Effectiveness of Human Error Taxonomy during Requirements Inspection: An Empirical Investigation

Vaibhav Anu, Gursimran Walia
Department of Computer Science
North Dakota State University
Fargo, USA
vaibhav.anu@ndsu.edu,
gursimran.walia@ndsu.edu

Wenhua Hu, Jeffrey C. Carver
Department of Computer Science
University Of Alabama
Tuscaloosa, USA
wh10@crimson.ua.edu,
carver@crimson.ua.edu

Gary Bradshaw
Department of Psychology
Mississippi State University
Mississippi State, USA
glb2@psychology.msstate.edu

***Abstract*—Software inspections are an effective method for achieving high quality software. We hypothesize that inspections focused on identifying errors (i.e., root cause of faults) are better at finding requirements faults when compared to inspection methods that rely on checklists created using lessons-learned from historical fault-data. Our previous work verified that, error based inspections guided by an initial requirements errors taxonomy (RET) performed significantly better than standard fault-based inspections. However, RET lacked an underlying human information processing model grounded in Cognitive Psychology research. The current research reports results from a systematic literature review (SLR) of Software Engineering and Cognitive Science literature - Human Error Taxonomy (HET) that contains requirements phase human errors. The major contribution of this paper is a report of control group study that compared the fault detection effectiveness and usefulness of HET with the previously validated RET. Results of this study show that subjects using HET were not only more effective at detecting faults, but they found faults faster. Post-hoc analysis of HET also revealed meaningful insights into the most commonly occurring human errors at different points during requirements development. The results provide motivation and feedback for further refining HET and creating formal inspection tools based on HET.**

Keywords-human error; requirements inspection; taxonomy; empirical study

I. INTRODUCTION

Software engineers spend around 80% of total development time during testing and debugging [13], a majority of which is spent on fixing faults that were committed during the early phases (e.g., requirements development). To address this problem, project managers focus on *software inspections* (i.e., reviewing requirements and design documents to identify faults) when they are easiest and cheapest to find and fix. Empirical evidence reports that, inspections at multiple organizations (e.g., IBM, TRW, ICL, Cardiac pacemakers) have led to significant improvements in quality, reduction in fault rate and delivering products within allocated time and cost [14].

In spite of this wide spread success, inspection techniques focus developers' attention only on different type of *faults* (e.g., missing or incorrect functionality) recorded in software artifacts. As a result, they cannot help inspectors detect all the problems without understanding underlying *errors* (i.e., source of faults). Therefore, we believe that, an inspection process that can help developers' focus on the cause of the problems (as opposed to just the symptoms of errors - *faults*) will be a significant improvement over standard fault-checklist based inspections. To that end, our prior research [6] has gathered initial evidence to show that structured *error* information help inspectors detect significantly large number of faults (that are otherwise overlooked or left undetected) as compared to *fault-checklist (FC)* based inspections. Even when comparing against the most advanced fault inspection technique (*Perspective based Reading* – PBR [6]), error based inspection performed better. A brief summary of prior research and motivation for the current research is provided in next paragraphs.

Prior Research: To help evaluate the feasibility of an error-based inspection, Walia and Carver (co-authors), created a preliminary classification of requirement errors by classifying errors reported in published literature into three main error types: *People*, *Process* and *Documentation* Errors. Next, a series of controlled experiments [3-5] validated that, an error-inspection (guided with RET) significantly improves the inspectors' effectiveness when compared to fault-based inspections. It was also reported that, educating developers on *errors* (as opposed to *faults*) made them less likely to make mistakes during the requirements development [6].

Motivation: While RET was found effective and useful, it was not meant to be complete and final product. A major drawback of RET was a lack of *human error* research on normal psychological processes that produce errors. A need for deeper investigation of human cognition failures was also highlighted by RET study results that reported *People Errors* (due to fallibilities of people involved) as a source of significantly large portion (i.e., up to 32%) of faults. These results motivated us to conduct an extensive psychological research on *human errors* in order to develop a human error taxonomy (HET). The resulting taxonomy (HET) is deeper (multi-level) and structured based on the theoretical human error types proposed by Reason's well-respected taxonomy of human errors [10]. Details of HET (Figure 1) appear in Section II.

Contribution: The current paper investigates whether HET can further enhance the inspection performance (as compared to RET) during an error based inspection. To answer this question, a control group experiment compared the fault detection *effectiveness* (# of faults), *efficiency* (faults/hour), and *usefulness* (subjects' feedback) of HET (Figure 1) vs. RET [2] during the inspection of SRS documents. In addition, the study provided insights into the nature of commonly occurring human errors at

DOI reference number: 10.18293/SEKE2016-177.

different points during the requirement development process. More details appear in Sections III-V.

II. Human Error Taxonomy

Human error taxonomy (HET) was developed based on the systematic literature review (SLR), and in direct interaction with human error experts (via human error workshop – WAHESE 2015; http://humanerrorinse.org/workshops). To enable the contribution from Cognitive Science perspective of human errors, HET was created under the guidance of a Cognitive Psychologist (who is also a co-author on this paper). As a result, Reason's [10] human error classification system (which has also been adapted in other domains) of ***slips***, ***lapses***, and ***mistakes*** was used to classify requirement phase human errors. When faced with a situation that requires problem solving, humans perform two cognitive activities: planning and execution. Reason [10] calls the cognitive failures (or human errors) that occur during planning, ***mistakes***, and the cognitive failures that occur during execution, ***slips*** and ***lapses***.

Reason's slips are results of inattention while executing routine tasks. Slips are exemplified in common day to day activities like typing something incorrectly or "fat-fingering" due to not paying attention. Lapses occur when executing well-planned tasks, but are failures of memory. For example, having planned to repair a broken machine-part, but forgetting it due to interruption (e.g., taking a lunch break) is a common lapse. Mistakes are planning failures and occur when trying to solve an unfamiliar problem. An example of a mistake is a physician misdiagnosing a patient due to either not properly studying this patient's symptoms or having no experience whatsoever with the symptom's exhibited by this specific patient. In order to aid reader's understanding, detailed information and examples of slips, lapses, and mistakes are provided in [7].

After we selected the right human error classification system (i.e. Reason's slips, lapses, and mistake), the software engineering literature was surveyed to identify requirement phase human errors. While selecting requirements phase human errors, we strictly adhered to the following definition of human errors: *a human error is a flaw in human thought process*. For example, misunderstanding user's need is an *error*, while writing incorrect requirements specification due to that misunderstanding is a *fault*. A total of 15 requirements phase human errors identified from Software Engineering literature were classified as a slip, a lapse, or a mistake. The resulting taxonomy is shown in Figure 1 with details in [7].

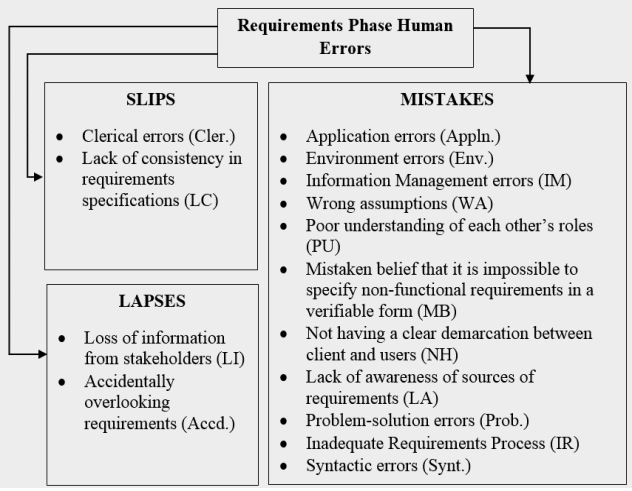


Figure 1. The Human Error Taxonomy (HET)

III. Experiment Design

The main goal of this experiment was to understand if changes made to the error taxonomy (i.e., development of HET) offer an improvement over RET (a proven verification technique) during the requirements inspection. To do so, a randomized pre-test post-test control group experiment was planned and executed in controlled settings. The control group used RET [2], whereas the experimental group used the newly developed HET to perform requirements inspection.

A. Experiment Methodology

This section describes research questions (RQ's), study variables, and artifacts with complete package available at http://humanerrorinse.org/Studies/2015/Fall_NDSU_Experiment_1/index.htm.

1) RQ's: The following four RQ's were investigated.

RQ 1: Which error taxonomy (HET vs RET) provides better fault detection effectiveness (# of faults found) and efficiency (faults/time) during the requirements inspection?

RQ 2: Which error taxonomy (HET vs RET) is perceived more useful for finding faults during the inspection?

RQ 3: What insights into the major type of *human errors* can be used to further enhance the inspection performance?

RQ 4: Can an inspector's performance on a pre-test accurately predict their inspection performance on a real project?

2) Variables:

Table I provides study variables and their descriptions.

Table I. Study Variables

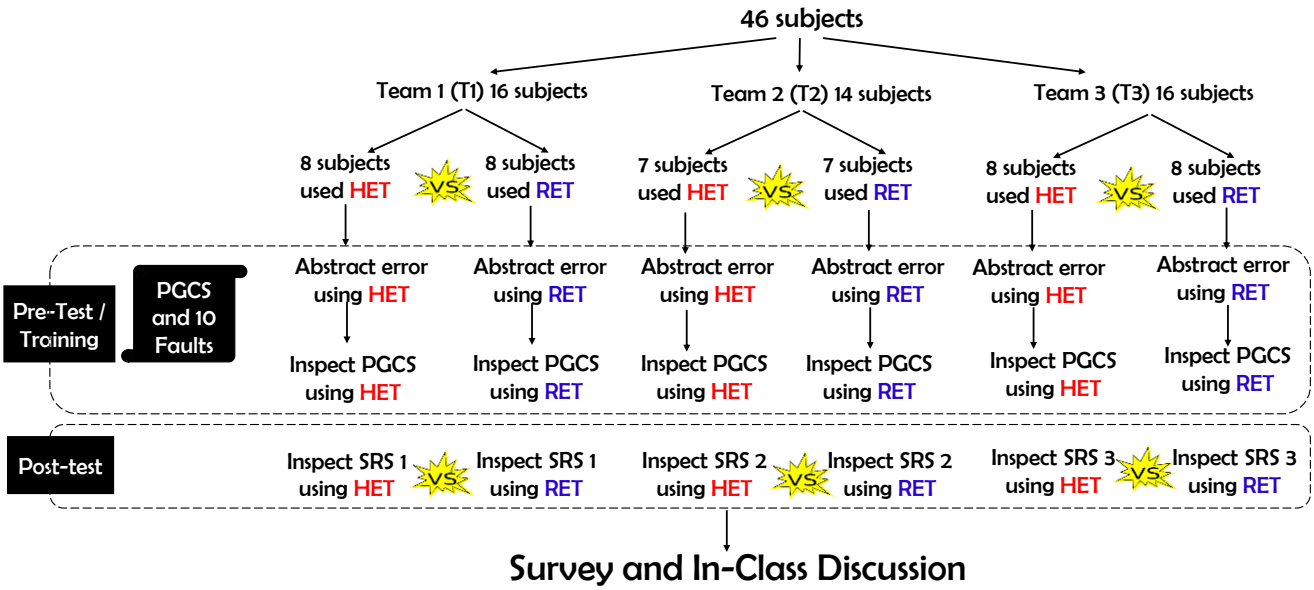
Independent Variables	Description
Pre-test	measures performance of subjects in a practice inspection exercise using HET/RET.
Effort Spent	time spent by the individual subjects to perform the inspection tasks.
Usefulness	perceived usefulness of HET/RET
Dependent Variables	**Description**
Effectiveness	Number of faults found by each subject
Efficiency	Number of faults found per hour

3) Subjects:

46 computer science students, enrolled in *Principles of Software Engineering* course at North Dakota State University (NDSU) participated in this study. The course required students to work in teams (teams were selected by the instructor prior to this study) to develop requirement artifacts for different software systems. To enable a comparison between HET vs. RET, researchers randomly divided subjects in each team into two equal groups (a *control group* that used RET and an *experiment group* that used HET). Figure 2 shows the division of subjects into three teams (e.g., team 1 had 16 subjects) and subdivision of each team into treatment groups (8 used RET and 8 HET).

4) Artifacts:

During the training (*pre-test*), participating subjects (23 in experimental group and 23 in control group) were trained on



their respective taxonomies (HET for experiment and RET for control group) by having them perform an error based inspection of an externally produced SRS document that was seeded with 30 realistic faults. The SRS used during the training described requirements for a Parking Garage Control System (PGCS).

During the *post-test*, subjects inspected the SRS's that they had developed (as part of a team) during the course of the semester. Table II provides a brief system description for the SRS's created by each team. As mentioned earlier and shown in Figure 2, half of the subjects within each team inspected their own SRS using HET (e.g., 8 in team 1) or RET (other 8 in team 1), depending on the treatment group they were assigned (and trained) during the pre-test.

TABLE II. TEAMS AND SYSTEM DESCRIPTIONS

Team#	System Name and Description
1	Fly-by: airline reservation and travel management platform
2	Campus Reconnection: college student information management and course management system.
3	FaceSpace: music streaming based on analysis of music played by user and other user-generated metrics.

B. *Experiment Procedure*

Figure 2 shows the comparison between the control group (RET) and experimental group (HET) at ***pre*** and ***post***-test:

1) *Pre-test Steps: Training on HET and RET*

During the ***Pre-test***, subjects were trained on HET and RET by having them perform an error-inspection on PGCS SRS document and report errors and faults. Details appear below:

Training: Subjects were trained on the importance of requirements inspections and different type of requirement faults. Next, in two separate sessions, 23 subjects were trained on HET and 23 subjects were trained on RET to teach them about the *error abstraction* using HET/RET (i.e., how to identify errors), and using the error information to perform *fault inspection* (i.e., how to identify new faults).

Error Abstraction - After the training, subjects were provided with 10 PGCS SRS faults (chosen randomly from 30 seeded faults). Subjects then used HET/RET to abstract and classify errors from 10 given faults. This step resulted in 46 error forms (23 for RET and 23 for HET).

Fault Inspection – Next, subjects used the abstracted error information (from error forms) to find additional faults in the PGCS SRS (i.e. subjects inspected PGCS SRS using errors). This step resulted in 46 individual fault-forms (23 for HET and 23 for RET) containing new faults in PGCS document.

2) *SRS Development (listed in Table II)*

Subjects then worked in their respective teams (three teams) to develop requirements for different systems.

3) *Post-test Steps: Error-Inspection on self-created SRS*

During the ***post-test***, every subject inspected their own SRS (they had developed as a team) using the technique that were trained during the pre-test (HET or RET) and reported faults. For example, of 16 subjects in Team 1, 8 used HET while the other 8 used RET to inspect the "*Fly-by*" SRS. This step produced individual fault-forms from HET and RET for each team.

Post-study Survey and Focus Group: The experimental group and the control group subjects rated HET and RET across various usefulness categories on a 5-point scale (ranging from "1 – not useful" to "5 – very useful"). A focus group discussion was conducted with the goal of understanding the problems faced by subjects while using HET/RET to find faults.

IV. DATA ANALYSIS AND RESULTS

The results from analysis of data collected during the study run and organized around four RQ's listed in III.A.

A. *Fault Detection Effectiveness of HET vs RET (RQ 1a):*

We compared the fault detection effectiveness of the experimental group subjects (who used HET) vs. control group subjects (who used RET), during the inspection of their self-created SRS documents. The fault reporting forms during the post-test were analyzed separately for each team (after removing

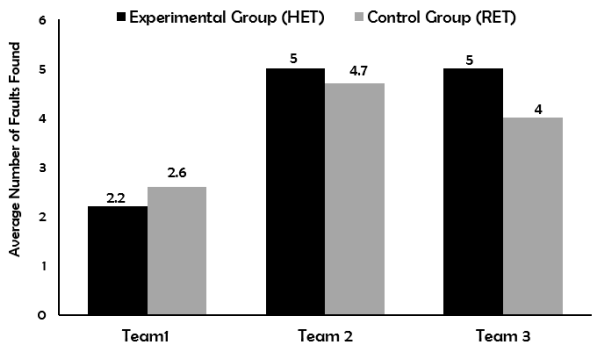


Figure 3. Comparison of average number of faults

the fault positives) to enable this comparison. Figure 3 reports the resulting comparison of the average number of faults detected by the experimental group subjects' vs the average number of faults detected by the control group subjects within each team. For example, for Team 3, experimental group subjects found an average of five (5) faults and the control group subjects found an average of four (4) faults.

For teams 2 and 3, as shown in Figure 3, the experimental group subjects generally found more faults than the control group subjects, however these results were not significant (based on an independent samples t-test). The inconsistency with Team 1's results was analyzed and we found that many subjects (6 out of 16) from Team 1 did not participate in this experimental task (didn't inspect using their assigned technique). Therefore, low fault count for Team 1 (irrespective of HET/RET) was found to be due to lack of participation and understanding of the tasks involved.

The results show that, when properly motivated, HET can provide better fault detection effectiveness as compared to RET.

B. Fault detection efficiency of HET vs RET (RQ 1b):

We also compared the *efficiency* (faults per hour) of the experimental and control group subjects within each team. The timing data (start and end times, the time each fault was fault, breaks) reported by subjects in their fault forms was used to calculate the fault rate. The *average fault rate* comparison of treatment groups is shown in Figure 4.

As shown in Figure 4, subjects using HET (experiment group), found faults at much faster rate when compared to the subjects who used RET (control group). The results from an independent samples t-test showed that although subjects using HET were consistently more efficient for all three teams, the improvement was not significant.

This was the very first investigation of HET whereas RET has been validated and improved through a series of comprehensive empirical analysis [3-5]. That said, subjects using the HET not only found more faults, they found it at a much faster rate. This also shows that, more effort (time spent) is needed to use RET to match the productivity that can be achieved with HET. That is, the learning curve for HET is smaller than that of RET, which justifies the motivation for the development of HET.

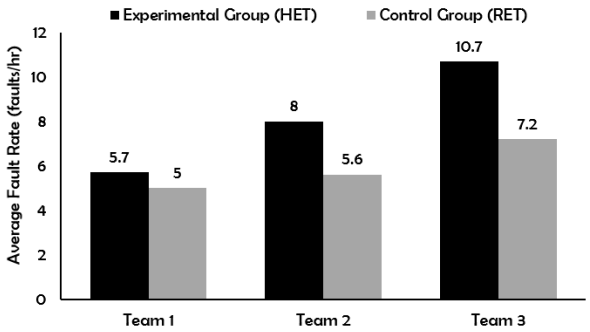


Figure 4. Comparison of average fault rate or efficiency (faults/hour)

C. Perceived Usefulness of the Two Error Taxonomies (RQ 2)

Next, we evaluated subjects' self-reported data (collected during post-study survey) regarding the *usefulness* of HET and RET on five essential attributes: (1) *usability- usab.*, (2) *orthogonality* (*Orth.*- a lack of overlap amongst error classes, (3) *usefulness* of error taxonomy in locating faults – *Usef.*, (4) *confidence* that taxonomy represented real RE problems – *Conf.*, and (5) *worthiness* of the effort spent in using the taxonomy – *Worth.*. A 5-point Likert scale ranging from 1-*Strongly Disagree* to 5- *Strongly Agree* was used to evaluate these attributes of both taxonomies. This comparison was performed for 23 subjects who used (and rated) HET vs. 23 subjects who used RET.

TABLE III. HET VS. RET COMPARISON USING 5-POINT SCALE

	Usab.	Orth.	Usef.	Conf.	Worth.
HET	2.94	2.94	3.89	3.89	3.33
RET	2.68	3.0	3.42	3.79	3.21

Table III compares of average ratings of HET and RET, across five attributes. Overall, HET was rated more positively (greater than 3 – midpoint on a 5-point scale) than RET in terms of *usefulness*, *confidence* and *worthiness* aspects of error based inspections. The shaded cells in Table III for HET, were rated significantly greater than the midpoint of scale ($p<0.001$) based on the results from one-sample t-test. However, RET received better feedback rating for the attribute, *orthogonality of error classes*. This was expected because unlike RET, HET includes errors within an error type (e.g., *Application error* – an error class under *Mistake* in Figure 1) that can happen at different points during the requirement development (i.e., elicitation, analysis, and verification). We plan to explain this in more detail during the training (in future studies) and also plan to provide more examples of errors (in HET training document) to make it more easy to use in future.

D. Insights into Human Errors during Requirements Development (RQ 3).

While human error research seems worthwhile (based on these results), we wanted to gain useful insights into the major sources of requirement faults. This in turn can help researchers (and practitioners) to focus on more frequently occurring human errors and reduce their frequency through interventions (e.g., a checklist or a tool-based assist). To perform this analysis, faults

and errors reported by the subjects (using HET) during the post-test (inspection of self-created SRS's) were used.

Figure 5 reports the results of this analysis in terms of the percentage contribution of *Slips*, *Lapses* and *Mistakes* to the overall faults reported by each team while inspecting their SRS. Results from Chi-square test showed that, for all three teams the observed contribution of error types (slips, lapses, and mistakes) were significantly different ($p<0.001$) from uniform distribution (33.33%). We also analyzed the contribution of each of 15 error classes within HET (Figure 1) to evaluate the most common types of *slips* and *mistakes* that the developers can be made aware prior to the development. The resulting contributions of each of 15 error classes (towards total reported faults) is shown in Table IV (highlighted fields reported higher contributions).

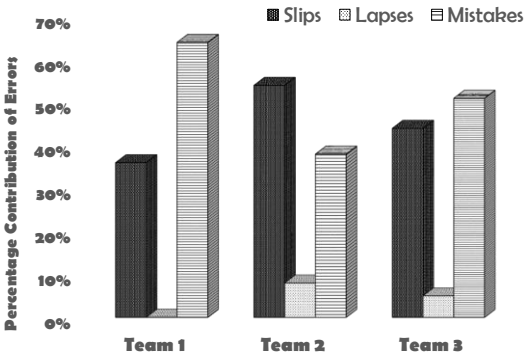


Figure 5. Percentage Contribution of Slips, Lapses and Mistakes

Major insights gained from this analysis is follows:

- Based on these results, *Slips* and *Mistakes* (as opposed to Lapses) are a major source of requirement faults.
- *Clerical Errors* (a sub class of Slips) and *Application errors* (a sub class of Mistakes) occurred at a higher frequency as compared to other error classes.
- *Clerical* errors occurred (reported retrospectively by the subjects) during the elicitation phase. *Application* errors, on the other hand occurred at different points, due to the misunderstanding of particular aspects of problem solution, and led to omission of relevant information from requirements document.

These results are different from the human error results reported in Cognitive Psychology (CP). CP literature reports *Slips* and *Lapses* contributing up to 60% of total errors [15], whereas the results from this study showed that *Slips* and *Mistakes* contributed around 90% of total faults (Team 1, did not report any lapses). A possibility could be that, since subjects are evaluating their own work, they might be less likely to report *lapses* (i.e., memory related failures). We plan to conduct multiple studies to be able to generalize the findings.

E. Prediction of Performance using Pre-test Data (RQ 4).

The performance of subjects during the ***pre-test*** (# of faults found during the PGCS inspection using HET/RET) was correlated with their performance during the post-test (i.e., # of faults found in their own SRS using HET/RET). The goal was to analyze whether the performance of subjects using HET (or RET) while inspecting their own SRS (i.e., post-test) could be predicted by their performance of respective taxonomies during the training. To perform this analysis, the number of actual faults found by subjects in the experiment (HET) and control (RET) group were compared at pre (PGCS SRS) and posttest (self-created SRS). Figure 6 shows the correlation between pre and posttest performance for the experimental and control groups.

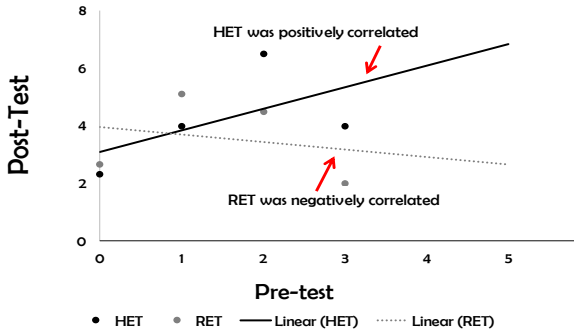


Figure 6. Pre-test vs post-test performance

Based on these result, the experimental group (HET), displayed a positive correlation between pre-and post-test performance whereas control group (RET) subjects showed a negative correlation between pre and post-test performance. This means that when using HET (rather than RET), project managers can help predict the inspectors' performance during the real project. This is objective information for project managers to help plan the inspection process.

V. VALIDITY THREATS

The authors tried to address some of the validity threats. The *selection* threat was reduced by randomizing the experimental group and control group subject placement. This also ensured that treatment groups were equivalent. The threat to *external* validity when using a toy requirements document was also removed by the fact that students inspected real requirements specification documents that they developed and contained naturally occurring faults. However, a threat remains that there might be additional faults (that were not reported by subjects) present in SRSs and we plan to perform additional analysis on

TABLE IV. CONTRIBUTION OF ERROR CLASSES TO TOTAL HUMAN ERRORS WHICH LED TO FAULTS

	Slips		**Lapses**		**----------Mistakes----------**										
	Clerical	LC	LI	Accd.	Appl.	Env.	IM	WA	PU	MB	NH	LA	Prob.	IR	Synt.
Team 1	36%				9%			27%					9%	18%	
Team 2	54%			8%	19%		4%		4%				12%		
Team 3	41%	3%		5%	10%	8%	3%	8%	5%					8%	10%

these SRSs. Also, the study was conducted in a classroom setting that is not representative of time and pressure in real settings. We plan to address this threat in future studies.

VI. Discussion Of Results

In this section, each original research question is revisited to discuss the implications of the results of data analysis.

RQ1: In terms of *effectiveness*, the subjects using HET found more faults compared to the subjects who used RET for two out of three teams. In terms of *efficiency*, subjects using HET across all three teams found faults at a much faster rate when compared to the subjects using RET. A major goal of this study was to evaluate if students' learning curve on identifying human errors and corresponding faults can be enhanced (from RET), and the *efficiency* results justifies the construction of HET. From project managers' perspective, an *efficient* inspection approach allows them to improve software quality without increasing overall projects costs.

RQ2: The results showed that while both error taxonomies were rated favorably, HET received slightly better feedback in four out of the five categories. Subjective feedback given by the experimental group subjects also shed light on the fact that NH and LA error classes under the *mistakes* (NH - *not having a clear demarcation between clients and users*; and LA - *lack of awareness of sources of requirements*) were hard to distinguish and need to be either merged together or supplied with more examples. Based on their subjects' feedback, the process of using error information to find faults can be formalized to further improve the usefulness of HET.

RQ3: The results showed that Slips and Mistakes occurred at a higher frequency as compared to Lapses. Further, *Clerical* errors and *Application* errors were most commonly occurring slips and mistakes respectively. Additional investigations are needed to evaluate if this lack of occurrence of *lapses* can be generalized to all software systems.

RQ4: The results showed that when using HET, an inspector's performance during the training can help predict their performance during an inspection on live project. This can help project managers to staff inspectors and better plan inspection process at their organizations.

VII. Conclusion And Future Work

This study reported the newly developed Human Error Taxonomy (HET) and compared it against the seasoned Requirement Error Taxonomy (RET), with regards to the fault detection effectiveness and efficiency of requirement inspectors when using HET vs RET. The overall results show that HET not only helped inspectors find more faults, but also at a faster rate. This encourages further research into the usage of human error abstraction and classification (using HET) for requirements defect detection. Subjects, in their feedback reported the need for a more systematic process of using errors to find faults. The authors plan to do more studies to collect extensive data set on human errors and its impact on the requirement. We also intend to develop a tool that will assist the inspectors during the error abstraction process. The tool will be developed to help inspectors abstract human errors from the perspective of - "*which requirement activity (elicitation, analysis, specification, verification, or management) did the human error occur in?*"

The authors expect a significant improvement in HET-based inspection approach with the inclusion of such aforementioned tools.

Acknowledgment

This study was supported by National Science Foundation Awards 1423279 and 1421006. The authors would like to thank the participating students enrolled at NDSU and the course instructor for their help during the study run.

References

[1] F. Lanubile, F. Shull, and V. R. Basili, "Experimenting with error abstraction in requirements documents," in 5th International Symposium on Software Metrics, Bethesda, 1998.

[2] G. S. Walia and J. C. Carver, "A systematic literature review to identify and classify software requirement errors," Inf. Softw. Technol., vol. 51, no. 7, pp. 1087–1109, 2009.

[3] G. S. Walia, J. C. Carver, and P. Thomas, "Requirement Error Abstraction and Classification: An Empirical Study," in 5th International Symposium on Empirical Software Engineering, New York, 2006, pp. 336–345.

[4] G. S. Walia, J. C. Carver, and T. Philip, "Requirement error abstraction and classification: A control group replicated study," in Proceedings - International Symposium on Software Reliability Engineering, ISSRE, Sweden, 2007, pp. 71–80.

[5] G. S. Walia and J. C. Carver, "Evaluating the use of requirement error abstraction and classification method for preventing errors during artifact creation: A feasibility study," in Proceedings of International Symposium on Software Reliability Engineering, ISSRE, San Jose, 2010, pp. 81–90.

[6] G. S. Walia and J. C. Carver, "Using error abstraction and classification to improve requirement quality: Conclusions from a family of four empirical studies," Empir. Softw. Eng., vol. 18, no. 4, pp. 625–658, 2013.

[7] V. K. Anu, G. S. Walia, W. Hu, J. C. Carver, and G. Bradshaw, "Usefulness of Human Error Taxonomy as an Effective Requirements Inspection Technique: An Empirical Investigation," TECHNICAL REPORT, Fargo, ND, 2015,
http://humanerrorinse.org/Studies/2015/Fall_NDSU_Experiment_1/index.htm

[8] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, B. K. Ray, and D. S. Moebus, "Orthogonal Defect ClassificationA Concept for In-Process Measurements," IEEE Trans. Softw. Eng., vol. 18, no. 11, pp. 943–956, 1992.

[9] M. Leszak, D. E. Perry, and D. Stoll, "A case study in root cause defect analysis," in Proceedings of the 22nd international conference on Software engineering - ICSE, Limerick, Ireland, 2000, pp. 428–437.

[10] J. Reason, Human error. Cambridge, U.K.: Cambridge University Press, 1990.

[11] S. A. Shappell and D. A. Wiegmann, "Applying Reason: the human factors analysis and classification system (HFACS)," Hum. Factors Aerosp. Saf., vol. 1, no. 1, pp. 59–86, 2001.

[12] D. Wiegmann and C. Detwiler, "Human Error and General Aviation Accidents : A Comprehensive , Fine-Grained Analysis Using HFACS," Security, pp. 1–5, 2005.

[13] B. Boehm and V. R. Basili, "Software Defect Reduction Top 10," Computer, vol. 34, no. 1, pp. 135–137, 2001.

[14] O. Laitenberger, "A Survey on Software Inspection Technologies," in Handbook on Software Engineering and Knowledge Engineering, vol. 2, 2002, pp. 517–555.

[15] A. Esgate, D. Groome, and K. Baker, An Introduction to Applied Cognitive Psychology. Psychology Press, 2005.