Linus Ververs
AG Software Engineering
Institut für Informatik
Fachbereich Mathematik & Informatik

Freie Universität Berlin

# Qualitative Forschung zu Paar-Programmierung

1. Paar-Programmierung
2. GTM & Forschung der AGSE
3. Forschungsprozess & Ergebnisse
4. Ausblick & Meta-Kommentare

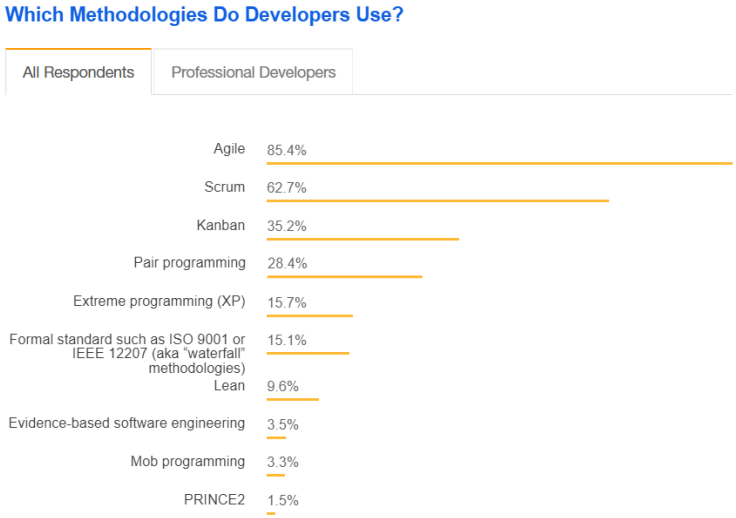
# Warum forschen wir?

“To but it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, **programming has become an equally gigantic problem**. – Edsger W. Dijkstra 1972 [1]

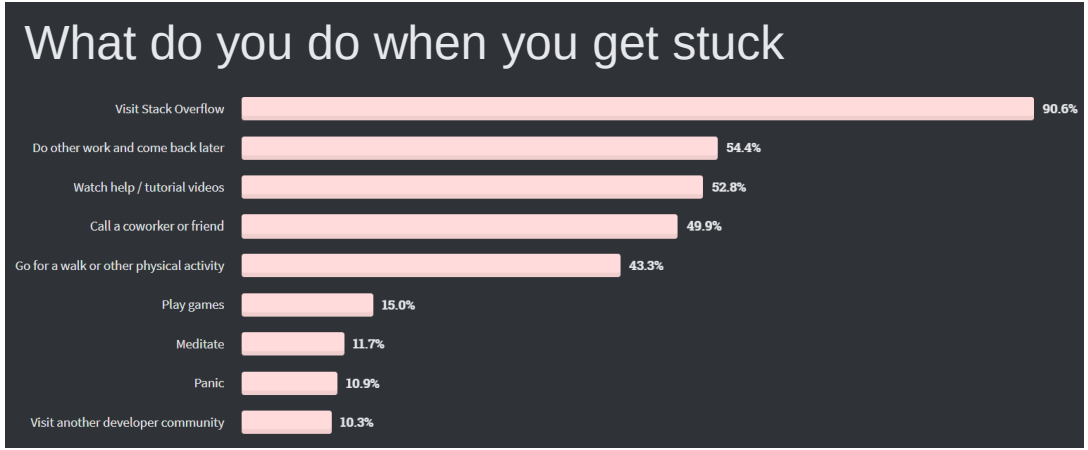
→ Ziel der Software Engineering Forschung:
Evidenzbasierte Empfehlungen zur Verbesserungen des Softwareentwicklungsprozesses

# Was ist Paar-Programmierung?

“Pair programming (PP) is the practice of two developers working closely together on one computer to solve a technical task.” – Franz Zieris [2]



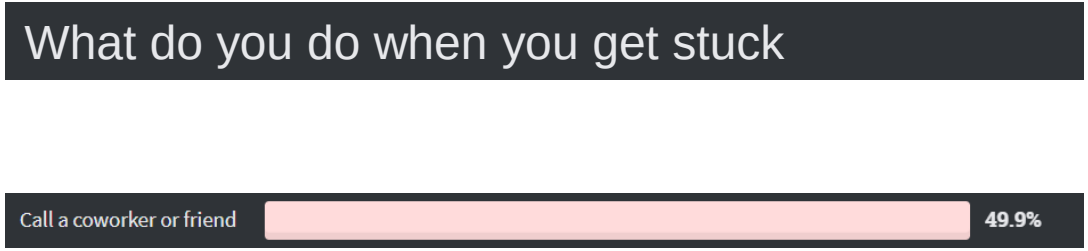

Stackoverflow.com-Umfrage 2018 [3]; n=58,981




Stackoverflow.com-Umfrage 2020 [4]; n≈65.000

# Was ist Paar-Programmierung?

“Pair programming (PP) is the practice of two developers working closely together on one computer to solve a technical task.” – Franz Zieris [2]

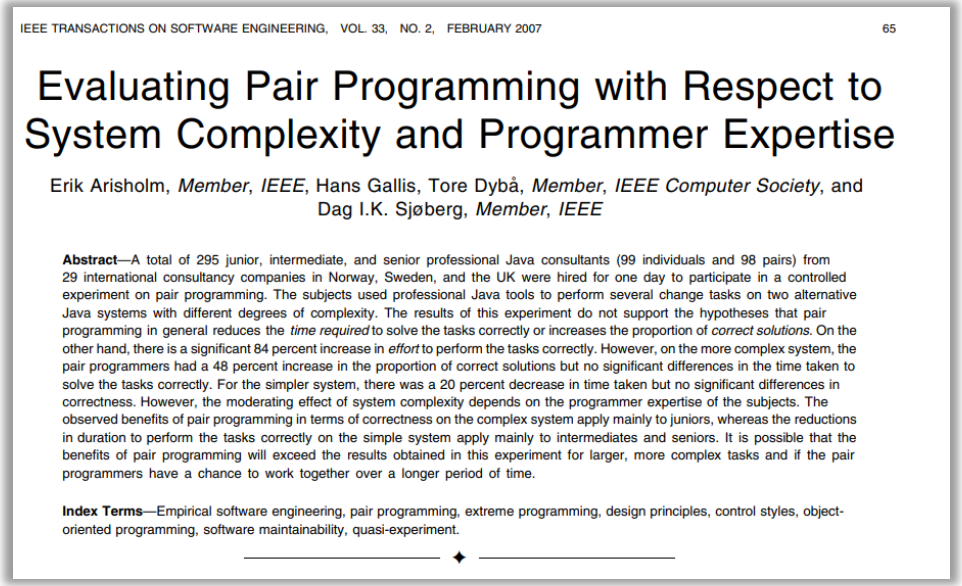




→ Vermutung: Das führt in vielen Fällen auch zu Pair-Programmierung

Warum diese Dissonanz???
→ PP als Practice vs. PP als Workmode
→ „Ach, das ist schon Pair Programming?!“

# Forschung zu Paar-Programmierung

- Experimentelle Studie mit 295 Consultants
- Einteilung in Junior, Intermediate & Senior
- 99 Solo-Entwickler vs. 98 Paare
- Bearbeitung von 2 unterschiedlich komplexen Aufgaben



IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 33, NO. 2, FEBRUARY 2007 65

## Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise

Erik Arisholm, *Member, IEEE*, Hans Gallis, Tore Dybå, *Member, IEEE Computer Society*, and Dag I.K. Sjøberg, *Member, IEEE*

**Abstract**—A total of 295 junior, intermediate, and senior professional Java consultants (99 individuals and 98 pairs) from 29 international consultancy companies in Norway, Sweden, and the UK were hired for one day to participate in a controlled experiment on pair programming. The subjects used professional Java tools to perform several change tasks on two alternative Java systems with different degrees of complexity. The results of this experiment do not support the hypotheses that pair programming in general reduces the *time required* to solve the tasks correctly or increases the proportion of *correct solutions*. On the other hand, there is a significant 84 percent increase in *effort* to perform the tasks correctly. However, on the more complex system, the pair programmers had a 48 percent increase in the proportion of correct solutions but no significant differences in the time taken to solve the tasks correctly. For the simpler system, there was a 20 percent decrease in time taken but no significant differences in correctness. However, the moderating effect of system complexity depends on the programmer expertise of the subjects. The observed benefits of pair programming in terms of correctness on the complex system apply mainly to juniors, whereas the reductions in duration to perform the tasks correctly on the simple system apply mainly to intermediates and seniors. It is possible that the benefits of pair programming will exceed the results obtained in this experiment for larger, more complex tasks and if the pair programmers have a chance to work together over a longer period of time.

**Index Terms**—Empirical software engineering, pair programming, extreme programming, design principles, control styles, object-oriented programming, software maintainability, quasi-experiment.

„However, on the **more complex system**, the **pair programmers had a 48 percent increase** in the proportion of **correct solutions** but no significant differences in the time taken to solve the tasks correctly.”[5]

# Experimente im Software Engineering

- Halte alle Eingangsvariablen konstant bis auf eine
- Welchen Einfluss hat die Änderung dieser einen Variable auf die Ausgangsvariablen?

- Experimente sind das Ding in den Naturwissenschaften
- Problematisch sobald Menschen ins Spiel kommen

# Experimente im Software Engineering

- Halte alle Eingangsvariablen konstant bis auf eine
- Welchen Einfluss hat die Änderung dieser einen Variable auf die Ausgangsvariablen?

- Experimente
  Natu
- enschen ins

Experimente im Software Engineering:
Hohe **Glaubwürdigkeit** auf Kosten der **Relevanz**!

# Qualitative Feldforschung der AGSE

Betrachtung / Untersuchung der Softwareentwicklung in der Praxis.

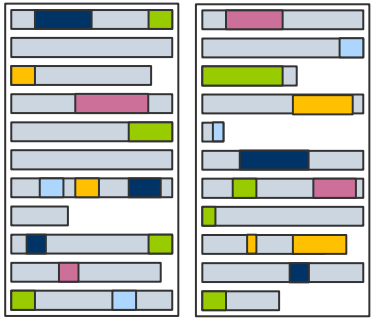
Qualitative, explorative Forschung um positive / negative Phänomene zu identifizieren & zu erklären.

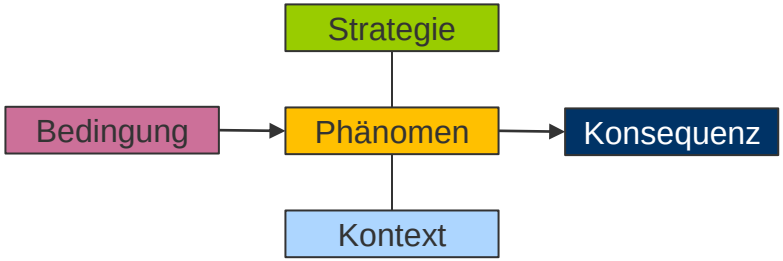
Formulieren von Theoriegebilden & Ableiten von Empfehlungen für die Softwareentwicklung in der Praxis.

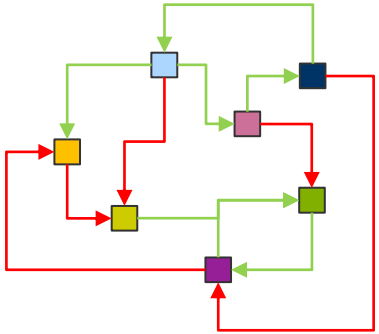
Verwendung der Grounded Theory Methodology aus den Sozialwissenschaften zur Bildung fundierter Theorien.

# Grounded Theory Methodology*

Offenes Codieren → Axiales Codieren → Selektives Codieren





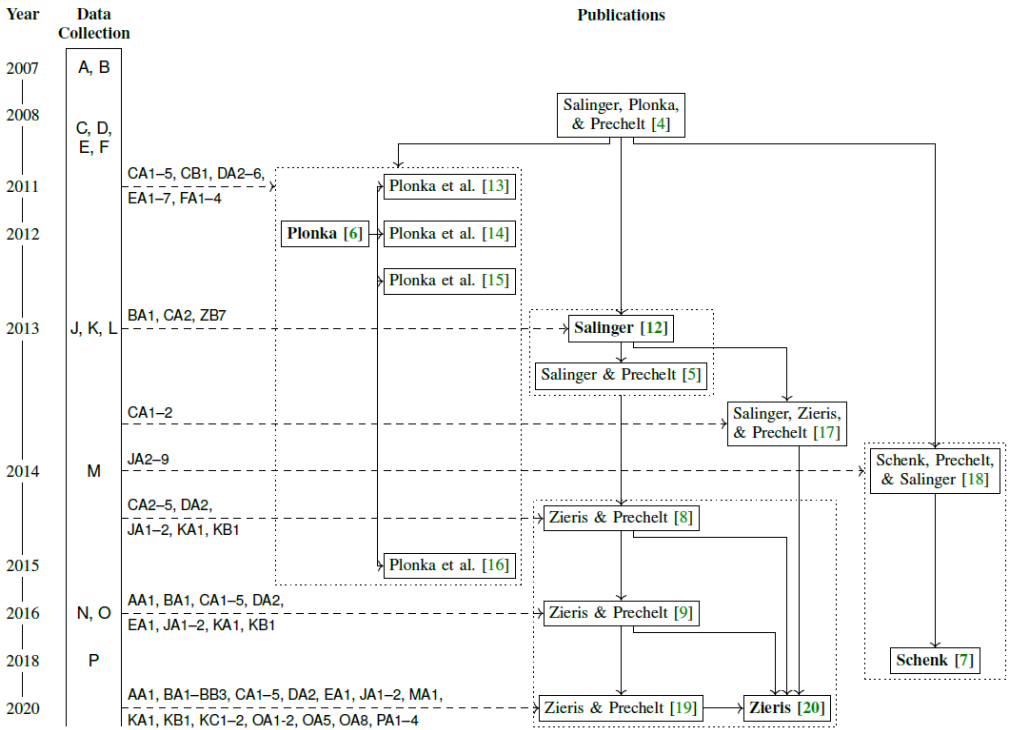
Theoretische Stichproben | Theoretische Sensitivität | Theoretische Sättigung

Theoretisches Codieren | Ständiges Vergleichen

*Hier sehr vereinfacht und verkürzt dargestellt.

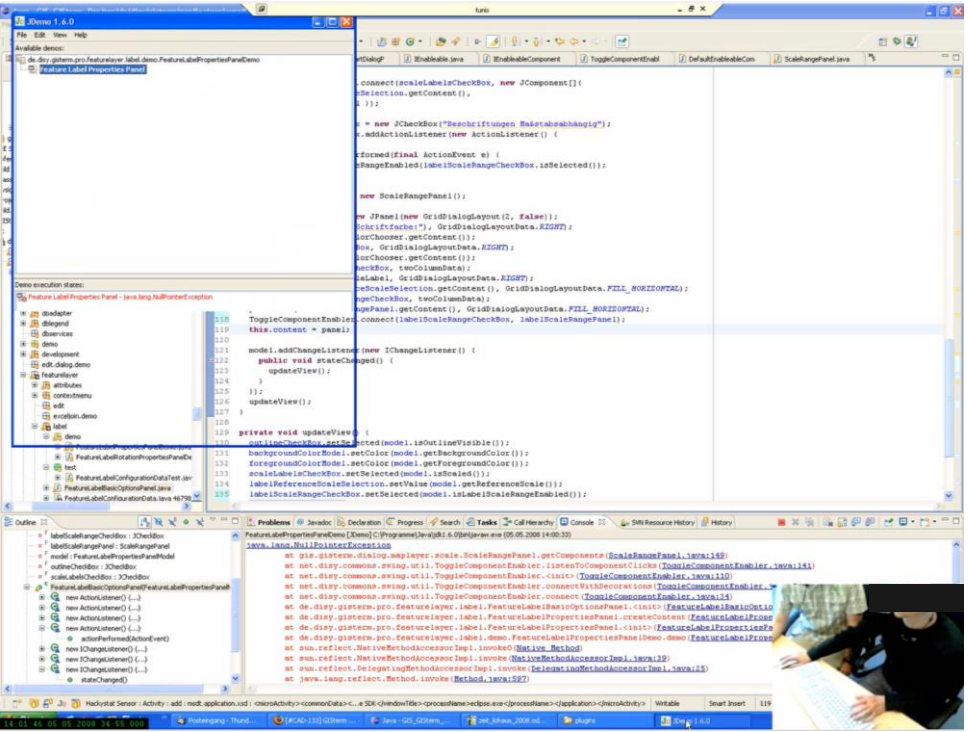
# PPind: Datensatz der Arbeitsgruppe [6]

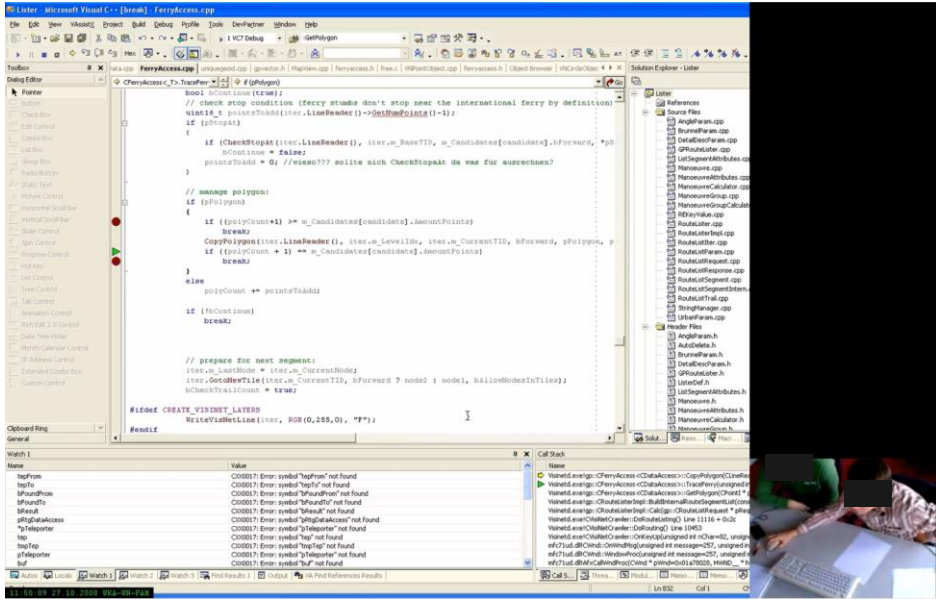
- Sammlung von PP-Sitzungen aus der Praxis seit 2007
- Insgesamt 67 Sitzungen mit ca. 100 Stunden Videomaterial
- Durchschnittliche Sitzung hat 1,5 Stunden
- Aus 13 Firmen mit 57 Entwicklern
- Klassische PP & Distributed PP




Grafik aus [6]

# Beispiel der Daten





# Base Layer für PP-Forschung (S. Salinger) [7]

## product-oriented concepts

| | |
|---|---|
| ***amend_design*** Extend a given proposal regarding the structure and content of the program without rejecting the proposal. | ***ask_design*** Ask for a concrete proposal regarding the structure and content of the program. |
| ***challenge_design*** Reject a given proposal regarding the structure and content of the program and make an alternative proposal instead. | ***agree_design*** Signal agreement with a given proposal regarding the structure and content of the program. |
| ***decide_design*** Select one from among several alternative proposals regarding the structure and content of the program. | ***propose_design*** Make one or several alternative proposals regarding the structure and content of the program. |
| ***disagree_design*** Reject a given proposal regarding the structure and content of the program without making an alternative proposal. | |
| | ***remember_requirement*** Remind the pair of a given (pre-specified) functional or non-functional requirement of the program. |
| ***challenge_requirement*** Reject a given or proposed requirement and propose an alternative one instead. | ***agree_requirement*** Signal agreement with a given or proposed requirement. |
| | ***propose_requirement*** Propose one or several alternative program characteristics that should be considered to be a requirement. |

## miscellaneous

| | |
|---|---|
| ***mumble_sth*** Make an incomprehensible utterance (highly fragmentary or acustically unclear). | ***say_off topic*** Make an utterance that has nothing to do with solving the programming task. |

## process-oriented concepts

| | | |
|---|---|---|
| ***amend_step*** Extend a given proposal regarding the next tactical work step without rejecting the proposal. | ***ask_step*** Ask for a concrete proposal regarding the next tactical work step. | ***explain_completion*** Make a statement regarding the degree of completion of the current tactical work step. |
| ***challenge_step*** Reject a given proposal regarding the next tactical work step and make an alternative proposal instead. | ***agree_step*** Signal agreement with a given proposal regarding the next tactical work step. | ***agree_completion*** Signal agreement with a statement regarding the degree of completion of the current tactical work step. |
| ***decide_step*** Select one from among several alternative proposals regarding the next tactical work step. | ***propose_step*** Make one or several alternative proposals regarding the next tactical work step. | ***challenge_completion*** Reject a statement regarding the degree of completion of the current tactical work step and make an alternative statement. |
| ***disagree_step*** Reject a given proposal regarding the next tactical work step without making an alternative proposal. | | ***explain_state*** Make a statement regarding the degree to which the current strategy or work plan has been worked through. |
| ***amend_strategy*** Extend a proposed strategy or work plan without rejecting it. | ***ask_strategy*** Ask for a concrete proposal regarding the strategy or work plan to be chosen. | ***agree_state*** Signal agreement with a statement regarding the degree to which the current strategy or work plan has been worked through. |
| ***challenge_strategy*** Reject a given proposal regarding the strategy or work plan and make an alternative proposal instead. | ***agree_strategy*** Signal agreement with a given proposal regarding the strategy or work plan. | ***challenge_state*** Reject a statement regarding the degree to which the current strategy or work plan has been worked through and make an alternative statement. |
| ***decide_strategy*** Select one from among several alternative proposed strategies or work plans. | ***propose_strategy*** Propose one or several alternative strategies or work plans. | ***propose_todo*** Suggest that a certain work item will need to be taken care of later in the process. |
| ***disagree_strategy*** Reject a given proposal regarding the strategy or work plan without making an alternative proposal. | | ***agree_todo*** Signal agreement with a statement saying that a certain work item will need to be taken care of later in the process. |

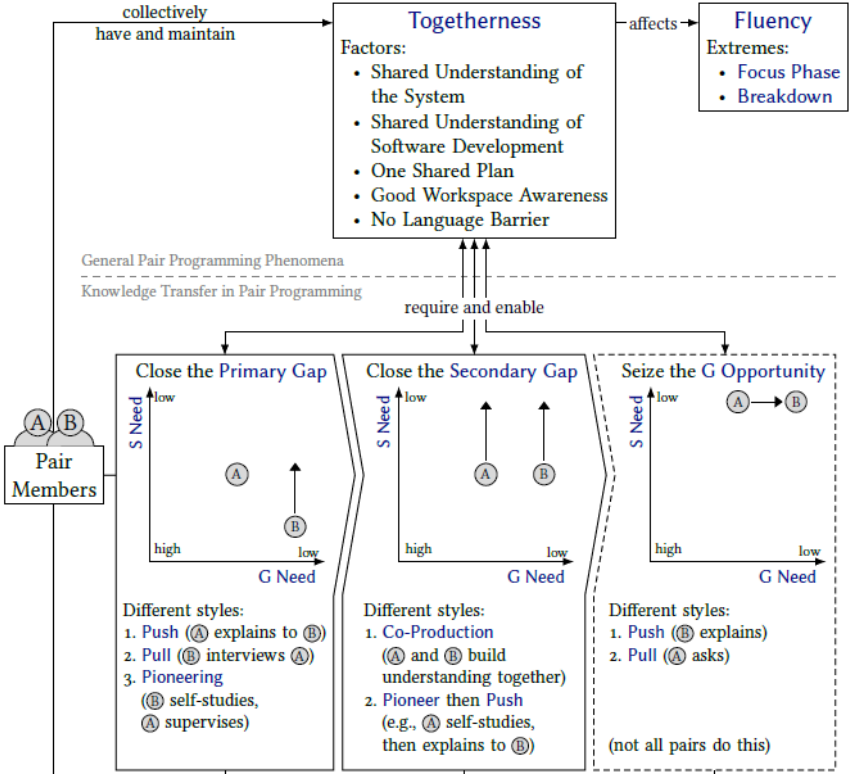
## universal concepts

| | | | |
|---|---|---|---|
| ***explain_gap in knowledge*** Verbalize that certain knowledge is not possessed by either member of the pair. | ***agree_gap in knowledge*** Signal agreement with a given gap in knowledge. | ***explain_standard of knowledge*** Explain or recapitulate one's own level of knowledge with respect to a certain topic. | ***ask_standard of knowledge*** Ask the partner for his/her level of knowledge with respect to a certain topic. |
| | | ***ask_knowledge*** Ask the partner for information of type 'declarative knowledge'. | ***stop_activity*** Suggest to stop or abort the current HCI or HEI activity. |
| ***explain_finding*** Verbalize a new insight; this includes interpreting an observed event. | ***propose_hypothesis*** Formulate a hypothesis or conjecture, e.g. regarding a property of the program, or the environment. | ***explain_knowledge*** Transfer information to the partner that is assumed to be correct declarative knowledge. | ***think aloud_activity*** Verbalize aspects of one's own current HCI or HEI activity. |
| ***agree_finding*** Signal agreement with a verbalized insight or interpretation. | ***agree_hypothesis*** Signal agreement with a given hypothesis or conjecture. | ***agree_knowledge*** Signal agreement (i.e. judge as correct) knowledge stated by the partner. | ***agree_activity*** Signal agreement with all or part of the current HCI or HEI activity. |
| ***challenge_finding*** Reject the content of a verbalized insight or interpretation and suggest an alternative one. | ***challenge_hypothesis*** Reject a given hypothesis or conjecture and formulate an alternative one. | ***challenge_knowledge*** Declare transfered knowledge as fully, partially, or potentially wrong by opposing it with one's own knowledge. | ***challenge_activity*** Reject all or part of the current HCI or HEI activity and suggest an alternative activity. |
| ***disagree_finding*** Declare transfered finding as fully, partially, or potentially wrong without explaining why. | ***disagree_hypothesis*** Reject a given hypothesis or conjecture. | ***disagree_knowledge*** Declare transfered knowledge as fully, partially, or potentially wrong without explaining why. | ***disagree_activity*** Reject all or part of the current HCI or HEI activity. |
| ***amend_finding*** Extend a verbalized insight or interpretation without rejecting it. | ***amend_hypothesis*** Extend a given hypothesis or conjecture without rejecting it. | | ***amend_activity*** Propose an extension to the current HCI or HEI activity. |

**facade concepts**

# Wissenstransfer in der PP (F. Zieris) [2]

- Togetherness als fundamentales Element, um die Vorteile von PP zu nutzen.
- Hohe Togetherness führt zu Focus Phases, niedrige zu Breakdowns
- Unterscheidung von S- & G-Knowledge
- S-Knowledge wird zuerst auf den für die Aufgabe notwendigen gleichen Stand gebracht
- Danach wird noch fehlendes S-Knowledge gemeinsam erarbeitet
- ggf. wird G-Knowledge transferiert

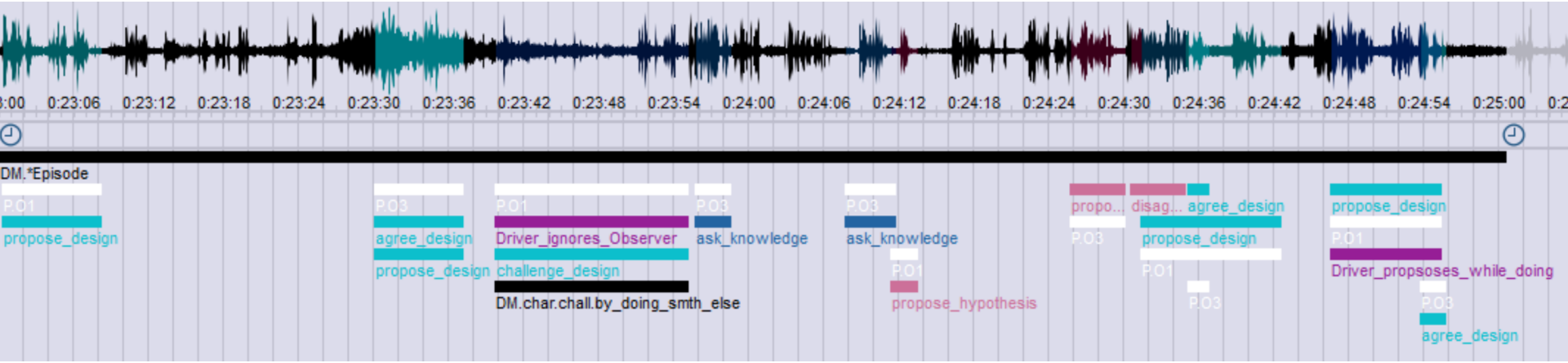



Grafik aus [2]

# Entscheidungsfindung in der PP (that‘s me)

- Gestartet im April 2022
- Explorativer Ansatz nach GTM
- Timeline bis jetzt:
  - Einarbeiten in die Arbeiten der Arbeitsgruppe / GTM
  - Gefühl für die Daten bekommen
  - Auffällige Phänomene identifizieren und Hypothesen bilden → Theoretische Sensitivität
  - Hypothesen validieren und anreichern durch Betrachtung weiterer Daten

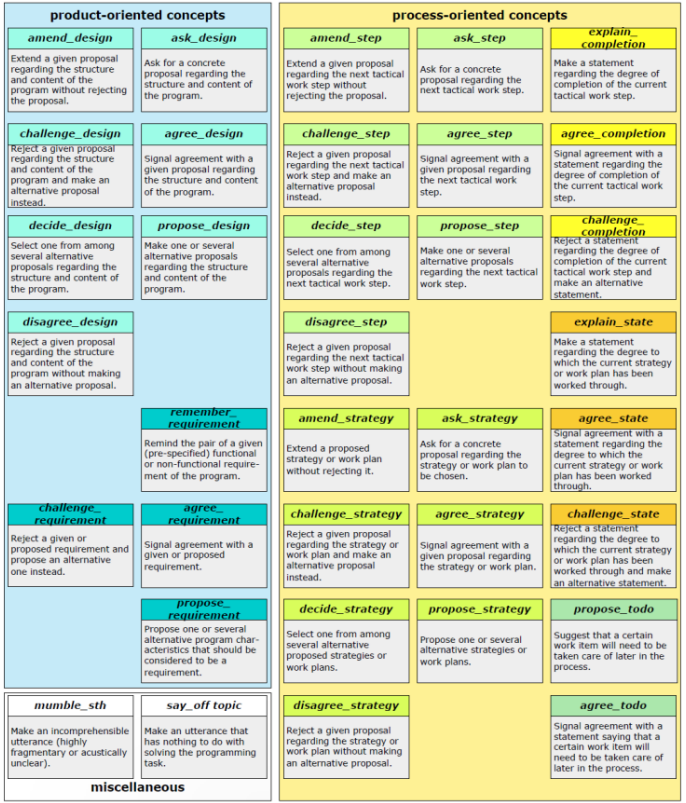
# Codierungsbeispiele

# Nuancen der Base-Layer Konzepte

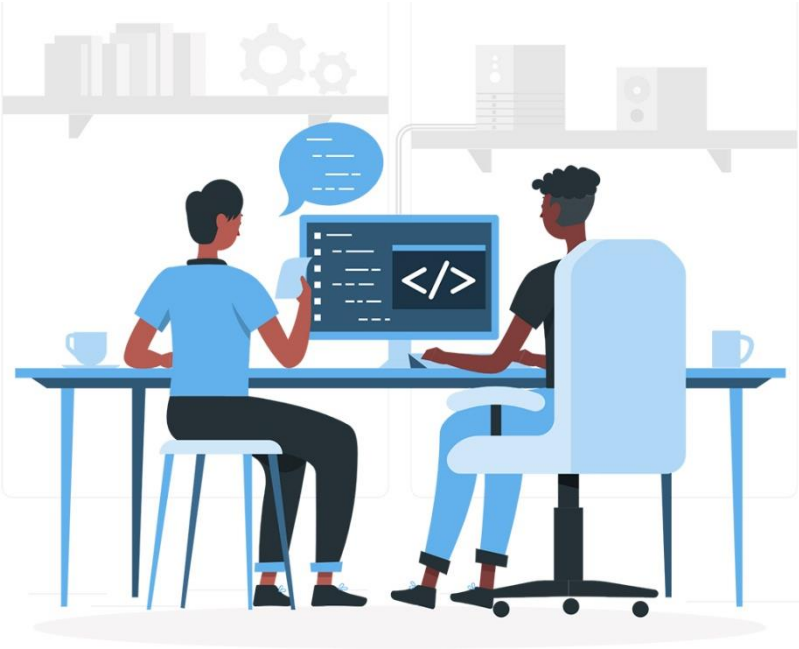
- Entscheidungsepisoden beginnen mit **propose_***, gefolgt von **disagree_***; **challange_*** ; bis zu einem **agree_***
- Es gibt unterschiedliche Arten des propse; challange, disagree, agree:
  - Mit oder ohne Begründung
  - „Zustimmung“ durch keinen Widerspruch
  - Propose by doing
  - Challange by doing something else

→ Für den Entscheidungsprozess spielt es eine enorme Rolle wer Driver bzw. Observer ist!



# Driver-Observer Dynamik im Fokus

- Wie macht der Driver Vorschläge:
  - Beginnt einfach zu coden
  - Macht Vorschlag während der Umsetzung
  - Macht Vorschlag bevor er mit der Umsetzung beginnt
- Und wie reagiert der Observer darauf?
  - Keine Reaktion
  - Intervenierend
  - Etc.




→ **Autoritäres Verhalten ist schädlich für die Togetherness**

# Autoritäres Verhalten in der PP

- 1 Sitzung mit einem autoritären Driver
  - Unterbricht den Observer, bittet um Ruhe, arbeitet alleine und teilt sein Wissen nicht, nimmt dem „Observer“ die Tastatur weg
  - Observer wird aus der Sitzung gedrängt, macht wenig Nachfragen, hat wenig Einfluss auf die Entscheidungssituationen
  - Vorteile der Paar-Programmierung gehen verloren

- 1 Sitzung mit autoritären Observer:
  - Belehrend, Macht sehr kleinteilige Handlungsvorschläge, sitzt zurückgelehnt
  - Driver reagiert zunehmend gereizt

# Aktueller Stand / Ausblick

- Bisher wenig autoritäres Verhalten in unseren Daten gefunden

- Nächste Schritte
  - Sitzungen angucken und explizit auf solches Verhalten achten
  - Neue Daten beschaffen, mit Fokus auf Entwickler, die normalerweise keine Paar-Programmierung machen
  - Auswirkungen von autoritärem Verhalten genauer analysieren

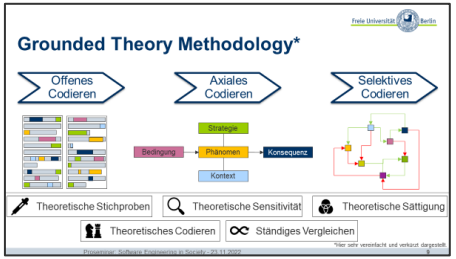
- Probleme:
  - Datenbeschaffung
  - Entscheidungssituationen sind generell sehr schwer zu analyiseren

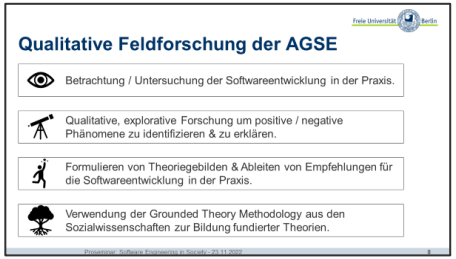
# Meta-Kommentare

- Wer diesen Vortrag in Teilen spannend fand → Masterveranstaltung: **Empirische Methoden im Software Engineering** von Prof. Dr. Lutz Prechelt
  - Vorlesung: Experimente, GTM, Umfragen, Fallstudien, Datenauswertung
  - Übung: Datenauswertung; Umfrage / GTM

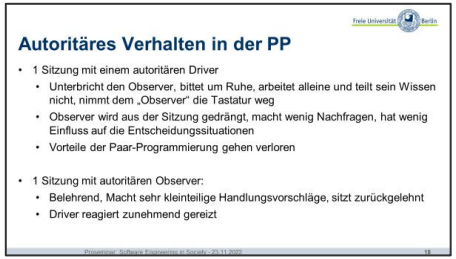
Meta-Kommentar zu den Folien:

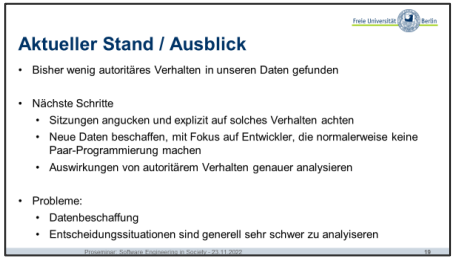
<u>Gute Folien (für einen Vortrag):</u>




<u>Weniger gute Folien:</u>



# Quellen

[1] **Dijkstra, Edsger W.** "The humble programmer." *Communications of the ACM* 15.10 (1972): 859-866.

[2] **Zieris, Franz.** *Qualitative analysis of knowledge transfer in pair programming*. Diss. 2020.

[3] Stack Overflow Developer Survey 2018

[4] Stack Overflow Developer Survey 2020

[5] **Arisholm, Erik, et al.** "Evaluating pair programming with respect to system complexity and programmer expertise." *IEEE Transactions on Software Engineering* 33.2 (2007): 65-86.

[6] **Zieris, Franz, and Lutz Prechelt.** "PP-ind: A repository of industrial pair programming session recordings." *Context* 50 (2020): 9.

[7] **Salinger, Stephan, and Lutz Prechelt.** *Understanding Pair Programming: The Base Layer*. BoD–Books on Demand, 2013.

# Bildquellen

1.1570789.figures.f1.jpeg (650×826) (scitation.org)

Pair Programming: Why You Should Care About It and How To Do It Remotely - DistantJob - Remote Recruitment Agency