Early Software Product Improvement with Sequential Inspection Sessions: An empirical Investigation of Inspector Capability and Learning Effects

Dietmar Winkler, Bettina Thurnher, Stefan Biffl

Institute of Software Technology and Interactive Systems
Vienna University of Technology

dietmar.winkler@qse.ifss.tuwien.ac.at
http://qse.ifss.tuwien.ac.at
Motivation

- The construction of high-quality software products requires (a) professional approaches (e.g., processes and methods) and well-trained engineers.
- Early detection and removal of defects, e.g., in the design phase, helps increase software quality and decrease rework effort and cost.
- Prior empirical studies showed that UBR (software inspection with a usage-based reading technique approach) can focus on most important use cases and spot on the detection of crucial and important defects.
- Inspection promises to be a vehicle to support learning.

Questions:
- How is the impact of inspector qualification on inspection performance?
- Is there any notable difference of learning effects regarding inspection performance in a sequence of sessions?
Defect Detection with Inspection

- Software Inspection …
  - is a **static analysis technique** to verify quality properties of software.
  - does not require executable code (applicable to design documents).
  - focuses on **defect types and location** in the inspected object.
  - Guidance of inspectors with reading techniques and guidelines (how to traverse a software document).

- “Best-practice” approach: **Usage-Based Reading (UBR)**
  - Well-investigated reading technique approach.
  - Goal: **focus on most important defects** first (classes “crucial” and “important”).
  - **User focus**: use cases lead the inspection process.
  - **Application of use cases and scenarios** from requirements documents in a pre-defined order (prioritized by a group of experts) to design documents.
Learning with Inspection

- Inspection supports learning due to
  - a *systematic and structured process* (inspection process) which is repeatable and traceable
  - *Active guidance* to support individual inspectors in defect detection tasks (guidelines, checklists, etc.)

- We refer “learning” as an *improvement of individual inspection performance* in a sequence of inspection sessions within a similar application domain.

- Research Questions:
  - Is there any difference of inspection performance regarding system complexity and inspector capability?
  - Can we identify differences of gained additional inspection experience in a second inspection session?
Dependent Variables and Hypothesis

- **Inspection effort** includes individual preparation time and inspection duration (we did not consider inspection pre-work, e.g. use case prioritization).
- **Effectiveness** is the number of defects in relation to the overall number of seeded (important) defects.
- **Efficiency** is the number of defects found per time interval (e.g., defects found per hour)
- **False Positives** is the share of "wrong defects found" by individual inspectors.

Hypothesis:
- Effectiveness and efficiency will **increase** in a second inspection session.
- False positives will **decrease** in a second inspection session.
- Inspectors will perform **better in the less complex part** of the system.
Experiment Description

- The system represents a snapshot of the development process of a taxi management system including requirements and design documents and source code fragments.

- Two parts of the system

  ![Diagram showing communication link between Taxi, Driver, Taximodule, Communication Link, Central, Operator, and Central](attachment:diagram.png)

- at different levels of system complexity (amount of inspection material). Complexity (Central) > Complexity (Taxi).

- Total number of **56 seeded important defects** within the design specification and the source code.

- Three experiment phases processed:
  (a) training & preparation, (b) individual inspection, and (c) data submission.
Study Arrangement (2x2 study design)

- **Subjects**
  - 104 graduate students in a class on quality assurance and software engineering: 18 less, 22 medium and 12 higher-qualified inspectors per group.
  - The subjects were randomly assigned to 2 groups.

- Data Set 1 (Central part) is the more complex part and Data Set 2 (Taxi) is the less complex part of the system.
Threats to Validity

Internal validity:
- Avoidance of communication between individuals during the study execution.
- Participants could take individual brakes, whenever necessary (break durations noted).
- Limitation of the overall inspection duration (3h for taxi, 5h for central due complexity reasons).
- Experience questionnaire to get an insight on prior knowledge.
- Feedback questionnaire to see if the participants followed the study process properly.

External validity:
- Well-known application domain to avoid domain-specific interpretation problems.
- Pilot test and reviews to assure correctness of experiment material.
- Control of variables due classroom setting.
Results: Effectiveness

- **Effectiveness**: number of defects found in relation to the number of seeded defects.

<table>
<thead>
<tr>
<th></th>
<th>Session 1</th>
<th></th>
<th>Session 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Central</td>
<td>Taxi</td>
<td>Central</td>
<td>Taxi</td>
</tr>
<tr>
<td>Mean</td>
<td>34.2%</td>
<td>51.0%</td>
<td>55.7%</td>
<td>74.6%</td>
</tr>
<tr>
<td>Std.Dev.</td>
<td>17.2%</td>
<td>25.7%</td>
<td>5.5%</td>
<td>11.6%</td>
</tr>
</tbody>
</table>

- **System Complexity**:
  - Significant differences between Group A and B in both sessions.
  - Only small advantages for high-qualified inspectors.

- **Similar system parts**:
  - Significant differences for all qualification classes and both groups.
  - Advantages for less- and medium-qualified inspectors.
Results: Efficiency

- **Efficiency**: number of found defects per hour.

<table>
<thead>
<tr>
<th></th>
<th>Session 1</th>
<th></th>
<th>Session 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Central</td>
<td>Taxi</td>
<td>Central</td>
<td>Taxi</td>
</tr>
<tr>
<td><strong>Mean</strong></td>
<td>2.7</td>
<td>5.3</td>
<td>4.4</td>
<td>7.4</td>
</tr>
<tr>
<td><strong>Std.Dev.</strong></td>
<td>1.4</td>
<td>3.1</td>
<td>0.7</td>
<td>2.0</td>
</tr>
</tbody>
</table>

- **System Complexity**:
  - We observed significant differences between less- and higher qualified inspectors in both session within the more complex part.
  - No significant differences in the taxi part.

- **Similar system parts**:
  - There is a notable learning effect (p<0.040) in session 2 for all qualification classes.
  - We observed the highest improvement factor for medium qualified inspectors and the lowest for higher-qualified inspectors.
Results: False Positives

- **False Positives (FP):** share of “wrong defects found”.

<table>
<thead>
<tr>
<th></th>
<th>Session 1</th>
<th></th>
<th>Session 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Central</td>
<td>Taxi</td>
<td>Central</td>
<td>Taxi</td>
</tr>
<tr>
<td>Mean</td>
<td>63.3%</td>
<td>36.9%</td>
<td>55.7%</td>
<td>41.7%</td>
</tr>
<tr>
<td>Std.Dev.</td>
<td>19.4%</td>
<td>10.9%</td>
<td>21.9%</td>
<td>9.6%</td>
</tr>
</tbody>
</table>

- **System Complexity:**
  - No significant differences within both sessions.

- **Similar system parts:**
  - The FP decreased for group A (starting with the more complex central part of the system) and increased for group B (starting with taxi).
Summary and Further Work

Summary:
- Software inspection is an appropriate method for defect detection in early software development phases and learning.
- The results showed that inspectors who started with a simple (less complex) system part are more successful regarding effectiveness, efficiency and false positives.
- We observed a smaller learning effect for higher-qualified inspectors. One reason might be that they apply their own suitable inspection approach.

Further Work:
- More detailed investigation of the results regarding inspector qualification, different defect types and document locations will lead to a deeper insight in learning with inspection.
Backup
Study Artifacts

System Overview: Taxi Management System

Software artifacts per part

<table>
<thead>
<tr>
<th></th>
<th>Central (DS1)</th>
<th>Taxi (DS2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements definitions</td>
<td>4 pages</td>
<td>4 pages</td>
</tr>
<tr>
<td></td>
<td>1,400 words</td>
<td>1,100 words</td>
</tr>
<tr>
<td></td>
<td>16 use cases</td>
<td>11 use cases</td>
</tr>
<tr>
<td>Design document</td>
<td>5 pages</td>
<td>3 pages</td>
</tr>
<tr>
<td></td>
<td>1,600 words</td>
<td>800 words</td>
</tr>
<tr>
<td>Source Code in Lines of Code (LoC)</td>
<td>750 LoC</td>
<td>650 LoC</td>
</tr>
<tr>
<td></td>
<td>2,100 words</td>
<td>1,400 words</td>
</tr>
<tr>
<td>Common system information</td>
<td>1 component diagram</td>
<td>1 class diagram</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 sequence charts</td>
</tr>
</tbody>
</table>

Seeded defects per part

<table>
<thead>
<tr>
<th></th>
<th>Central (DS1)</th>
<th>Taxi (DS2)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Defects</td>
<td>30</td>
<td>26</td>
<td>56</td>
</tr>
<tr>
<td>Share of Defects</td>
<td>54%</td>
<td>46%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Data Set 1 (Central part) is the more complex part (higher amount of inspection material) of the system.