Software Developers’ Work Habits and Expertise
Sketching, Code Plagiarism, and Expertise Development

Sebastian Baltes
@s_baltes

Universität Trier
empirical-software.engineering
Goals of Research

Observe
Describe
Explain
Predict

Phenomena $\Rightarrow$ Expand knowledge
Goals of my PhD Research

Software Developers’ Work Habits

Observe
Describe
Explain
(Predict)

Expand knowledge:
• Identify requirements for better tool support
• Point to possible process improvements
• Communicate results back to practitioners
“For me, thoroughly analyzing and understanding the **state-of-practice** is an essential first step towards **improving** how software is being developed, because too often, decisions are still rather opinion-based than **data-informed**.”
Habit?

„A settled tendency or usual manner of behavior“

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

2013
Sketches and Diagrams in Practice
SketchLink

Sketching

2018
How Developers Locate Performance Bugs
LivelySketches

Towards a Theory of Software Development Expertise
Expertise Development

Code Plagiarism

stackoverflow Code Snippets in GitHub Projects

Sebastian Baltes – Software Developers’ Work Habits and Expertise
“Parallel Thread”

Issues in Sampling Software Developers

Methodology

Open Data

Constructing Urban Tourism Space Digitally

Interdisciplinary Research

2013

2018

Sebastian Baltes – Software Developers’ Work Habits and Expertise

6
Studied Habits

2013
Sketches and Diagrams in Practice
SketchLink

Sketching
LivelySketches
How Developers Locate Performance Bugs

Towards a Theory of Software Development Expertise
Expertise Development

Code Plagiarism

stackoverflow Code Snippets in GitHub Projects
Sketches and Diagrams in Practice

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. They may contain different views, levels of abstraction, formal and informal notations, pictures, or generated parts. Developers create sketches and diagrams mainly to understand, to design, and to communicate. Media for sketch creation include whiteboards, engineering notebooks, scrap papers, but also software tools like Photoshop.

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/
Navigate, Understand, Communicate: How Developers Locate Performance Bugs

Sebastian Baltes*, Oliver Moseler*, Fabian Beck†, and Stephan Diehl*
* University of Trier, Germany
† VISUS, University of Stuttgart, Germany

Abstract—Background: Performance bugs can lead to severe issues regarding computation efficiency, power consumption, and user experience. Locating these bugs is a difficult task because developers have to judge for every costly operation whether runtime is consumed necessarily or unnecessarily. Objective: We wanted to investigate how developers, when locating performance bugs, navigate through the code, understand the program, and communicate the detected issues. Method: We performed a qualitative user study observing twelve developers trying to fix documented performance bugs in two open source projects. The developers worked with a profiling and analysis tool that visually depicts runtime information in a list representation and embedded into the source code view. Results: We identified typical navigation strategies developers used for pinpointing the bug, for instance, following method calls based on runtime consumption. The integration of visualization and code helped developers to directly because the steps and tools required to optimize a non-functional requirement like performance are substantially different from those applied for fixing a functional bug. These differences include: (i) developers cannot analyze whether a program is correct regarding performance because there only exist better or worse solutions; (ii) developers need to investigate not only program state but also runtime consumption; and (iii) collecting runtime information requires to set up realistic benchmarks that differ from usual regression tests. Also, Jin et al. [1] already pointed at the lack of studies on how performance bugs are fixed by developers.

The user study presented in this paper aims at filling this gap by investigating how developers navigate through code, understand performance problems, and communicate

https://empirical-software.engineering/projects/debugging/
Linking Sketches and Diagrams to Source Code Artifacts

Sebastian Baltes, Peter Schmitz, and Stephan Diehl
Computer Science
University of Trier
Trier, Germany
{s.baltes,diehl}@uni-trier.de

ABSTRACT

Recent studies have shown that sketches and diagrams play an important role in the daily work of software developers. If these visual artifacts are archived, they are often detached from the source code they document, because there is no adequate tool support to assist developers in capturing, archiving, and retrieving sketches related to certain source code artifacts. This paper presents SketchLink, a tool that aims at increasing the value of sketches and diagrams created during software development by supporting developers in these tasks. Our prototype implementation provides a web application that employs the camera of smartphones and tablets to capture analog sketches, but can also be used on desktop computers. Sketches can be linked to source code by associating them with code fragments or generated parts [5, 8, 20, 21]. Developers create sketches and diagrams mainly to understand, to design, and to communicate [1, 5]. Media used for sketch creation include not only whiteboards and scrap paper, but also software tools like Photoshop and PowerPoint [5, 10, 17, 22].

Sketches and diagrams are important because they depict parts of the mental model developers build to understand a software project [13]. Understanding source code is one of the most important problems developers face on a daily basis [5, 12, 13, 19]. However, this task is often complicated by documentation that is frequently poorly written and out of date [9, 15]. Sketches and diagrams, whether formal or informal, can fill in this gap and serve as a supplement to conventional documentation like source code comments. To this

https://empirical-software.engineering/projects/sketchlink/
Round-Trip Sketches: Supporting the Lifecycle of Software Development Sketches from Analog to Digital and Back

Sebastian Baltes, Fabrice Hollerich, and Stephan Diehl
Department of Computer Science
University of Trier
Trier, Germany
Email: research@sbaltes.com, diehl@uni-trier.de

Abstract—Sketching is an important activity for understanding, designing, and communicating different aspects of software systems such as their requirements or architecture. Often, sketches start on paper or whiteboards, are revised, and may evolve into a digital version. Users may then print a revised sketch, change it on paper, and digitize it again. Existing tools focus on a paperless workflow, i.e., archiving analog documents, or rely on special hardware—they do not focus on integrating digital versions into the analog-focused workflow that many users media [13], because digital sketches can more easily be edited, copied, organized, and shared [18]. Even if a digital version exists, analog sketches may be kept as a memory aid [19]. Context information is often needed to understand informal sketches [20] and information may get lost due to the transient nature of sketches [12], [14].

Despite the widespread usage of sketches in many domains, to the best of our knowledge there is currently no tool that

https://empirical-software.engineering/projects/livelysketches/
Sketching

https://www.youtube.com/watch?v=mG6xCiQpS80
Lively Sketches

1. analog
2. digital
3. access annotations, linked data, and revisions of the sketch
4. edit and annotate
5. link
6. 7.
7. 8.
8. 9.

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Studied Habits

Sketches and Diagrams in Practice

SketchLink

LivelySketches

How Developers Locate Performance Bugs

Towards a Theory of Software Development Expertise

Expertise Development

2013

2018

Stack Overflow: Code Snippets in GitHub Projects

Code Plagiarism
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 these tasks requires a certain set of skills and experience. Several studies investigated individual software development expertise, but what is missing is a unifying theory. We present a first conceptual theory of software development expertise that is grounded in data from a survey with 335 software developers and in literature review. 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, debugging [62, 96, 100], in which a software developer’s expertise 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].
Studied Habits

Sketches and Diagrams in Practice

How Developers Locate Performance Bugs

Towards a Theory of Software Development Expertise

SketchLink

LivelySketches

Code Plagiarism

stackoverflow Code Snippets in GitHub Projects

2013

2018

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

Sebastian Baltes¹ ID - Stephan Diehl¹ ID

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/
Usage and Attribution of

[Stack Overflow] Code Snippets

in [GitHub] Projects
Question 1

Who admits regularly copying non-trivial code snippets from Stack Overflow?
Question 2

Who knew that all content on Stack Overflow is licensed under CC BY-SA?
Background

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

• Not all code snippets on Stack Overflow are copyrightable
• “If two programmers would provide substantially the same piece of code, the code is not creative under copyright law” [Engelfriet 2016]
• “A snippet that is more than one or two lines of standard function calls would typically be creative enough for copyright” [Engelfriet 2016]
• No “international standard for originality” [Creative Commons 2017b]
“Do Stack Overflow authors care about attribution?”
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)
Implications of Stack Overflow’s License

"You must give **appropriate credit**, provide a link to the license, and indicate if changes were made."

If you remix, transform, or build upon the material, you must **distribute your contributions** under the same license as the original.
Implications of Stack Overflow’s License

• Courts in the US and Europe ruled that open source licenses are **enforceable contracts**
• Developers are able to **sue** when terms like the share-alike requirement are violated:
  • **Interdict distribution** of derived work
  • **Claim monetary damages**
• USA: DMCA takedown notices for allegedly infringed copyright
  • See, e.g., [https://github.com/github/dmca](https://github.com/github/dmca)
• Risk in mergers and acquisitions of companies
  • See, e.g., FSF vs. Cisco lawsuit
General Scientific Workflow

“Big picture”

Related Work

Data Collection

Analysis

Results

Discussion

Conclusion
Our Research Questions

**RQ1:** How often is code from Stack Overflow posts used in public GitHub projects without the required attribution?

**RQ2:** How often does the license of repositories containing code copied from Stack Overflow conflict with Stack Overflow’s license?

**RQ3:** Do developers adhere to the attribution requirements defined in the Stack Overflow terms of service?

**RQ4:** Are software developers aware of the licensing of Stack Overflow code snippets and its implications?
RQ1: Triangulation

• Term “triangulation” is an analogy to land surveying
• Increase validity of research by studying a phenomenon from several points of view

• Cross-validation from two or more sources:
  • Different data sources
  • Different aspects of the same phenomenon
  • Different research instruments
  • Different researchers

• Here: Use three different approaches to estimate the attribution ratio of snippets copied from Stack Overflow into GitHub projects.
Phase 1: Research Design

- **BigQuery GH data** (209m files in 4.1m projects)
  - Filter files: path ending with `.java`
  - Java files (13m files in 336,028 projects)
    - Check for possible license conflicts.
    - Find links to SO questions & answers.

- **BigQuery SO data** (21m answers)
  - Build regexes to match code snippets from ten most frequently referenced answers.
  - Use regexes to search Java files with BigQuery.
  - Matches (4,198 files)

- **Awareness Survey** (n=87)
  - Contact sample of developers who presumably copied code.
  - Manually analyze attribution.
  - Check if matches are attributed.

10 Java snippets, all Java files on GitHub*

* All Java files in the Google BigQuery GitHub dataset
Phase 1: 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 ? "kMGTPe" : "KMGTPe").charAt(exp-1) + (si ? "m" : "i");
    return String.format("%.1f %sB", bytes / Math.pow(unit, exp), pre);
}
```

https://stackoverflow.com/a/3758880
### Phase 1: Recall

Table 3  RQ 1 – Phase 1: Ten most frequently referenced code snippets from SO Java answers; estimated ratio of unattributed usages detected using regular expressions; number of matched files (ALL), distinct matches (DISTINCT), distinct matches with reference to SO (REF), distinct matches without reference to SO (NO-REF)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Matches</th>
<th></th>
<th></th>
<th></th>
<th>Recall</th>
<th>Attribution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALL</td>
<td>DISTINCT</td>
<td>REF</td>
<td>NO-REF</td>
<td>REF/F&lt;sub&gt;AQ&lt;/sub&gt;</td>
<td>REF/DISTINCT</td>
</tr>
<tr>
<td>1</td>
<td>997</td>
<td>448</td>
<td>97</td>
<td>351</td>
<td>79.5%</td>
<td>21.7%</td>
</tr>
<tr>
<td>2</td>
<td>1,843</td>
<td>913</td>
<td>60</td>
<td>853</td>
<td>60.0%</td>
<td>6.6%</td>
</tr>
<tr>
<td>3</td>
<td>2,662</td>
<td>902</td>
<td>87</td>
<td>815</td>
<td>80.6%</td>
<td>9.6%</td>
</tr>
<tr>
<td>4</td>
<td>420</td>
<td>170</td>
<td>18</td>
<td>152</td>
<td>94.7%</td>
<td>10.6%</td>
</tr>
<tr>
<td>5</td>
<td>1,492</td>
<td>402</td>
<td>25</td>
<td>377</td>
<td>73.5%</td>
<td>6.2%</td>
</tr>
<tr>
<td>6</td>
<td>2,642</td>
<td>807</td>
<td>65</td>
<td>742</td>
<td>87.8%</td>
<td>8.1%</td>
</tr>
<tr>
<td>7</td>
<td>160</td>
<td>124</td>
<td>12</td>
<td>112</td>
<td>29.3%</td>
<td>9.7%</td>
</tr>
<tr>
<td>8</td>
<td>355</td>
<td>174</td>
<td>22</td>
<td>152</td>
<td>61.1%</td>
<td>12.6%</td>
</tr>
<tr>
<td>9</td>
<td>295</td>
<td>225</td>
<td>5</td>
<td>220</td>
<td>10.6%</td>
<td>2.2%</td>
</tr>
<tr>
<td>10</td>
<td>65</td>
<td>33</td>
<td>11</td>
<td>22</td>
<td>42.3%</td>
<td>33.3%</td>
</tr>
<tr>
<td>All</td>
<td>10,931</td>
<td>4,198</td>
<td>402</td>
<td>3,796</td>
<td>61.9%</td>
<td>12.1%</td>
</tr>
</tbody>
</table>

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Phase 2: Research Design

• **Goal:** Search for clones of a sample of Stack Overflow snippets in a sample of GitHub projects using a more scalable approach

• 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
  • Sampling based on distribution of certain properties
Background: Sampling

Who do you want to generalize to?

What populations can you get access to?

How can you get access to them?

Who is in your study?

Who participated?

Theoretical Population

Study Population

Sampling Frame

Sample

Respondents

Random Selection

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Phase 2: 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

Sample size: 3,000 / 2,313
Phase 2: Stack Overflow Snippet Samples

- Snippets from 100 most frequently referenced Stack Overflow answers (phase 1)  \( \Rightarrow S_{top100} \)
- Snippets from answers referenced in GitHub projects  \( \Rightarrow S_{gh} \)

**Definition 1** Let \( C \) (copies) be a relation over a set of code snippets \( S \) and a set of source code files \( F \):

\[ C \subseteq S \times F \]

Let \( C_{so} \subseteq C \) be the set of copies identified by an SO answer URL in the source code file and \( C_{cpd} \subseteq C \) be the set of copies identified by CPD. Then we define precision and recall as follows:

\[
\text{precision} = \frac{|C_{so} \cap C_{cpd}|}{|C_{cpd}|} \quad \text{recall} = \frac{|C_{so} \cap C_{cpd}|}{|C_{so}|}
\]

**Sample size:**

111 / 137
Phase 2: Code Clone Detector Calibration

Comparison of CPD configurations

https://pmd.github.io/
Phase 3: Research Design

• **Goal:** Address shortcomings of phases 1 and 2
  • External sources
  • Small sample sizes
  • Some rather short snippets

• Select as many projects and snippets as possible and search only for (almost) exact matches

Many Java snippets, many Java projects
Phase 3: Research Design

BigQuery GH data set (4.1m projects)

Filter projects:
- Not a fork.
- At least 5 Java files.
- At least one watcher.

BigQuery GHT data set (3.3m projects)

Filter files:
- At least 68 NLOC (75% quantile of all Java files).

Java files
(1.7m files from 64,281 projects)

Normalization

Substring search using BigQuery

Java snippets
(29,370 snippets from 23,829 answers)

Normalization

Extract non-trivial snippets:
- Code block must have at least 6 NLOC and at least a score of 10.

Check for possible license conflicts.

Matches
(10,358 file-snippet pairs)

Remove false positives and check if matches are attributed (see Figure 7).

BigQuery SO data set (21m answers)

Filter answers:
- Corresponding question tagged with “java” or “android”.

RQ2

RQ1

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Phase 3: Filtering 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)
Phase 3: Filtering 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
Background: Exploratory Data Analysis
Phase 3: Snippets with External Source

Matches
(10,358 file-snippet pairs)

Analyse links to external sources in matched answers and exclude snippets available outside of SO.

7,068 file-snippet pairs

Use heuristic to exclude remaining matches in mirrors of JDK or Android source code.

6,779 file-snippet pairs

Manually analyse and rate all remaining snippets (n=909): Open SO posts in browser, read text and code, and exclude snippets that are too trivial, incomplete, or copied from an external source.

1,379 file-snippet pairs

Cleaned Matches
(1,369 file-snippet pairs)

Search matched files for links to one of the SO posts containing the snippet or the corresponding questions.

Search for commit adding snippet and exclude matches where commit is newer than post on SO.

Result: Only 7.3% of the matches were attributed.

RQ1
### RQ1: Results

**Table 8** Summary of results from phases 1 to 3: Distinct references to answers (A) or questions (Q) on Stack Overflow (SO) in the Java files from GitHub analyzed in each phase; number of analyzed files and repositories, files/repos containing a reference to SO, files/repos containing a copy of a SO snippet, attributed copies of SO snippets.

| Ph. | References | Files | | | | | Repositories |
|-----|------------|-------|------|------|-------|------|------------|------|------|------------|
|     | A          | Q     | COUNT| REF  | COPY | ATTR| COUNT | REF | COPY | |
| 1   | 5,014      | 16,298| 13.3m| 18,605| 4,198| 402 | 336k  | 11,086| 3,291| |
|     | 23.5%      | 76.5% | 0.09%| 0.03%| 9.6% | 3.3% | 1.0% |
| 2   | 209        | 463   | 445k | 634  | 297  | 70  | 2,313 | 274  | 199  | |
|     | 31.1%      | 68.9% | 0.14%| 0.07%| 23.6%| 11.9%| 8.6% |
| 3   | 1,551      | 4,843 | 1.7m | 5,354| 1,369| 104 | 64,281| 3,536| 1,332| |
|     | 24.3%      | 75.7% | 0.31%| 0.08%| 7.6% | 5.5% | 2.1% |
Our Research Questions

**RQ1:** How often is code from Stack Overflow posts used in public GitHub projects without the required attribution?

**RQ2:** How often does the license of repositories containing code copied from Stack Overflow conflict with Stack Overflow’s license?

**RQ3:** Do developers adhere to the attribution requirements defined in the Stack Overflow terms of service?

**RQ4:** Are software developers aware of the licensing of Stack Overflow code snippets and its implications?
Research Design: Phase 1

- **BigQuery GH data** (209m files in 4.1m projects)
  - Filter files: path ending with `.java`
  - **Java files** (13m files in 336,028 projects)
    - Find links to SO questions & answers.
    - Check for possible license conflicts.

- **BigQuery SO data** (21m answers)
  - Build regexes to match code snippets from ten most frequently referenced answers.
  - Use regexes to search Java files with BigQuery.
  - **Matches** (4,198 files)

- **Awareness Survey** (n=87)
  - Contact sample of developers who presumably copied code.

- **RQ4**
- **RQ3**
- **RQ1**
- **RQ2**
Survey Results

• Contacted owners of GitHub projects containing copies of Stack Overflow snippets

• Received 87 responses (11.8% response rate)

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

• **41%** admitted **regularly copying** code from Stack Overflow

• Many thankful comments
Survey Results: Stack Overflow Snippet in JDK

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

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
Limitations

• Focus on Java, generalizability to other programming languages is limited

• In phases 1 and 2, we only considered relatively small samples of snippets
  • Still found a considerable number of files with copies
  • Number of attributions was even smaller in phase 3, where we included more snippets and only searched for exact matches

• External sources
  • Analysis in paper
  • Excluded in phase 3

• Not all matches may be protected by copyright
  • Used proxies for originality
Background: Verifiability

“Big picture”

“Big picture”

Data Collection

Analysis

Provenance tracking

Discussion

Results

Conclusion

Provenance tracking

Publish results, raw data, and provenance information

Related Work

Question

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Usage and Attribution of Code Snippets in GitHub Projects

Sebastian Baltes
@s_baltes

snippets.sbaltes.com

Supplementary material available on Zenodo.
Context Switch
“Parallel Thread”

Issues in Sampling Software Developers

Methodology

Open Data

Constructing Urban Tourism Space Digitally

Interdisciplinary Research

2013

2018

Sebastian Baltes – Software Developers' Work Habits and Expertise
Studying the Origin, Evolution, and Usage of Stack Overflow Code Snippets

Dataset available on Zenodo and BigQuery
SOTorrent: Reconstructing and Analyzing the Evolution of Stack Overflow Posts

Sebastian Baltes
Lorik Dumani
research@sbaltes.com
dumani@uni-trier.de
University of Trier, Germany

Christoph 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 code snippets and free-form text on a wide variety of software artifacts. Questions and answers on SO are edited for clarity. To be able to search for bugs in code snippets, 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 like GitHub 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: Studying the Origin, Evolution, and Usage of Stack Overflow Code Snippets

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

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

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

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.
Why Reconstruct and Analyze SO Post Evolution?

• The content of **14.3 million posts** has been **edited** after creation
  (SO data dump 2018-03-01)

• Like other **software artifacts**, SO posts **evolve over time**:  
  • Bugs in code snippets are fixed  
  • Clarifications are added in text documenting the code  
  • Snippets are updated to new language/library versions

• **Copying code** from Stack Overflow (SO) is common, despite licensing, security, and maintainability implications
Why Reconstruct and Analyze SO Post Evolution?

**Evolution of code on SO** differs from regular software projects:

- **Short** code snippets (12 LOC on average)
- **No bug tracking** system (just comments and new answers)
- **No versioning** for individual snippets (just whole posts)
Example

Question
https://stackoverflow.com/q/309424

Answer
https://stackoverflow.com/a/5445161

Sebastian Baltes – Software Developers’ Work Habits and Expertise
Here's a way using only standard Java library (note that the stream is not closed, YMMV).

```java
static String convertStreamToString(java.io.InputStream is) {
    java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\A");
    return s.hasNext() ? s.next() : "";
}
```

I learned this trick from "Stupid Scanner tricks" article. The reason it works is because `Scanner` iterates over tokens in the stream, and in this case we separate tokens using "beginning of the input boundary" (\A) thus giving us only one token for the entire contents of the stream.

**Note,** if you need to be specific about the input stream's encoding, you can provide the second argument to `Scanner` constructor that indicates what charset to use (e.g. "UTF-8").

Hat tip goes also to Jacob, who once pointed me to the said article.

**EDITED:** Thanks to a suggestion from Patrick, made the function more robust when handling an empty input stream. **One more edit:** mixed try/catch, Patrick's way is more laconic.
Here's a way using only standard Java library (note that the stream is not closed, YMMV).

```java
static String convertStreamToString(java.io.InputStream is) {
    java.util.Scanner s = new java.util.Scanner(is).useDelimiter("\A\"\"; \r\n\n);  
    return s.hasNext() ? s.next() : "";
}
```

I learned this trick from "Stupid Scanner tricks" article. The reason it works is because Scanner iterates over tokens in the input stream and we separate tokens using the entire contents of the stream as a single "boundary" (\A\"\"; \r\n\n) thus giving us a single scan. Hat tip goes also to Jacob, who once pointed me to the said article.

**EDITED:** Thanks to a suggestion from Patrick, made the function more robust when handling an empty input stream. **One more edit:**ixed try/catch, Patrick's way is more laconic.
EDITED: Thanks to a suggestion from Patrick, made the function more robust when handling an empty input stream. One more edit: mixed try/catch, Patrick's way is more laconic.

@PavelRepin In my case, an empty inputStream caused a NPE during Scanner construction. I had to add if (is == null) return ""; right at the beginning of the method; I believe this answer needs to be updated to better handle null inputStreams. – CFLJaff Aug 9 '12 at 13:36

The problem with this approach I find is it does not handle CRLF translations too well. So you have to make sure your line endings are consistent. – Archimedes Trigo Feb 28 '13 at 12:15

@ArchimedesTrigo does InputStream, writer, encoding) deal with CRLF translations better? I think CRLF consistency is entirely unrelated issue. Not saying it isn’t an issue. – Pavel Repin Mar 11 '13 at 9:18

For Java 7 you can close in a by-with try(java.util.Scanner s = new java.util.Scanner(is)) { return s.useDelimiter(".*\n").hasNext() ? s.next() : ""; } – earcan Jun 13 '13 at 5:29

Unfortunately this solution seems to go and lose the exceptions thrown in my underlying stream implementation. – Tad Jul 16 '13 at 7:09

Excellent trick! Any ideas about performance of Scanner vs reading the stream in a more verbose way? – isapi Aug 28 '13 at 19:54

@gal I didn’t measure it. If you do, get it and I’ll append your results to the answer. – Pavel Repin Aug 28 '13 at 23:13

FYI, hasNext blocks on console input streams (see here). (Just ran into this issue right now.) This solution works fine otherwise... just a heads up. – Ryan Feb 24 '14 at 5:36

@earcan thanks for the tip! For those wondering how this works, it’s thanks to try-with-resources – Mark Mar 14 '16 at 21:33

1 looks like a neat trick, but it seems there are some limitations. For me it hangs when reading InputStream from Socket. When dealing with something like InputStream it works nicely. Reading from socket results in a hang. – Normunds Kalberzins Dec 16 '15 at 14:16

If the Scanner is going to be “giving us only one token for the entire contents of the stream” anyways, why not use a normal stream reader? Scanner is meant to pre-parse tokens out of the stream, not for being the stream reader (without any parsing being done). – Xenoffo Dec 26 '15 at 14:06

@AmguyR Scanner has built-in stream reading logic and we’re telling it that the stream has just one token. A special case of Scanner usage. Fair game. Good point though. This stuff is clearly a hack. – Pavel Repin Jan 15 '16 at 1:23

be careful using this method with socket stream is slow! Scanner(next()) hangs for a little while. – WeZFarmer Apr 20 '16 at 10:22

1 nice answer, the article link is on oracle website community.oracle.com/blogs/pal/2004/10/23/stupid-scanner-tricks – Eng_Samer Jul 23 '17 at 16:04

This stuff is clearly a hack.
Even for such a simple code snippet, the **context** is quite **complex**:

- The snippet is based on an **external source**
- Hidden in the **comments**, the author acknowledges that his solution is “**clearly a hack**”
- There are several **bug reports** pointing to issues
- Has the snippet been **edited** to fix those issues?
- Is the snippet **safe** to use?
SOTorrent
Retrieve all versions of a code snippet:

```sql
SELECT PostHistoryId, Content, Length, LineCount, PredSimilarity
FROM PostBlockVersion
WHERE PostId=5445161 AND LocalId=2 AND PredEqual=0
ORDER BY PostHistoryId DESC;
```
Retrieve line-based difference for latest version:

```
SELECT PostHistoryId, LocalId, PredLocalId, PostBlockDiffOperationId, Text
FROM PostBlockDiff
WHERE PostHistoryId=155295527 AND LocalId=2;
```
Extracting Links From Stack Overflow Posts

• Extracted **31.4m links** from 11.6m posts, pointing to 567k different domains using a regular expression (SOTorrent 2018-05-04)

[Diagram of links being extracted]

• Extracted **6.0m links** from 438k GitHub repos, pointing to 147k posts using Google BigQuery (SOTorrent 2018-05-04)

[Diagram of links being extracted]
Retrieve links from a post version:

```sql
SELECT PostId, PostHistoryId, Domain, Url
FROM PostVersionUrl
WHERE PostHistoryId=155295527;
```

<table>
<thead>
<tr>
<th>PostId</th>
<th>PostHistoryId</th>
<th>Domain</th>
<th>Url</th>
</tr>
</thead>
<tbody>
<tr>
<td>5445161</td>
<td>155295527</td>
<td>download.oracle.com</td>
<td><a href="http://download.oracle.com/javase/8/docs/api/java/util/Scanner.html">http://download.oracle.com/javase/8/docs/api/java/util/Scanner.html</a></td>
</tr>
<tr>
<td>5445161</td>
<td>155295527</td>
<td>stackoverflow.com</td>
<td><a href="https://stackoverflow.com/users/68127/jacob-gabrielson">https://stackoverflow.com/users/68127/jacob-gabrielson</a></td>
</tr>
<tr>
<td>5445161</td>
<td>155295527</td>
<td>stackoverflow.com</td>
<td><a href="https://stackoverflow.com/users/101272/patrick">https://stackoverflow.com/users/101272/patrick</a></td>
</tr>
</tbody>
</table>
Retrieve links from GitHub repos to post:

```
SELECT PostId, RepoName, Branch, Path, FileExt, Size, Copies
FROM PostReferenceGH
WHERE PostId=5445161;
```

Referenced in 103 distinct repos

<table>
<thead>
<tr>
<th>PostId</th>
<th>RepoName</th>
<th>Branch</th>
<th>Path</th>
<th>FileExt</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>5445161</td>
<td>resource4j/resource4j</td>
<td>master</td>
<td>core/src/main/java/com/github/resource4j/object/….</td>
<td>.java</td>
<td>2077</td>
</tr>
<tr>
<td>5445161</td>
<td>yugecin/opsu-dance</td>
<td>master</td>
<td>src/itdelatrisu/opsu/Utils.java</td>
<td>.java</td>
<td>16107</td>
</tr>
<tr>
<td>5445161</td>
<td>Roojin/persian-calendar-view</td>
<td>master</td>
<td>persiancalendar/src/main/java/ir/mirrajabi/persia/….</td>
<td>.java</td>
<td>16833</td>
</tr>
<tr>
<td>5445161</td>
<td>FITeagle/sfa</td>
<td>master</td>
<td>src/main/java/org/fiteagle/north/sfa/cdm/SFA_XM/….</td>
<td>.java</td>
<td>5426</td>
</tr>
<tr>
<td>5445161</td>
<td>Steguer/ProjetAndroid</td>
<td>master</td>
<td>ProjetAndroid/libs/android-maps-….</td>
<td>.java</td>
<td>1140</td>
</tr>
<tr>
<td>5445161</td>
<td>ScottSWu/opsu</td>
<td>master</td>
<td>src/itdelatrisu/opsu/Utils.java</td>
<td>.java</td>
<td>17943</td>
</tr>
<tr>
<td>5445161</td>
<td>massimiliano76/freedomatic</td>
<td>master</td>
<td>plugins/devices/restapi-v3/src/main/java/com/….</td>
<td>.java</td>
<td>3315</td>
</tr>
</tbody>
</table>
MSR Mining Challenge 2019
Abstracts due Feb 1, 2019
Papers due Feb 6, 2019

Dataset available on Zenodo and BigQuery
Question 3

How could we better support developers struggling with licenses of online code snippets?

• What could Stack Overflow as a platform do?
• What could project owners/companies do?