Software Developers’ Work Habits and Expertise

Sebastian Baltes
@s_baltes

empirical-software.engineering

Disputation @ Universität Trier
“For me, thoroughly analyzing and understanding the state-of-practice is an essential first step towards improving how software is being developed. Too often, decisions are still rather opinion-based than data-informed.”
Evidence-based Practice through Practice-based Evidence
Goal of my PhD Research

Observe
Describe
Explain

Software Developers’ Work Habits

 ⇒

Expand knowledge

Derive requirements for better tool support
Identify possible process improvements
Communicate findings back to practitioners
A habit is a "settled tendency or usual manner of behavior"

https://www.merriam-webster.com/dictionary/habit

**Personal habits**

**Work habits**
Studied Habits

2013

Sketching

Expertise Development

Code Plagiarism

stack overflow
Overview of this Talk

1. Sketching
2. Expertise Development
3. Code Plagiarism

2013
Overview of this Talk

1. 2013 Sketching
2. Expertise Development
3. Code Plagiarism

Sebastian Baltes – Software Developers' Work Habits and Expertise
Sketching
Research Questions

Questions:
How and why do software practitioners use sketches and diagrams?
How are they related to source code?
How can we provide better tool support?

Approach:
Field study, online survey, lab study, formative tool evaluations
Sketches and Diagrams in Practice

Sebastian Baltes
Computer Science
University of Trier
Trier, Germany
s.baltes@uni-trier.de

Stephan Diehl
Computer Science
University of Trier
Trier, Germany
diehl@uni-trier.de

ABSTRACT
Sketches and diagrams play an important role in the daily work of software developers. In this paper, we investigate the use of sketches and diagrams in software engineering practice. To this end, we used both quantitative and qualitative methods. We present the results of an exploratory study in three companies and an online survey with 394 participants. Our participants included software developers, software architects, project managers, consultants, as well as researchers. They worked in different countries and on projects from a wide range of application areas. Most questions in the survey were related to the last sketch or diagram that the participants had created. Contrary to our expectations and previous work, the majority of sketches and diagrams are important because they depict parts of the mental model developers build to understand a software project [21]. They may contain different views, levels of abstraction, formal and informal notations, pictures, or generated parts [6, 11, 41, 42]. Developers create sketches and diagrams mainly to understand, to design, and to communicate [6]. Media for sketch creation include whiteboards, engineering notebooks, scrap papers, but also software tools like Photoshop [6, 11, 41, 42].

1. INTRODUCTION
Over the past years, studies have shown the importance of sketches and diagrams in software development [6,11,43]. Most of these visual artifacts do not follow formal conventions like the Unified Modeling Language (UML), but have an informal, ad-hoc nature [6,11,23,25]. Sketches and diagrams are important because they depict parts of the mental model developers build to understand a software project [21].

https://empirical-software.engineering/projects/sketches/
Sketches and Diagrams in Practice

Revision
- 54% whiteboard (40%)
  - analog (58%)
- paper (18%)
- transitions between media are common
- computer (39%)
- tablet (<1%)
- 77% digital (40%)

Media

Archiving
- archived (38%)
- not archived (62%)

Life Span
- several work days
- several months

Relation to Source Code
- 47% of the sketches are rated as helpful for others to understand the related source code artifacts.

Purpose
- Designing (75%)
- Explaining (60%)
- Understanding (56%)
- Analyzing Requirements (45%)
Sketching

https://www.youtube.com/watch?v=mG6xCiQpS80
Future Work

• **Follow-up study:** What has changed since the widespread adoption of **smartphones and tablets**?

• **SketchLink:** Adapt and evaluate with industry partner(s)

• **Graphic Facilitation:** **Explore use cases** in software development teams based on preliminary results from interviews
Overview of this Talk

1. Sketching
2. Expertise Development
3. Code Plagiarism

2013
Expertise Development
Towards a Theory of Software Development Expertise

Sebastian Baltes
University of Trier
Trier, Germany
research@sbaltes.com

Stephan Diehl
University of Trier
Trier, Germany
diehl@uni-trier.de

ABSTRACT

Software development includes diverse tasks such as implementing new features, analyzing requirements, and fixing bugs. Being an expert in those tasks requires a certain set of skills, knowledge, and experience. Several studies investigated individual aspects of software development expertise, but what is missing is a comprehensive theory. We present a first conceptual theory of software development expertise that is grounded in data from a mixed-methods survey with 335 software developers and in literature on expertise and expert performance. Our theory currently focuses on programming, but already provides valuable insights for researchers, developers, and employers. The theory describes important properties of software development expertise and which factors foster or hinder its formation, including how developers’ performance may decline over time. Moreover, our quantitative results show that developers’ expertise self-assessments are context-dependent and that experience is not necessarily related to expertise.

expert performance [78]. Bergersen et al. proposed an instrument to measure programming skill [9], but their approach may suffer from learning effects because it is based on a fixed set of programming tasks. Furthermore, aside from programming, software development involves many other tasks such as requirements engineering, testing, and debugging [62, 96, 100], in which a software development expert is expected to be good at.

In the past, researchers investigated certain aspects of software development expertise (SDExp) such as the influence of programming experience [95], desired attributes of software engineers [63], or the time it takes for developers to become “fluent” in software projects [117]. However, there is currently no theory combining those individual aspects. Such a theory could help structuring existing knowledge about SDExp in a concise and precise way and hence facilitate its communication [44]. Despite many arguments in favor of developing and using theories [46, 56, 85, 109], theory-driven research is not very common in software engineering [97].

https://empirical-software.engineering/projects/expertise/
Software Development Expertise?

- Implementing new features
- Algorithms & Data structures
- Testing
- Debugging
- Communication
Software Development Expertise?

- Implementing new features
- Algorithms & Data structures
- Testing
- Debugging
- Communication

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Research Questions

Questions:
How to structure all those expertise-related aspects? Which factors influence expertise development over time?

Approach:
Iterative theory building
Research Design

- **Induction:** 335 online survey participants in total
- **Deduction:** Main source “Cambridge Handbook of Expertise and Expert Performance”
Our Expertise Model

• **Task-specific** (e.g., writing code, debugging, testing)
• Focuses on **individual developers**
• **Process** view (repetition of tasks)
• Notion of **transferable knowledge and experience** from related fields or tasks
• **Continuum** instead of discrete expertise steps
Conceptual Theory

Individual differences
- Motivation
- Personality (FFM)
- Mental abilities
- Skills

Task
- Task-specific knowledge
- Task context
- Task-specific experience

Behavior
- affects
- deliberate practice
- affects

Performance
- affects

General knowledge
- affects
- transfer

Mentoring*
- affects
- generates

Education
- generates
- transfer

Feedback
- feedback
- self-reflection

Monitoring
- feedback

Sebastian Baltes – Software Developers’ Work Habits and Expertise
"For myself, it’s mostly the effects of aging on the brain. At age 66, I can’t hold as much information in short-term memory, for example. [...] I can compensate for a lot of that by writing simpler functions with clean interfaces. The results are still good, but my productivity is much slower than when I was younger."

"Programming ability is based on desire to achieve. In the early years, it is a sort of competition. [...] I found that I lost a significant amount of my focus as I became 40, and started using drugs such as ritalin to enhance my abilities. This is pretty common among older programmers."

software architect, age 66
software developer, age 60
Summary

**Researchers** can...
- Use our theory to **design studies** on expertise development
- Adopt our **theory building** approach

**Developers** can...
- Learn what other developers expect from **experts/mentors**
- Learn which **behaviors** may lead to becoming an expert

**Employers** can...
- Learn what **(de)motivates** employees and thus fosters or hinders expertise development
- Reflect on ideas to build a work environment **supporting self-improvement** of their staff
• **Study**: Identify **specific challenges** that **older developers** face to prevent such experienced knowledge workers from dropping out of software development

• **Study**: Further investigate role of **feedback** and **team changes** in expertise development

• **Study**: Investigate expertise development and ageing from a **sociological perspective** (team expertise, discourse analysis)
Overview of this Talk

1. Sketching
2. Expertise Development
3. Code Plagiarism

2013

Sebastian Baltes – Software Developers' Work Habits and Expertise
Code Plagiarism
Usage and attribution of Stack Overflow code snippets in GitHub projects

Sebastian Baltes¹ · Stephan Diehl¹

Published online: 01 October 2018
© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract
Stack Overflow (SO) is the most popular question-and-answer website for software developers, providing a large amount of copyable code snippets. Using those snippets raises maintenance and legal issues. SO’s license (CC BY-SA 3.0) requires attribution, i.e., referencing the original question or answer, and requires derived work to adopt a compatible license. While there is a heated debate on SO’s license model for code snippets and the

https://empirical-software.engineering/projects/snippets/
Stack Overflow’s License

“If you [...] **build upon** the material, you must **distribute your contributions** under the same license as the original.”

“You must give appropriate credit [...] and indicate if changes were made.”

Attribution  Share-alike
Results from our Online Surveys

• **46%** of the participants admitted copying code from Stack Overflow **without attribution**

• **75%** did **not know** that content on SO is licensed under **CC BY-SA**

• **67%** did **not know** that **attribution is required**

→ Lack of awareness
Background

“Well, but these snippets are rather trivial and not protected by copyright.”

• Not all code snippets on Stack Overflow are copyrightable

• “A snippet that is more than one or two lines of standard function calls would typically be creative enough for copyright” [Engelfriet 2016]

• But no “international standard for originality” [Creative Commons 2017b]
Here's what I do:

1. First of all I check what providers are enabled. Some may be disabled on the device, some may be disabled in application manifest.
2. If any provider is available I start location listeners and timeout timer. It's 20 seconds in my example, may not be enough for GPS so you can enlarge it.
3. If you get update from location listener I use the provided value. I stop listeners and timer.
4. If you don't get any updates and timer elapses I have to use last known values.
5. I grab last known values from available providers and choose the most recent of them.

Here's how I use the class:

```java
locationResult = MyLocationManager.getInstance().getLastLocationResult();

// MyLocation

MyLocationManager manager = MyLocationManager.getInstance();

manager.getLastLocationResult();

And here's MyLocation class:

```java
public class MyLocation {
    Timer timer;
    LocationManager lm;
    LocationResultResult;
    boolean gps_enabled=false;
    boolean network_enabled=false;

    public boolean getLocationResult(Context context, LocationResultResult result) {
        // use locationResult callback to pass location value from MyLocation to same code.
        return true;
    }
}
```

Somebody may also want to modify my logic. For example if you get data from Network provider don't stop listeners but continue waiting. GPS gives more accurate data so it's worth waiting for it. If timer elapses and you've got update from Network but not from GPS then you can use value provided from Network.

One more approach is to use LocationClient http://developer.android.com/training/location/notice.html. But it requires Google Play Services api to be installed on user device.

https://stackoverflow.com/a/3145655

Sebastian Baltes – Software Developers’ Work Habits and Expertise

https://github.com/pacosal/ownmdm/blob/master/src/com/pacosal/mdm/MyLocation.java
Stack Overflow Code in the OpenJDK

Get rid of the humanReadableByteCount() method in openjdk/hotspot

Details
- Type: Bug
- Priority: P2
- Affects Version/s: 9
- Component/s: hotspot
- Status: RESOLVED
- Resolution: Fixed
- Fix Version/s: 9

implement the method humanReadableByteCount which body was copied from the Stack Overflow site: https://stackoverflow.com/a/3758880

It's just a few lines of code, but it **could cause legal issues**. The method should be either re-implemented or removed.

Besides the potential legal issues, duplicating a code is **not a good practice**.

https://bugs.openjdk.java.net/browse/JDK-8170860
Implications of Stack Overflow’s License

Permissive Licenses

• Permit using the licensed source code in proprietary software without publishing changes or the derived work
• Examples: MIT, Apache, and BSD license families

Copyleft Licenses

• Requires either modifications to the licensed content or the complete derived work to be published under the same or a compatible license (share-alike)
• Examples (weak copyleft): Mozilla/Eclipse Public Licenses
• Examples (viral copyleft): GNU General Public Licenses, Creative Commons Share-Alike Licenses (e.g., CC BY-SA)
Enforceability of Copyleft Licenses

- Courts in the US and Europe ruled that open source licenses are **enforceable contracts**
- Authors are able to **sue** when terms such as the share-alike requirement are violated:
  - **Interdict distribution** of derived work
  - **Claim monetary damages**
- USA: DMCA takedown notices for allegedly infringed copyright
  - Example: [https://github.com/github/dmca](https://github.com/github/dmca)
- Risk in mergers and acquisitions of companies
  - Example: FSF vs. Cisco lawsuit
Research Question

Question:
How **frequently** is code from Stack Overflow posts used in public GitHub projects **without** the required **attribution**?

Approach:
Triangulate an estimate for the attribution ratio using three different methods.
Method 1: Regular Expressions

209m files in 4.1m projects

Google Cloud

GitHub

209m files in 4.1m projects

Java

13m Java files in 336k projects

Check if attributed
(URL to answer or corresponding question)

Check for false positives

4,198 files with matches

Check external availability

Manually build regular expressions matching code snippets
(referenced usages as test cases)

10 most frequently referenced answers

...stackoverflow\.com...

...stackoverflow\.com...

13m Java files in 336k projects

5 most frequently referenced answers

10 most frequently referenced answers

Java

...stackoverflow\.com...

Check external availability

Manually build regular expressions matching code snippets
(referenced usages as test cases)
Exemplary Regex

```java
public static String humanReadableByteCount(long bytes, boolean si) {
    int unit = si ? 1000 : 1024;
    if (bytes < unit) return bytes + " B";
    int exp = (int) (Math.log(bytes) / Math.log(unit));
    String pre = (si ? "kMGTE" : "KGTE").charAt(exp-1) + (si ? "" : "i");
    return String.format("%.1f %sB", bytes / Math.pow(unit, exp), pre);
}
```

https://stackoverflow.com/a/3758880
## Results

<table>
<thead>
<tr>
<th>Rank</th>
<th>Matches</th>
<th>Recall</th>
<th>Attribution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALL</td>
<td>DISTINCT</td>
<td>REF</td>
</tr>
<tr>
<td>1</td>
<td>997</td>
<td>448</td>
<td>97</td>
</tr>
<tr>
<td>2</td>
<td>1,843</td>
<td>913</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>2,662</td>
<td>902</td>
<td>87</td>
</tr>
<tr>
<td>4</td>
<td>420</td>
<td>170</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>1,492</td>
<td>402</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>2,642</td>
<td>807</td>
<td>65</td>
</tr>
<tr>
<td>7</td>
<td>160</td>
<td>124</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>355</td>
<td>174</td>
<td>22</td>
</tr>
<tr>
<td>9</td>
<td>295</td>
<td>225</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>65</td>
<td>33</td>
<td>11</td>
</tr>
<tr>
<td>All</td>
<td>10,931</td>
<td>4,198</td>
<td>402</td>
</tr>
</tbody>
</table>
Method 2: Code Clone Detector

- **Goal**: Use code clone detector to find clones of a sample of Stack Overflow snippets in a sample of GitHub projects

- **Why samples?**
  - Code clone detection is computationally expensive

- **Which snippets and projects to select?**
  - Random samples: Many *toy projects* on GitHub and many *irrelevant snippets* on Stack Overflow
  - Purposive sampling: Limited generalizability
GitHub Project Sample

- Focus on popular GitHub projects
- High precision in selecting “engineered” software projects [Munaiah et al. 2017]
- Greater (potential) impact of licensing issues

Watcher count filter for non-fork Java GH projects (n=925,536)

Sample size: 3,000 / 2,313

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Stack Overflow Snippet Samples

• Non-trivial snippets retrieved from 100 most frequently referenced answers (n=111)
  \[ \Rightarrow S_{\text{top100}} \]

• Non-trivial snippets retrieved from answers referenced in GitHub projects (n=137)
  \[ \Rightarrow S_{\text{gh}} \]

• *External sources*: Only three snippets available under a more permissive license than CC BY-SA
Code Clone Detector Calibration

Comparison of CPD configurations

https://pmd.github.io/
## Results

<table>
<thead>
<tr>
<th>Set</th>
<th>ALL</th>
<th>MATCHED</th>
<th>ANSWERS</th>
<th>MATCHED</th>
<th>MATCH.</th>
<th>REF</th>
<th>Repos</th>
<th>MATCHED</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_{gh}$</td>
<td>137</td>
<td>53 (39%)</td>
<td>102</td>
<td>52 (51%)</td>
<td>163</td>
<td>58</td>
<td>36%</td>
<td>124 (5%)</td>
</tr>
<tr>
<td>$S_{top100}$</td>
<td>111</td>
<td>48 (43%)</td>
<td>85</td>
<td>46 (54%)</td>
<td>173</td>
<td>25</td>
<td>14%</td>
<td>125 (5%)</td>
</tr>
<tr>
<td>US</td>
<td>222</td>
<td>101 (46%)</td>
<td>169</td>
<td>86 (51%)</td>
<td>297</td>
<td>70</td>
<td>24%</td>
<td>199 (9%)</td>
</tr>
</tbody>
</table>
Method 3: Exact Matches

• **Goal:** Address shortcomings of Method 1 and 2
  - Increase sample sizes
  - Exclude snippets available on external sources
  - Systematically exclude short snippets

• Select as many projects and snippets as possible and search for (almost) exact matches
Method 3: Exact Matches

- **Google Cloud**
  - Project is not a fork, has ≥ 5 Java files and ≥ 1 watcher(s)
  - File has ending .java has ≥ 68 NLOC (Q₃)

- **GitHub**
  - 209m files in 4.1m projects

- **Stack Overflow**
  - Question tagged java or android
  - Answer score ≥ 10
  - Code block ≥ 6 NLOC

- **Normalization and substring search**
  - 1.7m Java files in 64k projects
  - 10,358 matches
  - 29k snippets from 24k answers

Sebastian Baltes – Software Developers' Work Habits and Expertise
Details: Filtering of GitHub Projects

File size filter for GH Java files (n=6,851,022)

Watcher count filter for GH Java projects (n=260,498)

Fork filter for GH projects containing Java files (n=307,489)

File count filter for GH Java projects (n=260,498)
Details: Filtering of Stack Overflow Snippets

Score filter for SO Java answers (n=851,795)

Length filter for SO Java code blocks (n=1,063,993)

Proxies for originality
Method 3: Filtering of Matches

- Use heuristic to detect and exclude matches in mirrors of JDK and Android source code
- Manually analyze answers, exclude snippets that are too trivial, incomplete, or copied from an external source
- Use GitHub API to remove matches where commit adding snippet is older than answer on Stack Overflow

10,358 matches —→ 1,379 matches —→ 1,369 matches

Only 7.6% attributed

Check if attributed (URL to answer or corresponding question)

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Attribution

Attribution ratio:
• Method 1 (regular expressions): 23 %
• Method 2 (code clone detector): 24 %
• Method 3 (exact matches): 8 %

Conservative estimate:
• Attribution ratio ≤ 25%
Only 2% of all analyzed repositories (all methods) containing code from Stack Overflow attributed its source and used a compatible license (not CC BY-SA, but GPL 3.0).

<table>
<thead>
<tr>
<th>SPDX license name</th>
<th>Method 1</th>
<th>Method 2</th>
<th>Method 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of repos containing a SO code snippet clone that was unattributed (n = 2,962)</td>
<td>attributed (n = 329)</td>
<td>numbers of repos containing a SO code snippet clone that was unattributed (n = 144)</td>
</tr>
<tr>
<td>Apache-2.0</td>
<td>921 (31.1%)</td>
<td>99 (30.1%)</td>
<td>56 (39.9%)</td>
</tr>
<tr>
<td>MIT</td>
<td>621 (20.4%)</td>
<td>72 (21.0%)</td>
<td>31 (22.9%)</td>
</tr>
<tr>
<td>GPL-3.0</td>
<td>425 (14.7%)</td>
<td>66 (18.6%)</td>
<td>17 (11.8%)</td>
</tr>
<tr>
<td>GPL-2.0</td>
<td>284 (9.6%)</td>
<td>21 (6.4%)</td>
<td>6 (4.2%)</td>
</tr>
<tr>
<td>BSD-3-Clause</td>
<td>93 (3.1%)</td>
<td>9 (2.7%)</td>
<td>4 (2.8%)</td>
</tr>
</tbody>
</table>
Reaching out to Developers

• **Contacted owners** of GitHub repositories containing copies of Stack Overflow snippets

• **75% not aware** of CC BY-SA licensing (see slide about online surveys)

• Many thankful responses
Limitations

• Limited **generalizability** due to focus on Java

• Relatively **small samples** of snippets for Method 1 and 2
  • Still found a considerable number of files with copies
  • Attribution ratio was even smaller for Method 3, where we included more snippets and only searched for exact matches

• Focus on **type-1 clones** of snippets

• **External sources**
  • Analyzed for Method 1 and 2
  • Excluded in Method 3

• Not all matches may be protected by **copyright**
  • Used proxies for originality
SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts

Sebastian Baltes  
Lorik Dumani  
research@sbaltes.com  
University of Trier, Germany

Christopher Treude  
christoph.treude@adelaide.edu.au  
University of Adelaide, Australia

Stephan Diehl  
diehl@uni-trier.de  
University of Trier, Germany

Abstract—Stack Overflow (SO) is the most popular question-and-answer website for software developers, providing a large amount of copiable code snippets. Like other software artifacts, code on SO evolves over time, for example when bugs are fixed or APIs are updated to the most recent version. To be able to analyze how code and the surrounding text on SO evolves, we built SOTorrent, an open dataset based on the official SO data dump. SOTorrent provides access to the version history of SO content at the level of whole posts and individual text and code blocks. It connects code snippets from SO posts to other platforms by aggregating URLs from surrounding text blocks and comments, and by collecting references from GitHub files to SO posts. Our vision is that researchers will use SOTorrent to investigate and understand the evolution and maintenance of code on SO and its relation to other platforms such as GitHub.

sotorrent.org

Dataset available on Zenodo and BigQuery

Open Data
Future Work

• **Tool support**: Support maintainability of copied snippets by automatically adding links to sources, integration into CI tools

• **Education**: Help developers understand complex licensing situations (not only for complete libraries but also for individual snippets)

• **Study**: Analyze links to better understand Stack Overflow’s role in the ecosystem of documentation resources