Empirical Software Engineering Research

Knowledge: Systematic Literature Review (SLR)

Norman Peitek, Annabelle Bergum, Lina Lampel, Sven Apel

Learning Goals

- Gain an overview over typical secondary research methods
- Understand the difference between a literature review and a structured literature review

Primary versus Secondary Research

- Primary research is studying something with a scientific method
  - Different primary research results may return inconsistent results
  - For example, they might find different effects or different effect sizes
- Secondary research is studying primary studies in a scientific way
  - Systematic literature reviews compile all the evidence in one place, assess their respective quality or extract their exact contribution to a given question and synthesize a new answer
  - Statistical meta analysis combines different quantitative studies for example by averaging or weighting the effect sizes in the respective studies
  - Secondary research can be done on any primary research (including qualitative studies)
- The goal of secondary research is
  - to synthesize higher level insight from primary studies, and
  - to achieve higher levels of validity than individual studies

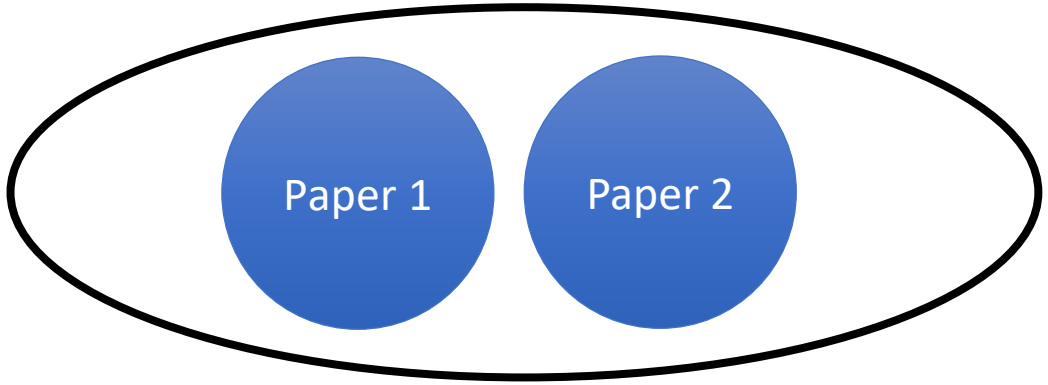
Literature Review

- Have you reviewed literature for a specific topic?

- What was your process and experience?

Literature Review

- When exploring a new topic scientifically, researchers need to familiarize themselves with the current state of the art
  - Search for relevant literature
  - Understand current knowledge on the topic
  - Identify current limitations
  - Synthesize different sources



Literature Review: Reading Papers

- Do not blindly read the entire paper
- Focus first on important aspects
  - Title
  - Abstract
  - Intro (+ contributions)
  - Conclusion
- Only if paper is very relevant, read the entire paper

What Drives the Reading Order of Programmers? An Eye Tracking Study

Norman Peitek
Leibniz Institute for Neurobiology
Magdeburg, Germany

Janet Siegmund
Chemnitz University of Technology
Chemnitz, Germany

Sven Apel
Saarland University
Saarbrücken, Germany

ABSTRACT

Background: The way how programmers comprehend source code depends on several factors, including the source code itself and the programmer. Recent studies showed that novice programmers tend to read source code more like natural language text, whereas experts tend to follow the program execution flow. But, it is unknown how the *linearity of source code* and the comprehension strategy influence programmers' linearity *of reading order*.
Objective: We replicate two previous studies with the aim of additionally providing empirical evidence on the influencing effects of linearity of source code and programmers' comprehension strategy on linearity of reading order.
Methods: To understand the effects of linearity of source code on reading order, we conducted a non-exact replication of studies by Busjahn et al. and Peachock et al., which compared the reading order of novice and expert programmers. Like the original studies, we used an eye-tracker to record the eye movements of participants (12 novice and 19 intermediate programmers).
Results: In line with Busjahn et al. (but different from Peachock et al.), we found that experience modulates the reading behavior of participants. However, the linearity of source code has an even stronger effect on reading order than experience, whereas the comprehension strategy has a minor effect.
Implications: Our results demonstrate that studies on the reading behavior of programmers must carefully select source code snippets to control the influence of confounding factors. Furthermore, we identify a need for further studies on how programmers should structure source code to align it with their natural reading behavior to ease program comprehension.

CCS CONCEPTS

• **Human-centered computing** → *Empirical studies in HCI; HCI design and evaluation methods;*

KEYWORDS

program comprehension, eye tracking, source code linearity, linearity of reading order, programmer experience

ICPC '20, October 5–6, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7958-8/20/05…$15.00
https://doi.org/10.1145/3387904.3389279

ACM Reference format:
Norman Peitek, Janet Siegmund, and Sven Apel. 2020. What Drives the Reading Order of Programmers? An Eye Tracking Study. In *Proceedings of 28th International Conference on Program Comprehension, Seoul, Republic of Korea, October 5–6, 2020 (ICPC '20),* 12 pages.
https://doi.org/10.1145/3387904.3389279

1 INTRODUCTION

In the past decades, much research has focused on how programmers comprehend source code, which is a central activity in software development [26, 52]. The underlying cognitive process, *program comprehension*, is a prerequisite for all subsequent programmer activities, such as testing, debugging, and maintenance. Past research theorized on two main strategies of how programmers comprehend software. *Bottom-up comprehension* is used when programmers lack domain knowledge, experience, or context to efficiently understand source code [39]. Instead, they have to understand individual source code lines and statements and integrate their semantic meaning to eventually build an overarching understanding (i.e., chunking [46]). *Top-down comprehension* is used when programmers take advantage of previous experience or domain knowledge for an efficient hypothesis-driven comprehension process [31], for example, guided by variable identifiers [10].

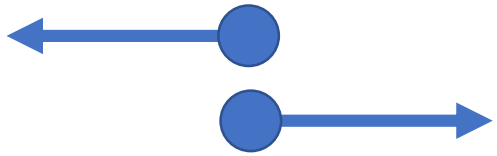
Although there is some evidence for the validity of these existing comprehension models, there are still knowledge gaps, such as when and how programmers are able to apply top-down comprehension. Program comprehension is an internal cognitive process and as such inherently difficult to measure [47]. Conventional methods, such as think-aloud protocols or measuring task efficiency, cannot provide deep insights into the underlying cognitive processes of program comprehension.

One important aspect of program comprehension is observing the way programmers read source code. Eye tracking has proved useful to observe programmers reading source code and answer such fundamental research questions on program comprehension (e.g., [13, 44, 53]). For example, Sharif and Maletic replicated a conventional study with eye tracking and found that naming style affects program comprehension in that programmers are able to read under_score style faster than camelCase style [8, 44].

Previous research suggested that the *linearity of the reading order* could be an indicator of how efficient programmers comprehend source code [13]. Busjahn et al.'s seminal study described several eye-gaze measures to gauge linearity of reading order. They showed that programmers read source code less linear than natural text and also that expert programmers read source code less linearly than novices [13]. This study indicates that comprehending source code is a skill that needs to be developed and honed with experience. A replication of Busjahn et al.'s study by Peachock et al. supports

Literature Review: Finding More Papers

- Use most relevant paper as starting point
  - Typically, your advisor can provide hints
- Read related work/background sections for further literature
- Snowballing
  - Backward: which papers does this paper cite?
  - Forward: which new papers cite this paper?



- Search by author, venue, topic/method

Systematic Literature Review

- A literature review is an unstructured, informal process
- A systematic literature review (SLR) is a **structured** method of surveying and analyzing all available research on a particular topic
  - It is like a literature review (“related work section”), but in an extreme form
  - A SLR allows statements about the state of the art with scientific certainty, not just as an individual’s (expert) opinion
  - Unlike a literature review, a SLR is considered scientific
  - For example, the search strategy is pre-defined and can be assessed by a third party
  - SLRs force researchers to include all research (even if it does not fit their narrative)

Systematic Literature Review

- The benefit of SLR is that we can cover a substantial amount of material
  - The drawback is (clearly) that flaws in the underlying primary research affects the quality of the SLR adversely
  - SLR are a huge amount of work and can be quickly outdated
- A systematic literature can serve three purposes:
  - Summarize existing evidence
  - Identify trends and gaps in current research to outline future work
  - Provide framework of existing research to position and justify future work
- In 2010, an SLR on SLRs concluded that “SLR cannot be yet considered a mainstream SE research method”

Systematic Literature Review: Process

- Like any other study, set objectives and design protocol
  - Remember the objectives template
  - Document objectives, including why it is important

- Then, execute search according to protocol

Systematic Literature Review: Process

- Reduce search results by explicit inclusion/exclusion criteria
  - Inclusion: easy, objective, based on meta data/abstract
  - Exclusion: slower, more effort, subjective, requires full paper
- Once resulting set is achieved, analyze each study in detail
  - Extract methods (experiment design, sample, context, …) and results (findings, conclusions)
  - Depending on the context, you may need to contact authors for additional information (takes a lot of time!)
- Synthesize results: qualitatively summarize and discuss the resulting studies

Systematic Literature Review: Process

- Specify databases or venues
  - Google scholar, dblp, IEEE, ACM DL, Scopus, …
  - Specific conferences/journals
  - Make sure to have access
- Reviewing papers can be very tedious
  - In my case, peer reviewers did not believe that we manually analyzed 800+ papers
  - If you consider conducting one for a thesis, make sure to set a very narrow scope
- Ideally, find a well-suited supporting software

Systematic Literature Review: Upsides

- A well-executed SLR reduces bias (but not bias in the primary studies)
- SLRs provide information on a larger abstraction level
  - If all primary studies come to the same conclusion, the SLR provides evidence that the effect is robust and transferable
  - If the primary studies conflict each other, the sources of variation must be studied
- SLRs provide the strongest form of evidence (if they follow accepted guidelines)

Meta Analysis

Meta Analysis

- A meta analysis follows the same process as an SLR
  - → secondary research
- But, it applies a quantitative, statistical analysis of the data of each found paper
  - Requires individual access to the data (or open data)
  - Particularly useful if individual studies have a small scope/small sample
  - Can combine different methods
  - SLR and meta analysis can be combined
- Suffers from publication bias (negative results may not have been published)
- In some fields, there are meta-meta analyses

Meta Analysis

- A meta analysis can identify the effect and effect sizes across different studies
- It can also show the variability in effect between studies
  - Can we identify a reason for this variability?