Freie Universität Berlin

# The Normative Fallacy in Software Engineering Research

Aufhänger: "Beware the Normative Fallacy",
https://arxiv.org/abs/2005.03084

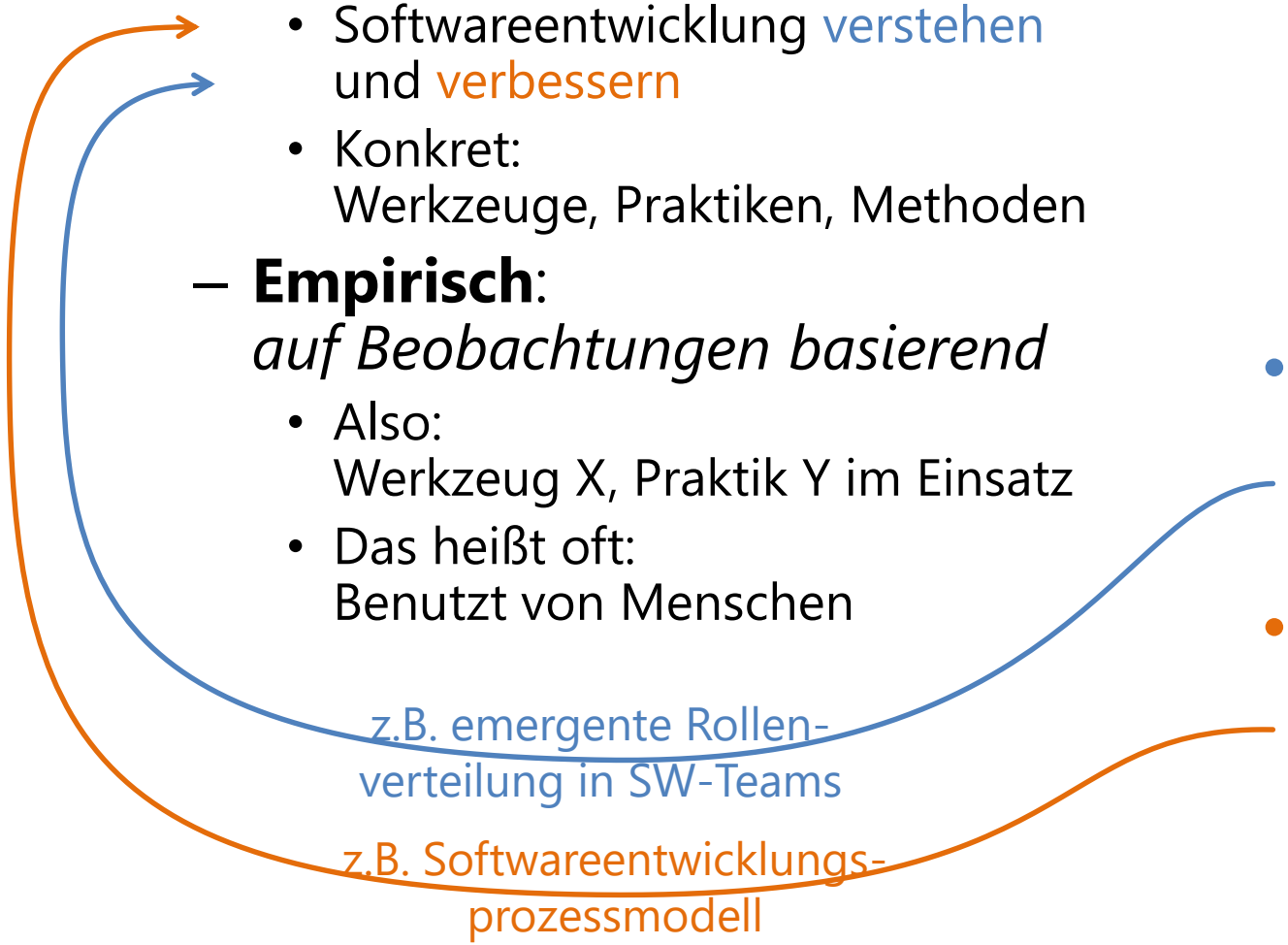
Seminar: Beiträge zum Software Engineering

2020-05-18

**Franz Zieris**
zieris@inf.fu-berlin.de

# Orientierung

- Empirische SE-Forschung
  - **Software Engineering-Forschung**
    - Softwareentwicklung verstehen und verbessern
    - Konkret: Werkzeuge, Praktiken, Methoden
  - **Empirisch**: *auf Beobachtungen basierend*
    - Also: Werkzeug X, Praktik Y im Einsatz
    - Das heißt oft: Benutzt von Menschen




- Zentral: **Theorie** [Sjøberg et al., 2008]
  - What? Hat Elemente (constructs)
  - How? Beziehung der Elemente
  - Why? Erklärung der Beziehungen
  - Where? Welcher Teil der Realität
- ***deskriptiv*** oder beschreibend
  - Realität ist gegeben
  - Elemente werden benannt
- ***normativ*** oder vorgebend (auch: "präskriptiv")
  - Realität wird geformt
  - Elemente werden postuliert

# Die "Normative Fallacy"

- Beide Arten von Theorien sind wertvoll in anwendungs-orientierter Forschung

- Problematisch wird es
  - wenn diese Unterscheidung nicht gemacht wird,
  - und normative Theorie für deskriptiven Zweck verwendet wird

**Diskussion**: Einverstanden? Änderungsvorschläge oder Ergänzungen?

- **Normative Fallacy:**
  - Die **Annahmen** in verwendeten Konzepten sind empirisch nicht validiert.
  - Diese prägen Datenerhebung und -interpretation
  - und daraus werden irreführende Schlussfolgerungen gezogen.

# Beispiel: Entscheidungsfindung [Becker, 2020]

- Normative Theorie:
  **Rational decision-making**
  - Entscheidung als Auswahl der besten Option aus mehreren Optionen, die anhand mehrerer gewichteter Kriterien geordnet wurden
- Datenerhebung:
  - *"What factors are considered when you make a decision about when to fix a defect?", "How are these factors weighted?"* [Snipes et al., 2012]
- Falsche Schlussfolgerung:
  - (fiktiv:) Für erfahrene Entwickler/innen ist Faktor X ist der wichtigste, Faktor Y der zweitwichtigste, …

- Deskriptive Theorie:
  **Naturalistic decision-making**
  - Ein mögliches Vorgehen basierend auf Erfahrung und konkreter Situation gedanklich simulieren, verwerfen oder anpassen bis passendes Vorgehen gefunden
    - Kein Ranking, nicht mehrere Kriterien
- Datenerhebung:
  - *"How did you reason about this?"*

  Vorschlag [Becker, 2020]
- Bessere Schlussfolgerung:
  - (fiktiv:) Kritischen Defekt zuerst anzugehen verringert den psychologischen Druck

# Randnotiz: Es ist kompliziert …

Becker (2020) bemängelt:

[14]. For example, when a peer-reviewed empirical study in trade-off decision making asked participants about their use of reasoning, the questions framed the situation exclusively using the empirically invalid framework of rational choice:

- "What factors are considered when you make a decision about when to fix a defect?"
- ''How are these factors weighted?"

These questions may appear perfectly normal, but only if we take the normative theories of rational choice as descriptively valid. They are not, so the questions above simply fail to adequately capture the participants reasoning [4]. Examples of

Heißt: Gewähltes Beispiel ist nicht gut, evtl. sogar falsch.

Wir arbeiten trotzdem mit der Idee "Normative Fallacy" weiter.

Snipes, Robinson, Guo, & Seaman (2012):

In the fourth section we framed a set of questions to elicit how a decision is made about when to fix a defect and the impact of the decision. We directly address RQ 2 by asking "what factors are considered when you make a decision about when to fix a defect?" and "how are these factors weighted?" We also asked questions on a specific decision path such as "how do you decide to defer a defect?" In addition to these general questions, we asked the interviewees to give examples of deferred defects, or we provided the examples selected in the first stage of the study.

There is a formal process for handling defects in ABB. The developer defines one or more fix proposals to repair the defect, describes the risk of making the change(s) and the affected test scenarios and whether a system level validation step is required.

# Beispiele für normative Konzepte

- Bereich: Technische Schulden
  - (Victor?)
  - Normatives Konzept:
    Techn. Schulden als kontext-freie statische Quellcode-**Metriken**
    - Führt zu: fraglichen MSR-Studien
  - Normatives Konzept:
    Refactoring "**nach Lehrbuch**"
- Bereich: Integrationstests
  - (Kelvin?)
  - Normatives Konzept:
    **Test-Pyramide**
    - Unit-Tests
    - Integrationstests
    - End-to-End-Tests
  - Aber:
    - Grenzen sind Praktikern unklar
      - "In welcher Testart bin ich?"
    - Fraglich wäre also: MSR-Studie

MSR: Mining Software Repositories

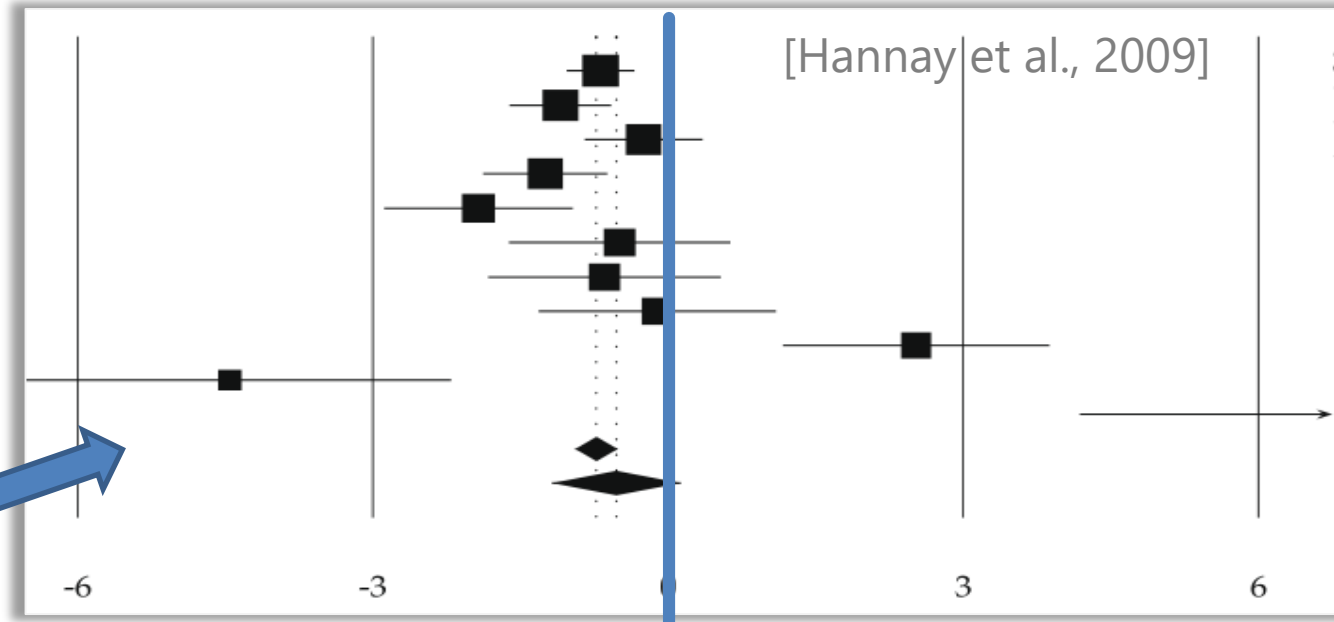
# Beispiel: Pair Programming

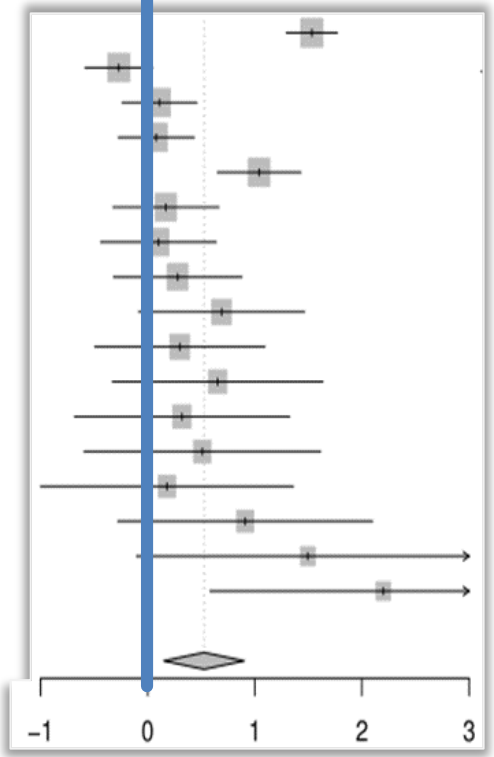
*In pair programming, two programmers jointly produce one artifact (design, algorithm, code). The two programmers are like **a unified, intelligent organism working with one mind**, responsible for every aspect of this artifact. One partner, the **driver**, controls the pencil, mouse, or keyboard and **writes the code**. The other partner continuously and actively **observes** the driver's work, watching for defects, thinking of alternatives, looking up resources, and considering strategic implications. The partners deliberately switch roles periodically.* [Williams et al., 2000]

- **Normative** Theorie, wie an anderer Stelle deutlich wird:
  - *"The pair **must** cease considering themselves as a two-programmer team and **must** start considering themselves as one coherent, intelligent organism working with one mind."* [Williams, 2000, S. 53]
  - *"One is the '**designated** driver.' This engineer has control of the mouse, keyboard, or writing utensil and is actively creating the design, code, or test."* [Williams, 2000, S. 3]

# Typische PP-Studien

- Vergleich von PP und Solo
  - Oft: Mehrere Entwicklerpaare und Solos bekommen gleiche Aufgabe → Aufwand und Qualität vergleichen
  - Auch: Vergleich von Teams aus Solos mit Teams aus Paaren
- Enthaltene Annahme:
  - *"PP ist kanonisch, zwei Entwickler/innen tun das halt und es bedeutet für alle das Gleiche."*
    - (Ansonsten würde der Vergleich mit Solos nicht viel Sinn ergeben)




[Zieris, 2020]

Man sieht: Effekt schwankt *erheblich* von Studie zu Studie

# Großes PP-Experiment

[Arisholm et al., 2007]

- ~300 bezahlte Consultants
  - 100 Solos, 100 Paare
  - Junior, Intermediate, Senior
  - Entwicklungsaufgabe:
    Simpler vs. komplizierter Entwurf
- Hypothesen:
  - Mehr PP-Nutzen für schwieriges Setting (wenig Erfahrung, komplizierter Entwurf)
- Übliche implizite Annahmen:
  1. PP ist kanonisch
  2. Paare sind um Faktor X schneller (oder langsamer) und produzieren um Y% bessere (oder schlechtere) Qualität

- Ergebnisse: Im Mittel sind Paare schneller und liefern bessere Qualität
  - Aber: im fairen Vergleich kein signifikanter Unterschied
- Blick in die Daten zeigt andere Effekte:
  - Paare unbeeinflusst von Komplexität
  - vorhersagbarere Bearbeitungsdauer
    - Schwankende Dauer bei Solos
    - Mit Erfahrung stetig sinkend bei Paaren
  - konstante Qualität
    - Mit Erfahrung stetig steigende Qualität bei Solos
    - Konstante (!) Qualität bei Paaren
- Wegen Annahme 2 nicht Teil der Analyse

# Ergebnisse im Detail [Arisholm et al., 2007]

- Dauer (Minuten)

| Scope | | | Unadjusted Means | | |
|---|---|---|---|---|---|
| Prog. Cat | Term | CS | Ind | Pair | % diff |
| All | PP | CC+DC | 87 | 63 | -28 % |
| | PPxCS | CC | 98 | 64 | -35 % |
| | | DC | 72 | 62 | -15 % |
| Junior | PP | CC+DC | 92 | 81 | -12 % |
| | PPxCS | CC | 95 | 85 | -10 % |
| | | DC | 86 | 78 | -9 % |
| Intermed | PP | CC+DC | 104 | 61 | -41 % |
| | PPxCS | CC | 113 | 60 | -47 % |
| | | DC | 87 | 63 | -27 % |
| Senior | PP | CC+DC | 74 | 51 | -31 % |
| | PPxCS | CC | 87 | 53 | -39 % |
| | | DC | 61 | 49 | -20 % |

- Qualität (Anteil korrekter Lsg.)

| Scope | | | Unadjusted Means | | |
|---|---|---|---|---|---|
| Prog. Cat | Term | CS | Ind | Pair | Odds ratio |
| All | PP | CC+DC | 60 % | 82 % | 3.01 |
| | PPxCS | CC | 68 % | 80 % | 1.93 |
| | | DC | 51 % | 83 % | 4.68 |
| Junior | PP | CC+DC | 48 % | 84 % | 5.60 |
| | PPxCS | CC | 63 % | 83 % | 3.00 |
| | | DC | 33 % | 85 % | 11.00 |
| Intermed | PP | CC+DC | 53 % | 80 % | 3.53 |
| | PPxCS | CC | 65 % | 80 % | 2.18 |
| | | DC | 40 % | 80 % | 6.00 |
| Senior | PP | CC+DC | 75 % | 82 % | 1.48 |
| | PPxCS | CC | 76 % | 79 % | 1.15 |
| | | DC | 74 % | 84 % | 1.90 |

(Farben korrespondieren mit vorheriger Folie)

# Referenzen


- Arisholm et al. (2007): Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise (TSE)
- Becker (2020): Beware the Normative Fallacy (arXiv)
- Campbell (1970): The Normative Fallacy (Philosophical Quarterly)
- Hannay et al. (2009): The effectiveness of pair programming: A meta-analysis (IST)
- Sjøberg et al. (2008): Building Theories in Software Engineering
- Snipes et al. (2012): Defining the Decision Factors for Managing Defects: A Technical Debt Perspective (MTD'12)
- Williams (2000): The Collaborative Software Process
- Williams et al. (2000): Strengthening the Case for Pair Programming (IEEE Software)
- Zieris (2020): Qualitative Analysis of Knowledge Transfer in Pair Programming