actual defects found by each perspective against the expected marginal values. Figure 3 displays the number of defects of each type found by each subject. The null hypothesis for H2 is that there is no difference between the number of perspective-unique defects and the number of non-perspective unique defects found by each subject. Chi Square analysis produces a significant effect ($X2= 7.403$, $p < 0.01$). Therefore, this study statistically validates H2, and supports the claim that PBR perspectives actually lead to the detection of perspective-related defects.

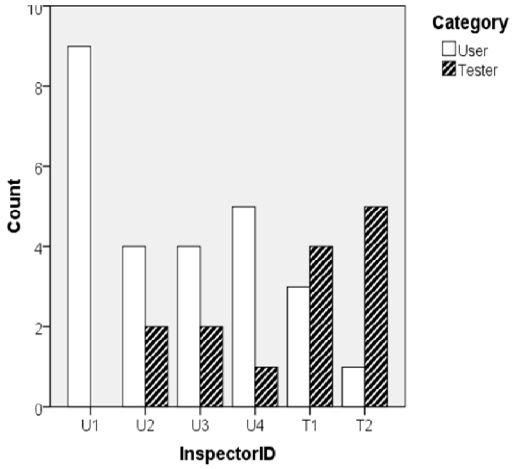


**Figure 3. Defects found per category by subject**

**H3: The effect of background and experience.** The final hypothesis investigated by our study focuses on the role of background and experience in software inspections. In a series of studies, Carver identified a number of variables that are hypothesized to affect software inspections [3]. Two variables were built into the current coding scheme: perspective-specific knowledge and domain knowledge.

The artifact used in the current study supported the

**Table 2. Defects found and perspective category**

| Defect | Category |
|---|---|
| What does "Upgrade Account" Info entail? | Tester |
| What form is displayed to the author for signup? | Tester |
| Payment info is outside the scope of the system. | Tester |
| Info about time slot seems out of place in Submit Review | Tester |
| What information is required on the review form? | Tester |
| Register conference does not describe details from admin | Tester |
| Announcement posting does not clarify possible "Issues receiving announcement file" | Tester |
| "Delete Paper" has no variations [alternative steps] | User |
| Number of papers that can be submitted at one time is ambiguous | User |
| Confusion between user and author | User |
| "View Comments on Own Paper" has no assumptions | User |
| Inconsistency between being logged in and having an account | User |
| Missing variation steps for emailing paper | User |
| Use of the word status is ambiguous | User |
| Add reason for declining | User |
| What screen is displayed after the system upgrades? | User |
| How does author "indicate wishes"? | User |
| How is paper displayed to author ? | User |
| Should have precondition of "paper must be accepted" before viewing comments | User |
| Submit camera ready paper is identical to resubmit paper. | User |

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

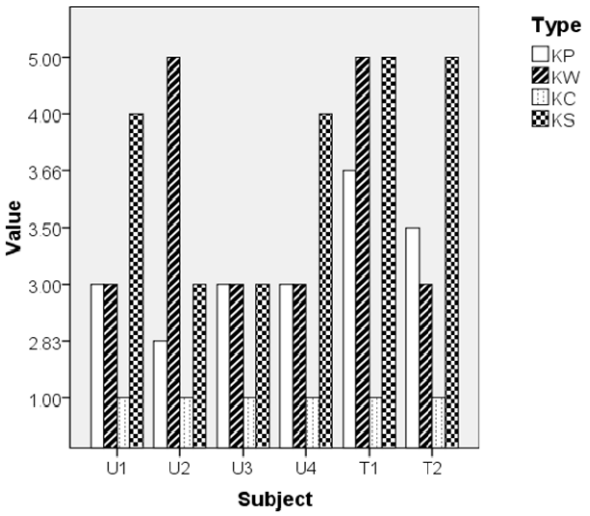


**Figure 5. Experience per category by subject**

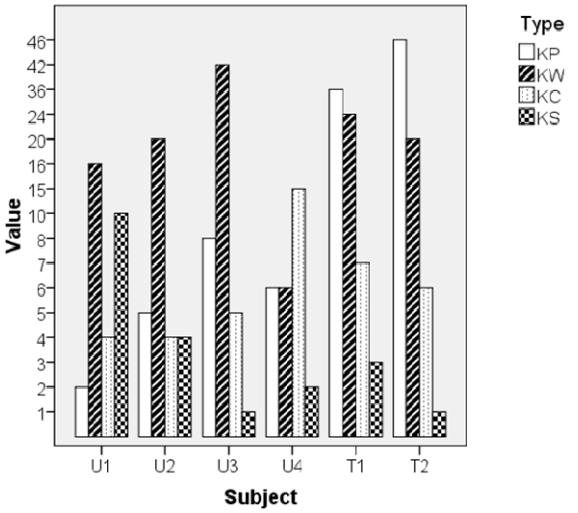


**Figure 4. Knowledge use per category by subject**

development of a conference paper submission system. The domain knowledge relevant to this system includes knowledge of the web domain (KW), knowledge of general software engineering (KS), and knowledge of the conference paper submission process (KC).

These knowledge types, in addition to perspective-specific knowledge (the KP code), have corresponding questions on the experience survey. Participant responses on the experience survey are averaged per subject by experience category. Each category maps 1-to-1 with the KP, KW, KS, and KC codes, allowing for testable hypothesis H3:

H3: The use of specific types of knowledge, as evidenced by the KP, KW, KS, and KC codes during inspection, should correspond to the respective experience levels indicated on the experience self-survey.

To test H3 for each type of knowledge indicated, Spearman's Rank Correlation Coefficient was used. Spearman's coefficient depends on ranked values rather than requiring a normal distribution, thus it is more appropriate for the analysis of categorical data [7]. Figure 4 shows the survey values for each subject by category. On the survey, a value of 1 indicates very little experience and a value of 5 indicates a high level of experience. Figure 5 shows the use of knowledge during inspection for each subject, also by knowledge category.

One observation from the survey data is that no subject rated themselves higher than 1.0 (very low) experience with the conference submission process (code KC). Therefore, no correlation is possible on the KC data, so it is excluded from the analysis.

Another interesting pattern in the use of knowledge is the stark contrast between perspectives in the use of KP knowledge. Inspectors employing the user perspective averaged only 5.25 KP uses per inspection, but tester perspective inspectors averaged 41 KP uses per inspection. This anomaly is likely due to the format of the artifact. All of the requirements under review were presented as use cases. Therefore, the user perspective inspectors could rely on the presented information without introducing new knowledge. Inspectors using the tester perspective, however, had to rely on perspective-specific knowledge to extract inputs, expected outputs, and additional information from the presented use cases.

The null hypothesis for H3, then, is that there is no difference between the number of domain-specific facts recalled by reviewers with different reported levels of experience. The Spearman's correlation value is significant only for the correlation of KP survey values with KP usage values at the alpha = 0.10 level (Spearman's rho = 0.759, $p < 0.081$). The KW comparison (Spearman's rho = 0.315, $p > 0.543$) and KS comparison (Spearman's rho = -0.121, $p > 0.819$) were not significant. Therefore, we can only statistically validate the hypothesis that perspective-specific experience leads to an increase in the usage of perspective-specific knowledge in PBR requirements inspections.

## 5. Threats to validity

The study described in this report represents a novel, exploratory effort. The primary goal of the results presented throughout Section 4 is to provide justification for the continued use of cognitive science

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

methods in software engineering. In the following discussion of threats to validity, we separate threats due to the protocol analysis method from threats isolated to the current study's context.

### 5.1 Threats due to protocol analysis

Protocol analysis has a number of threats native to the technique that must be controlled for experimental results to be valid.

The coding process must be carried out objectively, with a focus on inter-rater reliability (IRR) and similar metrics that control for objectivity. The current study did not address IRR due to the limited number of subjects and the exploratory nature of the coding scheme itself. However, as replications and future work build on the use of protocol analysis in software engineering, objective coding must be built in from the start.

Secondly, the statistical analysis available to protocol analysis studies is limited by the nature of the categorical data collected. The two analyses used in the current study – Chi Square and Spearman's correlation – are both valid for categorical data. The conclusions drawn from Chi Square depend on an existing model of cognitive activity. Additionally, the correlational underpinnings of these analyses limit the overall interpretation of results.

For the purposes of validity, the categorical data of protocol analysis are also participant variables, meaning they cannot be directly controlled in an experimental context. However, quasi-experimental designs are often used in empirical software engineering due to the varied subject populations. Cognitive factors, because they represent an individual difference, will almost always appear as participant variables.

Finally, the "density" of protocol analysis data threatens the power of statistical results if not scoped appropriately. The overhead of transcribing and coding data generally limits the number of subjects that can be used in a study. If only one data point is used per subject, for example, then the overall power of the study will be limited. However, it is possible to extract a large number of data points from a single subject, as shown with hypothesis H1 in the current study, which used 52 data points over 6 subjects.

Despite the identified limitations of protocol analysis, the technique still has promise. The concerns of objectivity, the need for existing models, and the need for statistical power can all be addressed through refined replication. The limitations of correlational analysis and quasi-experimental design are not uncommon in empirical software engineering. Therefore, we believe that the limitations of protocol analysis certainly do not outweigh the foreseeable benefits to the empirical software engineering field.

### 5.2 Threats due to current context

The current study had a number of limitations that affect the validity of the results, apart from the limitations of the protocol analysis method.

The study only utilized 6 subjects, with an unbalanced number of subjects assigned to each perspective. This threatens the overall power of statistical conclusions, particularly for H2 and H3, which had a low number of data points and subsequent degrees of freedom.

The study also used student subjects in an academic setting, where motivation and generalizability are typical concerns. While a natural requirements artifact was used for a full-lifecycle system, the document was created by a group of students in a capstone course, and not by professionals.

The authors of this research also served as the facilitators of the experiment due to the exploratory nature of the study, including coding activities. This particularly threatens the validity of coding activities, including protocol analysis coding and the a posteriori categorization related to H2. In the future, independent blind coders should be used.

Finally, the data collection periods for the current study were not typical of the PBR experimental environment. Most PBR studies allow for open-ended defect detection on a fixed-length section of the artifact rather than a fixed-length defect detection session that leads to the review of a variable number of requirements. The subjects of the current study were advised that the integrity of think-aloud data was more important to the research than review performance.

Overall, the current study obviously requires robust replication if the current statistical conclusions are to be verified. Because a systematic review was used to identify hypotheses, however, the hypotheses considered have largely been addressed in the existing literature. Future replications can focus on increasing the number of subjects and data points available. In order for industrial replication to be cost effective, an objective coding scheme (as discussed in section 5.1) must also be developed and tested in controlled settings.

## 6. Conclusions and future work

Given the results of Section 4 and the threats of Section 5, we believe that this study allows for a number of qualitative and quantitative conclusions that

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

should readily support future work addressing cognitive factors in software engineering. We describe conclusions related to the protocol analysis method as well as conclusions regarding PBR.

## 6.1 Protocol analysis method

We believe that the development of a coding scheme from cognitive psychology literature supported reliable analysis of the PBR task. The current scheme is in a very generic form. Replications of the current study or of any cognitively intense software engineering task should allow for the refinement of coding schemes. Coding schemes can eventually lead to quantitative cognitive models, even for seemingly complex problem solving tasks.

Protocol analysis was originally developed for the study of expertise, and not just discovery of cognitive processes. As software engineering and particularly software quality assurance methods advance, expertise will play an increasing role in research and practical scenarios. With an effective coding scheme, protocol analysis has the potential to describe expertise. Particularly in the ESE field, capturing and accelerating expertise is an important high-level goal that protocol analysis is designed to support.

Future work should also focus on limiting attrition of subjects during protocol collection, and as mentioned in Section 5.1, objectivity should be a goal in the development of future coding schemes. Practical experience with protocol analysis data collection can help resolve issues with both data collection and coding schemes. For this reason, a pilot study is also recommended for future efforts.

## 6.2 Perspective-based reading

The new approach presented in this study surpasses traditional high-level, "black box" studies in its ability to understand the cognitive mechanisms supporting PBR. While effectiveness and efficiency are very critical real-world goals, understanding the underlying cognitive factors can potentially unify existing work in the field, as shown by the diverse hypotheses considered and statistically validated in the current study. Additionally, a cognitive framework can offer new insight to the technique, allowing for it to advance at a practical level.

Future work with PBR and protocol analysis should not lose sight of PBR's long line of empirical validation in industrial settings. Traditionally, ESE studies performed in industry are limited, but the non-invasive think-aloud protocol could offer a link between the ease of qualitative data collection and the robustness of quantitative support.

The statistical claims of the current research appear to support at least three separate hypotheses from literature, as shown in Section 4.2 The notion of defect triggers [4] seems directly linked to the cognitive processes employed during inspection. With analysis of the CK and WD codes, the defect trigger proposed by Chaar, et al., was statistically validated by our study. The analysis also leveraged the 52 data points that were distributed across all 6 reviewers, adding to the power of the statistical conclusion.

H2 addressed a fundamental claim of PBR and its parent family of SBR. This issue had been addressed previously by Regnell, Runeson, and Thelin [13] as well as Maldonado, et al. [10], Porter and Votta [11], and others. The defects found by applying a scenario should be related to the scenario (and for PBR, the perspective) assigned. We proposed a new a posteriori method of classifying all defects based on a qualitative evaluation of which perspective is most likely to find each defect, given a detailed understanding of the foci of the perspectives. Then, the analysis considers whether reviewers find more defects of the type associated with their perspective or scenario. A Chi Square test for independence can then be used to determine if the type of defects found by each perspective is significantly different. In our case, the difference between perspectives was very significant ($p < 0.01$). The same analysis, because it is applied after the data has been collected, could be used on existing data from other PBR and SBR studies to consider the same hypothesis or others regarding the nature of defects found.

Finally, our work supports Carver's hypothesis that perspective-specific experience significantly affects PBR inspections [3]. Specifically, we found that perspective experience is brought into STM from LTM during the PBR task. Also, reviewers with higher levels of perspective-specific experience use significantly more perspective-specific knowledge, as indicated by the KP code. For the artifact used, the three types of domain knowledge considered did not have the same significant correlation with cognitive activities of the PBR task, as indicated by the KW and KS codes. The subject population did not have differing levels of experience with the conference submission and review process, which disallowed a consideration of conference submission experience with the KC code. We propose that future work could consider the role of domain knowledge in more specific domains, with subject populations possessing differing levels of experience. This approach should also be more comparable to industry teams, when team

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

members possess differing levels of domain-specific knowledge.

## 7. Acknowledgments

We wish to thank the members of the Empirical Software Engineering Group at Mississippi State University. We also wish to thank Dr. Gary Bradshaw of the Mississippi State University Psychology department for aid in developing and evaluating the coding scheme. Finally, we thank the students who participated in the study, and their instructor, Dr. Thomas Philip.

## 8. References

[1] R.C. Atkinson and R.M. Shiffrin, "Human Memory: A Proposed System and its Control Processes", *Psychology of Learning and Motivation II*, Edited by K. W. Spence and J. T. Spence, Academic Press, New York, NY, 1968, 89-195.

[2] V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S. Sørumgård, and M.V. Zelkowitz, "The Empirical Investigation of Perspective-Based Reading", *Empirical Software Engineering*. 1(2), Springer, 1996, pp.133-164.

[3] Carver, J.C., *The Impact of Background and Experience on Software Inspections*, Doctoral Thesis, UMI Order Number: AAI3094462., University of Maryland at College Park, 2003.

[4] J.K. Chaar, M.J. Halliday, I.S. Bhandari, and R. Chillarege, "In-Process Evaluation for Software Inspection and Test", IEEE Transactions on Software Engineering, 19(11), 1993, pp. 1055-1070.

[5] Ericsson, K. A. and H. A. Simon, *Verbal Reports as Data,* MIT Press, Cambridge, MA, 1993.

[6] L. He and J. Carver, "PBR vs. Checklist: A Replication in the N-Fold Inspection Context", in *Proceedings of the 2006 ACM/IEEE international Symposium on Empirical Software Engineering* (Rio de Janeiro, Brazil, September 21 - 22, 2006). ACM, New York, NY, 2006, pp. 95-104.

[7] D. C. Howell, *Statistical Methods for Psychology*, 5th Edition, Thomson/Wadsworth, Toronto, CA, 2005.

[8] B. A. Kitchenham, T. Dybå, and M. Jørgensen, "Evidence-Based Software Engineering", in *Proceedings of the 26th International Conference on Software Engineering* (Edinburgh, Scotland, May 23-28 2004), IEEE, 2004, pp. 273-281.

[9] O. Laitenberger, K. El Emam, and T.G. Harbich, "An Internally Replicated Quasi-Experimental Comparison of Checklist and Perspective-Based Reading of Code Documents", *IEEE Transactions on Software Engineering*, 27(5), IEEE, 2001, pp. 387-421.

[10] J.C. Maldonado, J. Carver, F. Shull, S. Fabbri, Dória, E., L. Martimiano, M. Mendonça, and V. Basili, "Perspective-Based Reading: A Replicated Experiment Focused on Individual Reviewer Effectiveness", *Empirical Software Engineering*, 11(1), Springer, 2006, pp. 119-142.

[11] A. Porter and L.G. Votta, "An Experiment to Assess Different Defect Detection Methods for Software Requirements Inspections", in *Proceedings of the 16th International Conference on Software Engineering*, (Sorrento, Italy, May 16-21, 1994), IEEE, 1994, pp 103-112.

[12] A.A. Porter, L.G. Votta, and V.R. Basili, "Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment", IEEE Transactions on Software Engineering, 21(6), IEEE, 1995, pp. 563-575.

[13] B. Regnell, P. Runeson, and T. Thelin, "Are the Perspectives Really Different? – Further Experimentation on Scenario-Based Reading of Requirements", *Empirical Software Engineering*, 5(4), Springer, 2000, pp. 331-356.

[14] F. Shull, I. Rus, and V. Basili, "How Perspective-Based Reading Can Improve Requirements Inspections", *Computer*, 33(7), IEEE, pp. 73-79.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE