Exercise Project Introduction

Lutz Prechelt, Freie Universität Berlin

V+Ü "Empirical Methods in Software Engineering"

- Domain: Quality of published SE research articles
- Example: Quality of Abstracts
- Our goal: Quality of Conclusions
- Tasks this week

Einführung in das Übungsprojekt

Lutz Prechelt, Freie Universität Berlin

V+Ü "Empirische Methoden im Software Engineering"

- Domäne: Qualität von Forschungsartikeln
- Beispiel: Qualität von Abstracts
- Unser Ziel: Qualität von Conclusions
- Aufgaben diese Woche

Outline

- 1 Domain: Quality of published SE research articles
  - 1.1 Typical article structure
  - 1.2 Recurrent quality issues
  - 1.3 Approach: Characterize, quantify, recommend
- 2 Example "qabstracts": Quality of Abstracts
  - 2.1 Example abstract & its structure
  - 2.2 Coding procedure
  - 2.3 Codebook
  - 2.4 Some results
- 3 Our topic: Quality of Conclusions
  - 3.1 Plausible roles of Conclusions sections
  - 3.2 Conjectured recurrent quality issues
  - 3.3 Status quo & plan
- 4 Tasks this week: qabstracts
  - 4.1 Setup Git repo and Python scripting
  - 4.2 Learn qabstracts codebook
  - 4.3 Practice qabstracts coding procedure

(Empirical works vs. constructive works)

- Abstract
- 1 Introduction
  - Background, motivation, questions, contributions
- 2 Related work
  - Who did what on the same or nearby questions? What do we build on?
- 3 Methods
  - What in general?
  - Details
- 4 Results
  - (possibly long; topic-dependent structure)
- 5 Threats to validity
  - What will/may have distorted our results?
  - Limitations of extent or meaning
- 6 Conclusions
  - Take-home message

- Abstract
  - Generally uninformative, lack of detail, no take-home message
- 1 Introduction
  - Question vague, contributions not stated
- 2 Related work
  - Misrepresentation of works
- 3 Methods
  - Lack of detail
- 4 Results
  - Data graveyards, lack of visualization
  - Misuse of statistics
- 5 Threats to validity
  - Trivial issues (non-problems) discussed at length
  - Important threats left out completely
- 6 Conclusions
  - Vague; no take-home message really
- OVERALL
  - Lack of relevance

Which two are most important?
Which two are easiest to assess? Or hardest?

Idea for broadly solving these issues:

- (example will follow)

- Characterize the issues in GTM fashion
  - Grounded in actual articles
  - Precise abstract concepts, exemplified by real-life quotations
- Quantify their frequency
  - Content analysis → objective results
- Recommend good practices
  - By inverting problem patterns
  - illustrated by examples of great quality

- SERQco: Software Engineering Research Quality Coalition (since 2022)
  - group of a dozen researchers
  - is currently performing the first study: qabstracts

"Desiderata is a general term for stakeholder needs, desires or preferences.
Recent experiments demonstrate that presenting desiderata as templated requirements specifications leads to less creative solutions. {{background}}

However, [they] do not establish *how* the presentation […] affects design creativity. {{gap}}

This study, therefore, aims to explore the cognitive mechanisms by which presenting [templated] desiderata […] reduces creativity during […] design. {{objective}}

42 software designers […] participated in a dialog-based protocol study.
Their interactions were transcribed and […] analyzed in two ways: (1) using inductive process coding and (2) using an a-priori coding scheme [for] fixation and critical thinking. {{method}}

Process coding shows […] 7 categories of behavior: making design moves, uncritically accepting, rejecting, grouping, questioning, assuming and considering quality criteria.
Closed coding shows that participants tend to accept given requirements and priority levels while rejecting newer, more innovative design ideas. {{result}}

Overall, the results suggest that designers fixate on desiderata presented as templated requirements specifications, hindering critical thinking.
More precisely, requirements fixation mediates the negative relationship between specification formality and creativity." {{conclusion}}

- Sample of 500 abstracts from 5 venues
- Concepts and codes are defined in a codebook
- Two coders code each abstract
  - sentence-granularity
  - in two pre-processed text files
  - working in blocks of 8 abstracts, git-based
- A checking script shows discrepancies
- The coders sort these out, either
  - agreeing on the best coding
  - or agreeing to disagree
    - using code –ignorediff
- The codings are evaluated quantitatively:
  - gaps in information,
  - weird train-of-thought etc.

Freie Universität Berlin

- 6 base rules (Section 1)
  - prefer single codings, be friendly, etc.
- The abstracts *archetype (2.1)*:
  - the typical structure of good abstracts:
    (1) Introduction
    (2) Study description
    (3) Outlook
  - Two turning points: objective, conclusion
- Main codes (2.2-2.6, 2.8):
  - as seen: background, …, conclusion
  - h-background, …: headers in structured abstracts
  - a-result, …: mere announcements
- Less common codes; extra codes; special codes (2.7, 2.9, 2.10)
  - e.g. –ignorediff
- Discrepancy-avoidance rules (Sec. 3)
- Rules for :iu suffixes (4.1-4.3)
  - result:i2: sentences with "information gaps"/"understandability gaps"

```
##### The sets of codes applied are different, please check:
abstracts/abstracts.A/GerMarLat22.txt (Lloyd, Block 9)
abstracts/abstracts.B/GerMarLat22.txt (Julian, Block 9)
[9] In the biological models we showed invariants to
developers and they confirmed that some of these point to parts of
the model that are biologically incorrect or incomplete suggesting
we may be able to use ChemFlow to improve model quality.
{{result,conclusion}} (Lloyd)
{{method,result,conclusion}} (Julian)
```

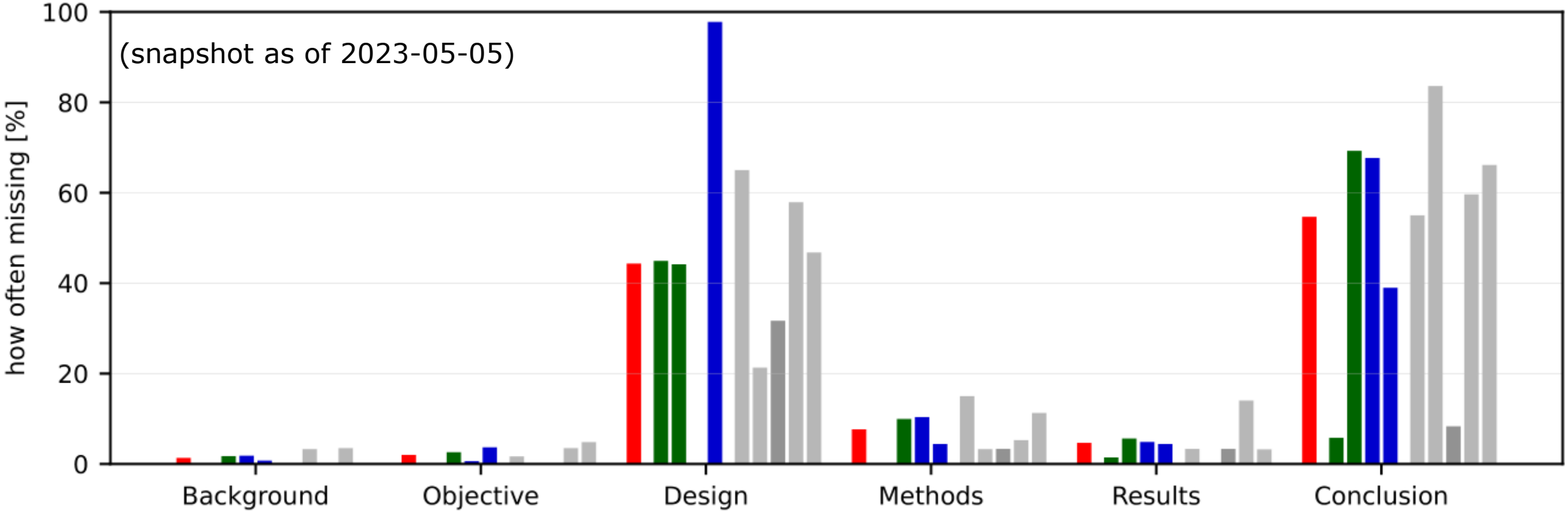

Fig. 4. How often is a topic not present at all in an abstract? The plots in each group show these different subsets of abstracts: all, structured, non-structured, design, empirical, EMSE, ICSE, IST, TOSEM, TSE

- What should a final section "Conclusion" or "Conclusions" contain?
  - A summary
    - with more information than in the abstract!
  - A conclusion ("Schlussfolgerung") that generalizes the result
    - This will be a conjecture
  - Interpretation and outlook
    - Implications for researchers
    - Future work of the authors
    - Implications for practitioners
  - (what else?)
- Quality issues we want to diagnose:
  - Summaries that add no value
  - Null-generalizations
  - Exaggerated generalizations
  - Vague implications
  - Exaggerated implications
  - …and probably others

- What we have:
  - A very preliminary codebook
- What we will do (phase 1):
  - Try out the codebook, understand its limitations
    - alone or in pairs
  - Apply GTM-style open coding and constant comparison to find improvements
    - in pairs
  - Agree on improvements
    - all together
- What we will do (phase 2):
  - Code many conclusions sections with high quality
    - alone or in pairs
- What we will do (phase 3):
  - Perform some evaluation
- But before that:
  What we will do (phase 0):
  - Get acquainted with such a study based on qabstracts

Freie Universität Berlin

- **ASAP**: Send me your ZeDat username and GitHub username
  - you cannot see the repos before we give you access
- Learn and set up git
  - everybody!
  - We will need at least
    clone --recurse-submodules,
    add, status, commit,
    push, pull --rebase, log
  - Configure a suitable name and email
  - Configure ssh key pair, ssh-agent
  - git clone --recurse-submodules
    git@git.imp.fu-berlin.de:agse/students/qabstracts-v-empir2023.git
- Work through procedure.md
  - Setup a la Section 2.2 cannot be used
  - You *must* have Python set up (Section 2.4)

Freie Universität Berlin

codebook.md:

- alone or in pairs

- Learn the 6 basic rules thoroughly
  - Really thoroughly
- Fully understand the archetype (2.1)
- Review the descriptions of the main codes (2.2-2.5)
  - come back to all these parts later whenever needed for precise coding
- Review the descriptions of the less complex other codes (2.6-2.10)
- Review the additional rules (Sec. 3)
- Understand the ideas behind (and rules for) the :iu suffixes
  - Sections 4, 4.1-4.3
  - The codes are considered objective, but the :iu suffixes are not

Freie Universität Berlin

- We start coding at the first block that is completely empty
  - Work in pairs; form them **now**
- Code one block or more by **Wednesday** evening
  - Ask all your questions about codebook or procedure by then, too
  - If you are coder B, make sure to get a different coder A on your block 2, if any
- Resolve differences with your fellow coders by **Friday** evening
  - This is highly cumbersome by email.
  - Recommendation: a meeting.
    Make an appointment early.

- Goal: Have a full understanding how qabstracts works
- qconclusions will be the same except:
  - We need to improve the codebook first
    - and it will not be mature even afterwards
  - The coding will be more difficult

Thank you!