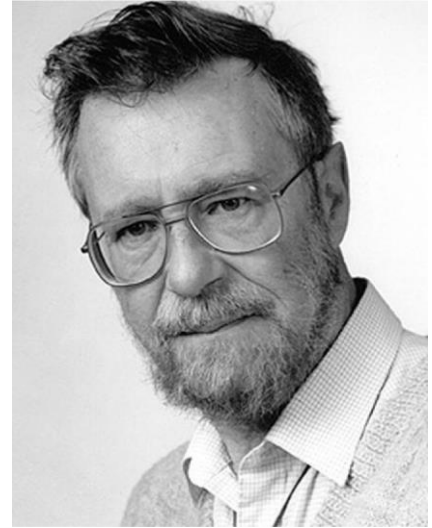


Qualitative Forschung zu Paar-Programmierung

1. Paar Programmierung
2. Grounded Theory Methodology
3. Forschung der AGSE
4. Ergebnisse + Beispiele aus den letzten 8 Monaten
5. Forschungsprozess & Ausblick

Warum forschen wir?

“To put 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]

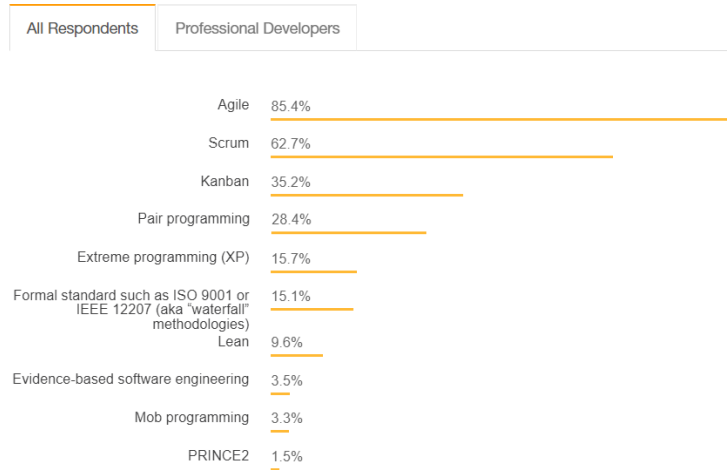


Ziel der Software Engineering Forschung: Evidenzbasierte Empfehlungen zur Verbesserung des Softwareentwicklungsprozesses in der Praxis.

Was ist Paar-Programmierung?

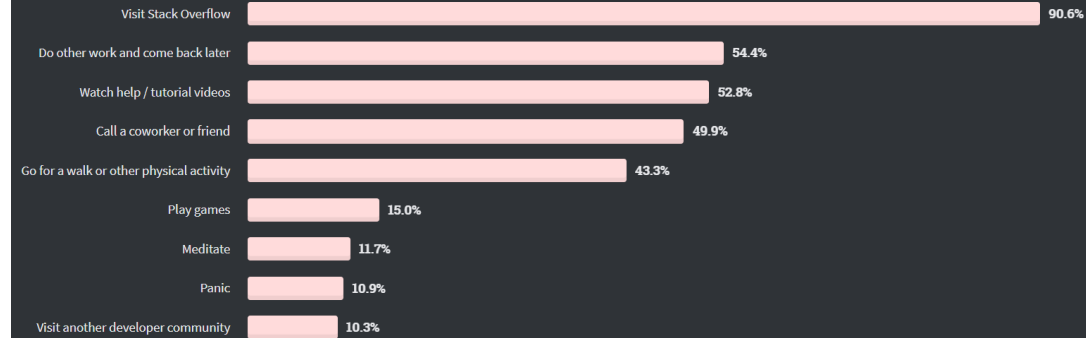
“Pair programming (PP) is the practice of two developers working closely together on one computer to solve a technical task.” – Franz Zieris [2]

Which Methodologies Do Developers Use?



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

What do you do when you get stuck



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]

Which Methodologies Do Developers Use?

What do you do when you get stuck

Pair programming 28.4%

Call a coworker or friend

49.9%

→ Vermutung: Das führt in vielen Fällen auch zu Paar-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]

Forschung zu Paar-Programmierung

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

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

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

65

Evaluating Pair Programming on System Complexity

Erik Arisholm

Society, and

consultants (99 individuals and 98 pairs) from the same company were hired for one day to participate in a controlled experiment. They were asked to perform several change tasks on two alternative Java tools. The results of this experiment do not support the hypotheses that pair programming increases the proportion of correct solutions or increases the proportion of correct solutions. On the other hand, there was a significant 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 the proportion of correct solutions. 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]

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*

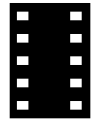


*Hier sehr vereinfacht und verkürzt dargestellt.

PPind: Datensatz der Arbeitsgruppe [6]



Sammlung von PP-Sitzungen aus der Praxis seit 2007



Insgesamt 67 Sitzungen mit ca. 100 Stunden Videomaterial



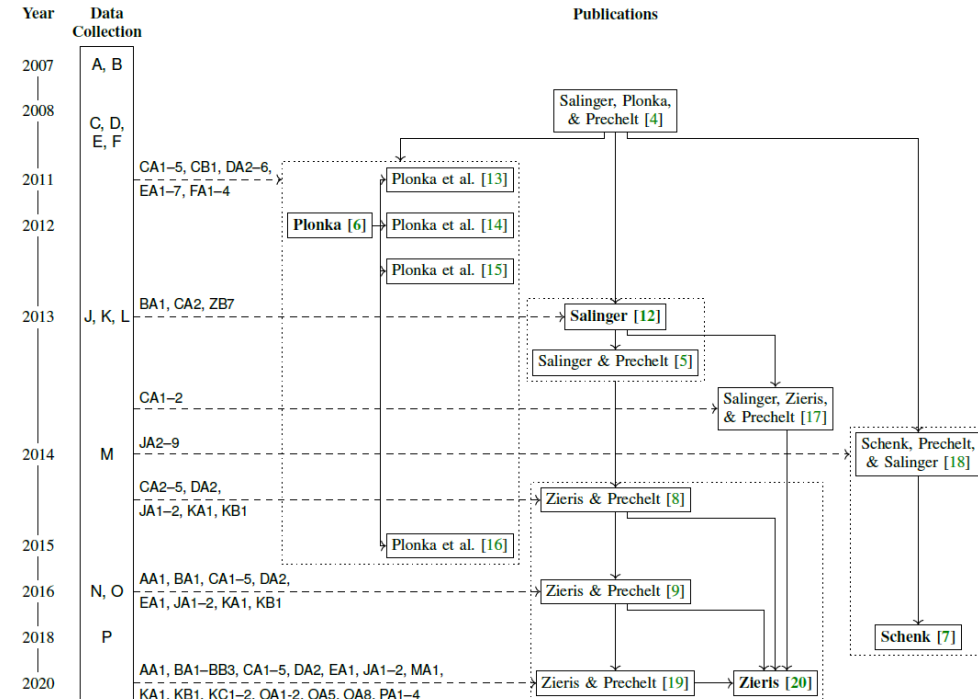
Aus 13 Firmen mit 57 Entwicklern



Diverse Technologien, Zielsetzungen und Paar-Konstellationen



Klassische (vor-Ort) PP & Distributed (virtuelle) PP



Grafik aus [6]

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

product-oriented concepts		process-oriented concepts			universal concepts			
amend_design	ask_design	amend_step	ask_step	explain_completion	explain_gap_in_knowledge	agree_gap_in_knowledge	explain_standard_of_knowledge	ask_standard_of_knowledge
Extend a given proposal regarding the structure and content of the program without rejecting the proposal.	Ask for a concrete proposal regarding the structure and content of the program.	Extend a given proposal regarding the next tactical work step without rejecting the proposal.	Ask for a concrete proposal regarding the next tactical work step.	Make a statement regarding the degree of completion of the current tactical work step.	Verbalize that certain knowledge is not possessed by either member of the pair.	Signal agreement with a given gap in knowledge.	Explain or recapitulate one's own level of knowledge with respect to a certain topic.	Ask the partner for his/her level of knowledge with respect to a certain topic.
challenge_design	agree_design	challenge_step	agree_step	agree_completion			ask_knowledge	stop_activity
Reject a given proposal regarding the structure and content of the program and make an alternative proposal instead.	Signal agreement with a given proposal regarding the structure and content of the program.	Reject a given proposal regarding the next tactical work step and make an alternative proposal instead.	Signal agreement with a given proposal regarding the next tactical work step.	Signal agreement with a statement regarding the degree of completion of the current tactical work step.			Ask the partner for information of type 'declarative knowledge'.	Suggest to stop or abort the current HCI or HEI activity.
decide_design	propose_design	decide_step	propose_step	challenge_completion	explain_finding	propose_hypothesis	explain_knowledge	think_aloud_activity
Select one from among several alternative proposals regarding the structure and content of the program.	Make one or several alternative proposals regarding the structure and content of the program.	Select one from among several alternative proposals regarding the next tactical work step.	Make one or several alternative proposals regarding the next tactical work step.	Reject a statement regarding the degree of completion of the current tactical work step and make an alternative statement.	Verbalize a new insight; this includes interpreting an observed event.	Formulate a hypothesis or conjecture, e.g. regarding a property of the program, or the environment.	Transfer information to the partner that is assumed to be correct declarative knowledge.	Verbalize aspects of one's own current HCI or HEI activity.
disagree_design		disagree_step		explain_state	agree_finding	agree_hypothesis	agree_knowledge	agree_activity
Reject a given proposal regarding the structure and content of the program without making an alternative proposal.		Reject a given proposal regarding the next tactical work step without making an alternative proposal.		Make a statement regarding the degree to which the current strategy or work plan has been worked through.	Signal agreement with a verbalized insight or interpretation.	Signal agreement with a given hypothesis or conjecture.	Signal agreement (i.e. judge as correct) knowledge stated by the partner.	Signal agreement with all or part of the current HCI or HEI activity.
	remember_requirement	amend_strategy	ask_strategy	agree_state				
	Remind the pair of a given (pre-specified) functional or non-functional requirement of the program.	Extend a proposed strategy or work plan without rejecting it.	Ask for a concrete proposal regarding the strategy or work plan to be chosen.	Signal agreement regarding the degree to which the current strategy or work plan has been worked through.				
challenge_requirement	agree_requirement	challenge_strategy	agree_strategy	challenge_state	challenge_finding	challenge_hypothesis	challenge_knowledge	challenge_activity
Reject a given or proposed requirement and propose an alternative one instead.	Signal agreement with a given or proposed requirement.	Reject a given proposal regarding the strategy or work plan and make an alternative proposal instead.	Signal agreement with a given proposal regarding the strategy or work plan.	Reject a statement regarding the degree to which the current strategy or work plan has been worked through and make an alternative statement.	Reject the content of a verbalized insight or interpretation and suggest an alternative one.	Reject a given hypothesis or conjecture and formulate an alternative one.	Declare transferred knowledge as fully, partially, or potentially wrong by opposing it with one's own knowledge.	Reject all or part of the current HCI or HEI activity and suggest an alternative activity.
	propose_requirement	decide_strategy	propose_strategy	propose_todo	disagree_finding	disagree_hypothesis	disagree_knowledge	disagree_activity
	Propose one or several alternative program characteristics that should be considered to be a requirement.	Select one from among several alternative proposed strategies or work plans.	Propose one or several alternative strategies or work plans.	Suggest that a certain work item will need to be taken care of later in the process.	Declare transferred finding as fully, partially, or potentially wrong without explaining why.	Reject a given hypothesis or conjecture.	Declare transferred knowledge as fully, partially, or potentially wrong without explaining why.	Reject all or part of the current HCI or HEI activity.
mumble_sth	say_off_topic	disagree_strategy		agree_todo	amend_finding	amend_hypothesis		amend_activity
Make an incomprehensible utterance (highly fragmentary or acoustically unclear).	Make an utterance that has nothing to do with solving the programming task.	Reject a given proposal regarding the strategy or work plan without making an alternative proposal.		Signal agreement with a statement saying that a certain work item will need to be taken care of later in the process.	Extend a verbalized insight or interpretation without rejecting it.	Extend a given hypothesis or conjecture without rejecting it.		Propose an extension to the current HCI or HEI activity.
miscellaneous								

facade concepts

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

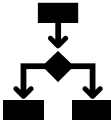
↔ Togetherness als fundamentales Konzept in der PP.



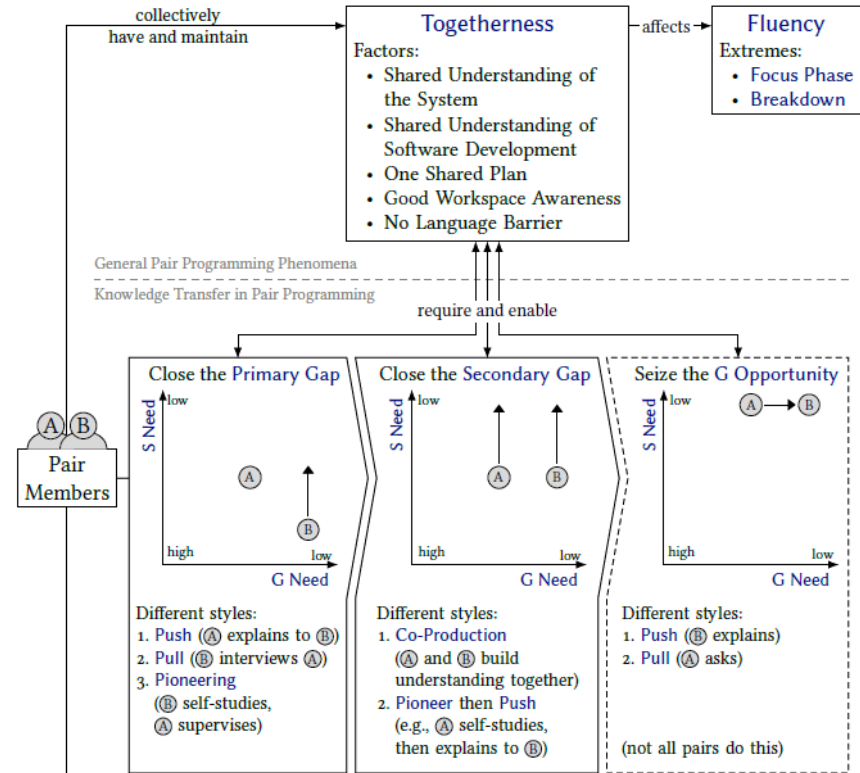
Hohe Togetherness führt zu Focus Phases, niedrige zu Breakdowns



Unterscheidung von S(pecific)- & G(eneral)-Knowledge



Wissenstransfer und -erwerb in unterschiedlichen Modi



Grafik aus [2] *364 seitige Dissertation: Ergebnisse in Auszügen und verkürzt

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

←→ Togetherness als fundamentales Konzept in der PP.



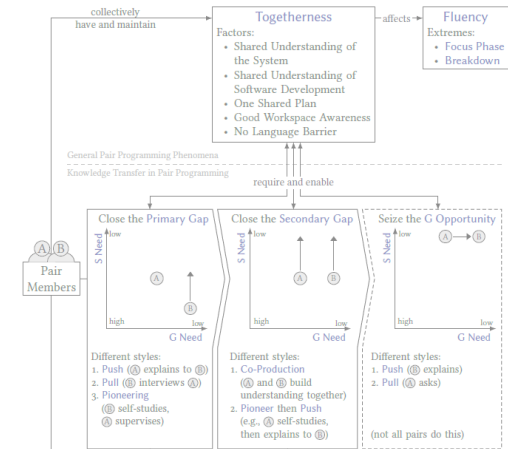
Hohe Togetherness führt zu Focus Phases, niedrige zu Breakdowns



Unterscheidung von S(pecific)- & G(eneral)-Knowledge



Wissenstransfer und -erwerb in unterschiedlichen Modi

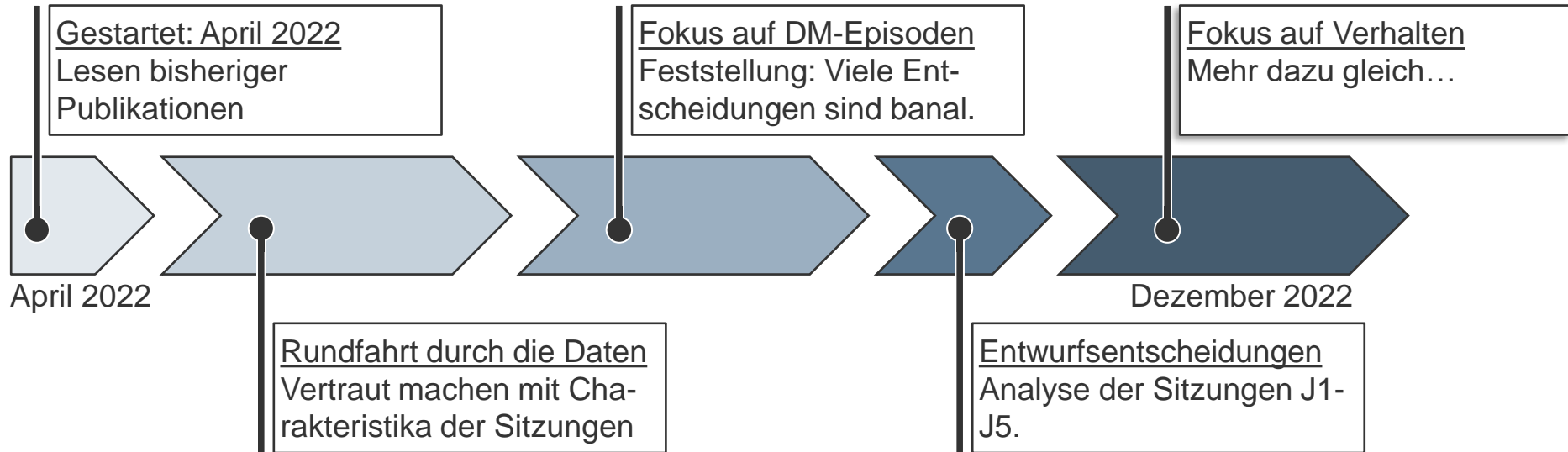


Grafik aus [2] *364 seitige Dissertation: Ergebnisse in Auszügen und verkürzt

Wie habe ich gearbeitet?

Forschungsprozess der letzten Monate & Beispielhafte Datenanalyse

Entscheidungsfindung in der PP (that's me)



Codierungsbeispiele



Beispiel: CA1



Observer: C1
Hat vor der Sitzung an der Aufgabe gearbeitet.



Driver: C2
Seit Beginn Driver und bleibt Driver für ca. 90% der Sitzung.



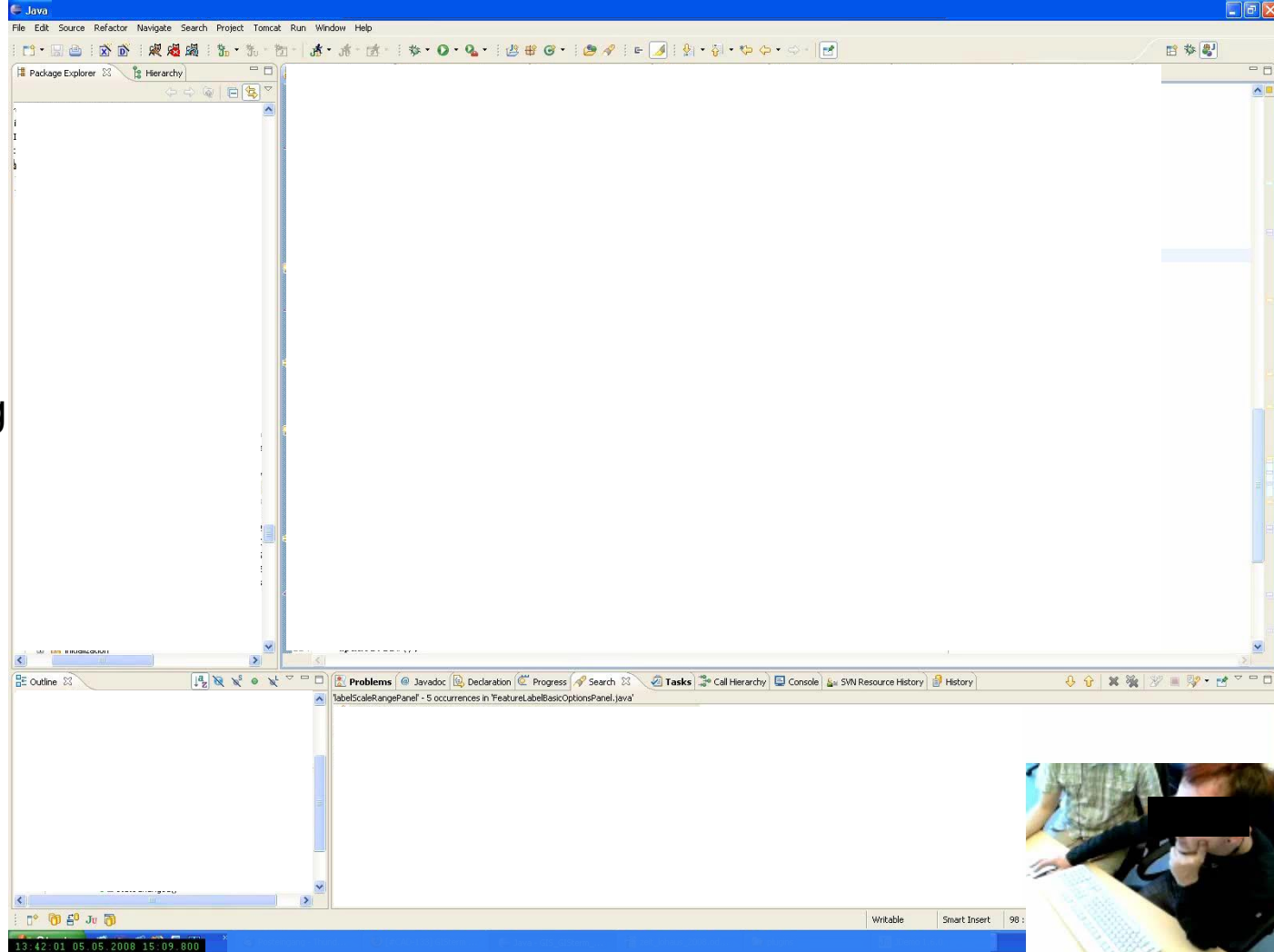
Neuartiges GUI-Element hinzufügen. Design und Verhalten ähnelt einer Komponente, ist aber nicht identisch.



Der Clip beginnt mit dem Vorschlag von C1 eine Zeile, die er vor der Sitzung geschrieben hat, einzukommentieren und sich die Effekte im GUI anzuschauen.

Beispiel: CA1

- 15 Minuten nach Beginn der Sitzung
- C1 (links) hatte bereits im Vorfeld an der Aufgabe gearbeitet



Beispiel: CA1 0:15:20 - 0:17:12

C1

C2

Beispiel: CA1 0:15:20 - 0:17:12

propose_step

C1



C2

„Du kannst ja mal einkommentieren und dann schauen wir uns in der Demo an, wie es aussieht. Dann kann ich dir auch gleich die Effekte zeigen, die da auftreten.“

Beispiel: CA1 0:15:20 - 0:17:12

propose_step

C1



propose_hypoth.

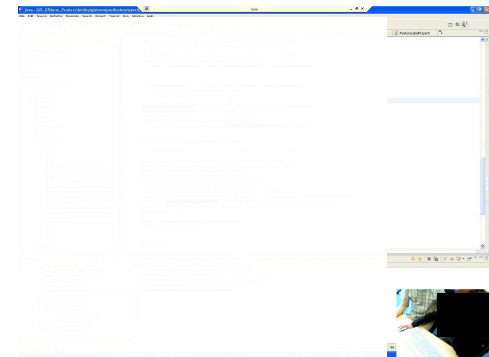
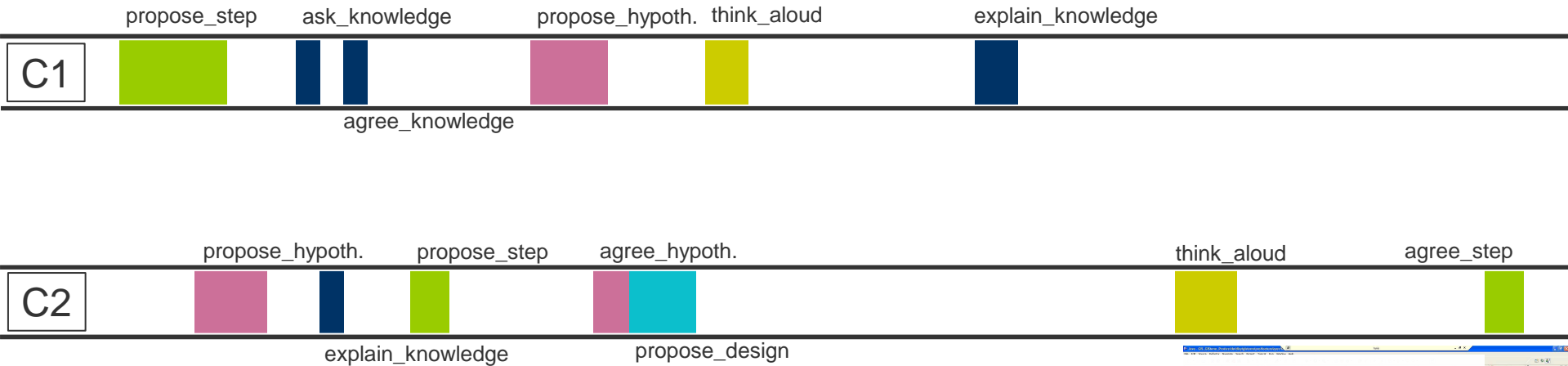
C2



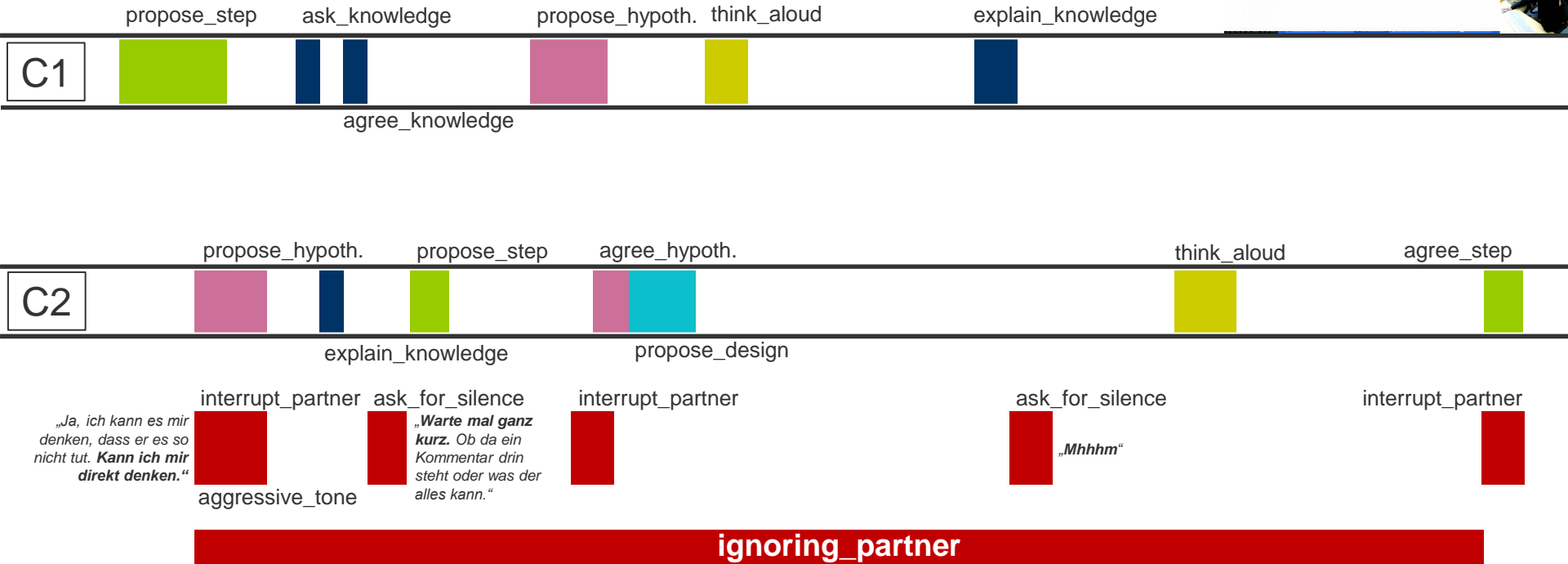
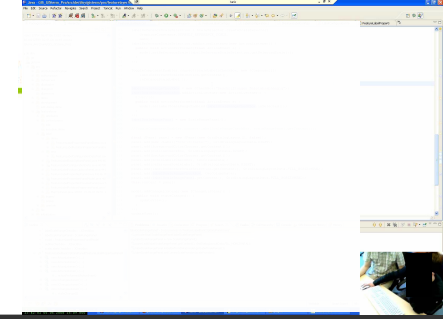
„Du kannst ja mal einkommentieren und dann schauen wir uns in der Demo an, wie es aussieht. Dann kann ich dir auch gleich die Effekte zeigen, die da auftreten.“

„Ja, ich kann es mir denken, dass er es so nicht tut. Kann ich mir direkt denken.“

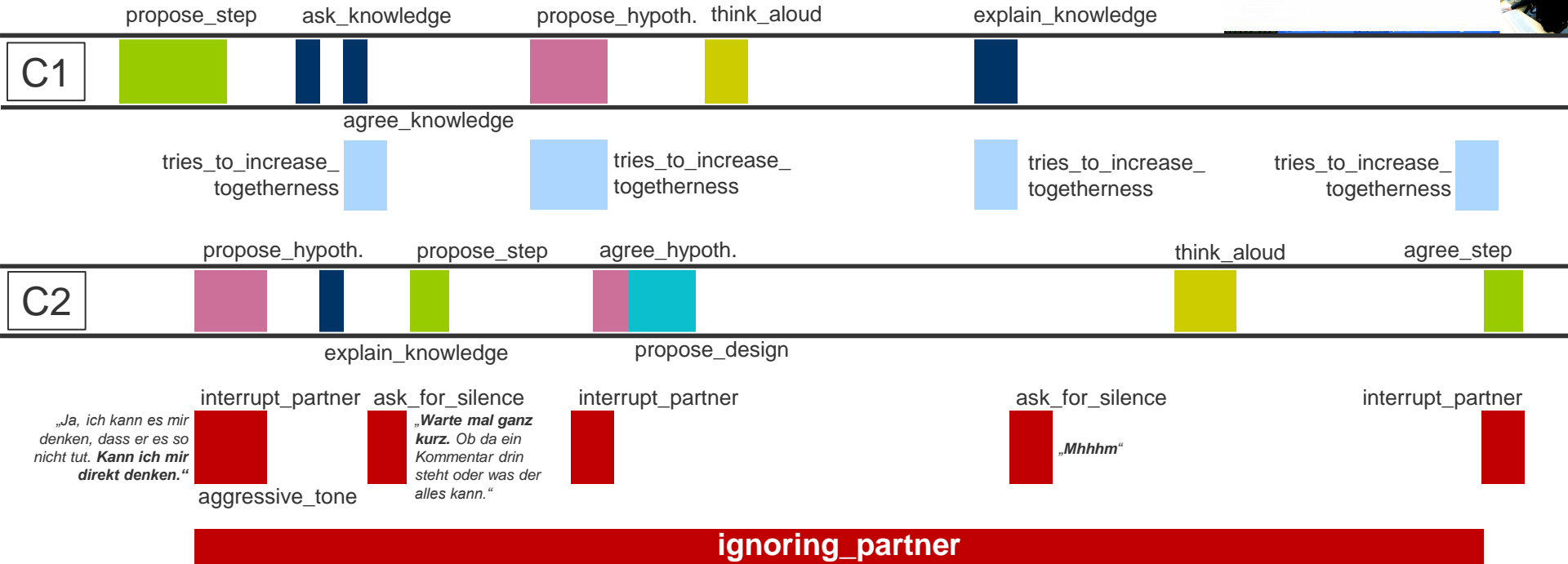
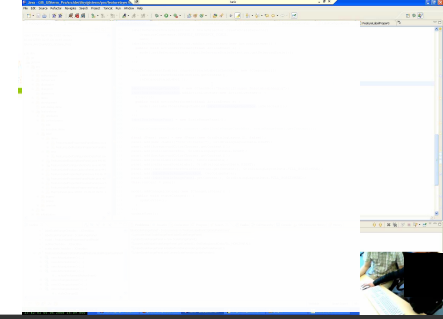
Beispiel: CA1 0:15:20 - 0:17:12



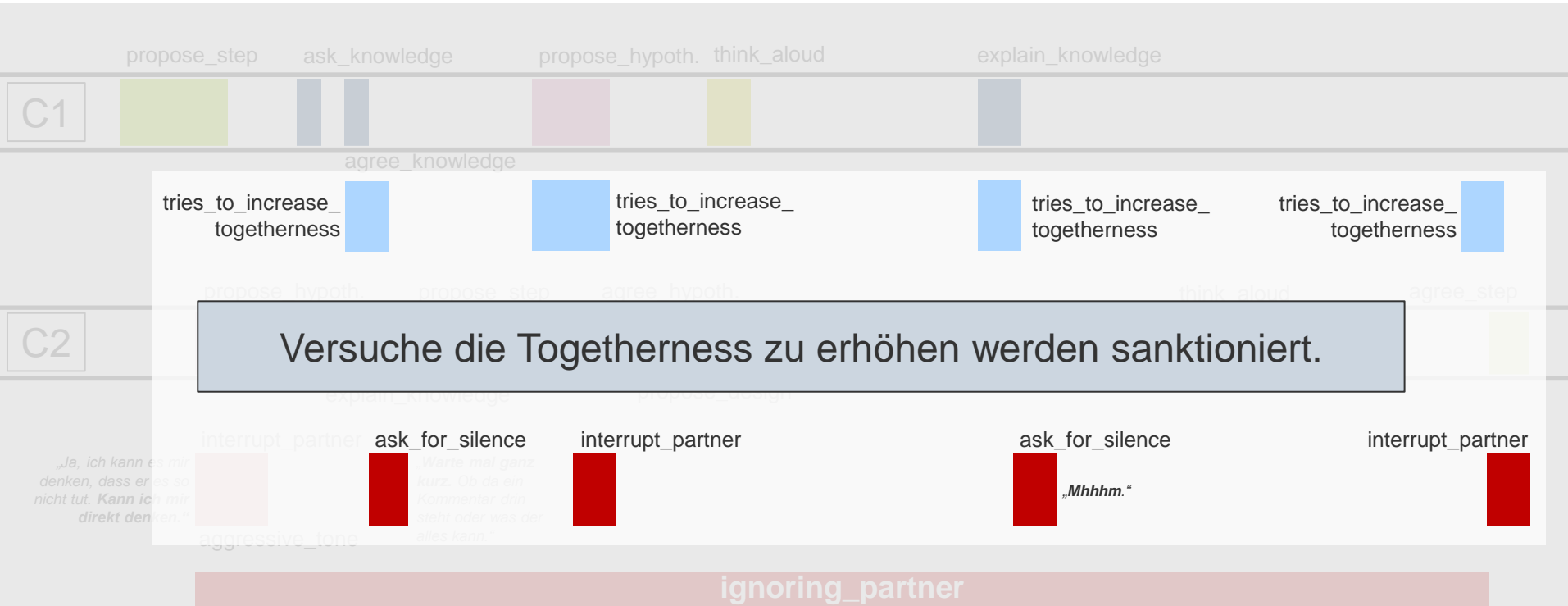
Beispiel: CA1 0:15:20 - 0:17:12



Beispiel: CA1 0:15:20 - 0:17:12



Beispiel: CA1 0:15:20 - 0:17:12



Beispiel CA1: Weitere Auffälligkeiten



C1 ist jeweils nur zum Commiten als Driver tätig
C2 reißt die Tastatur, sobald es komplexer wird, an sich.



Lediglich auf eine Entscheidung zum GUI-Design hat C1 einen Einfluss. Ansonsten keinen Faktor bei Entscheidungen.



Wissen wird aufgrund des Verhaltens von C2 und dem in die Passivität gedrängten C1 kaum vermittelt.

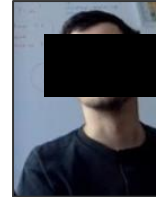


Vorteile der Paar-Programmierung gehen durch das autoritäre Verhalten von C2 verloren.

Beispiele aus Sitzung PA3



Driver: P3
Driver für die gesamte Sitzung



Observer: P1
Meist zurückgelehnt dasitzend

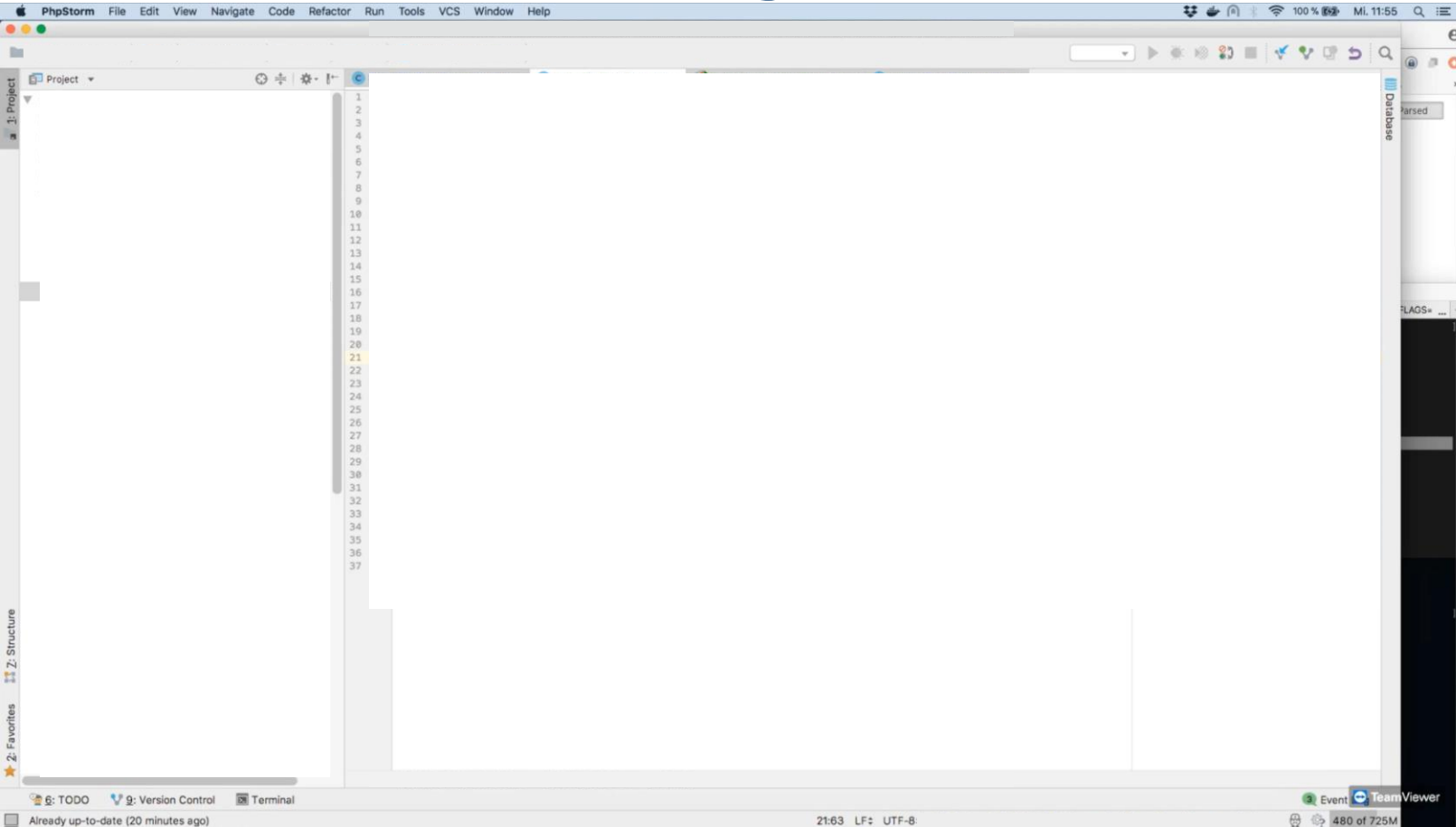


Entwicklung eines API-Endpoints zum Anzeigen von Daten im Frontend.
Die Daten sind als Prozentwerte gegeben und sollen im Frontend als Werte zwischen 0 und 1 angezeigt werden.



Zum aktuellen Zeitpunkt ist die Funktionalität bereits fertig und sie haben gerade die Einrückung zur besseren Lesbarkeit angepasst.
P1 macht einen Verbesserungsvorschlag...

Beispiele aus Sitzung PA3



Beispiel PA3: Weitere Auffälligkeiten



P1 neigt zu einem belehrenden Tonfall.



P1 sitzt die meiste Zeit zurückgelehnt und gibt P3 sehr genaue Anweisungen, was zu tun ist.

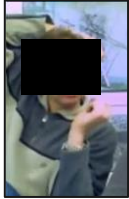


P3 ist an einigen Stellen (hier am deutlichsten) genervt von dem Verhalten von P3 und beschwert sich nach der Sitzung darüber.

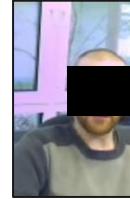


Ungefragte Belehrung von P1, führt zu Anspannung / Genervtheit von P3 und damit u.a. zum Treffen einer schlechten Entscheidung.

Beispiele aus Sitzung AA1



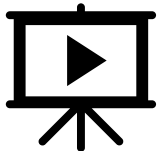
Observer: A1
Observer für gesamte Sitzung.



Driver: A2
Driver für gesamte Sitzung.

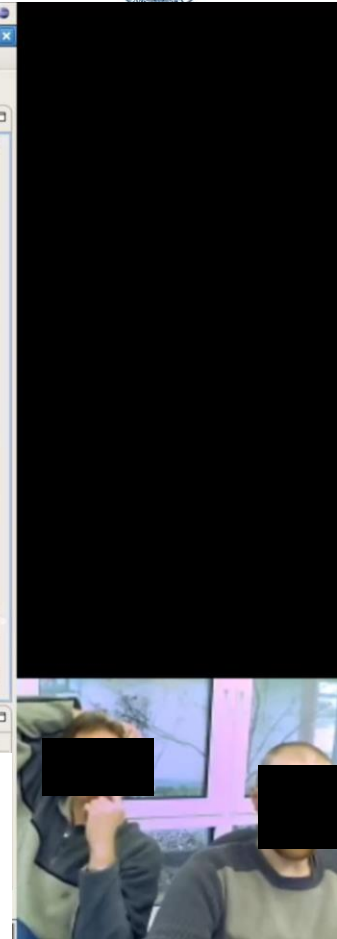
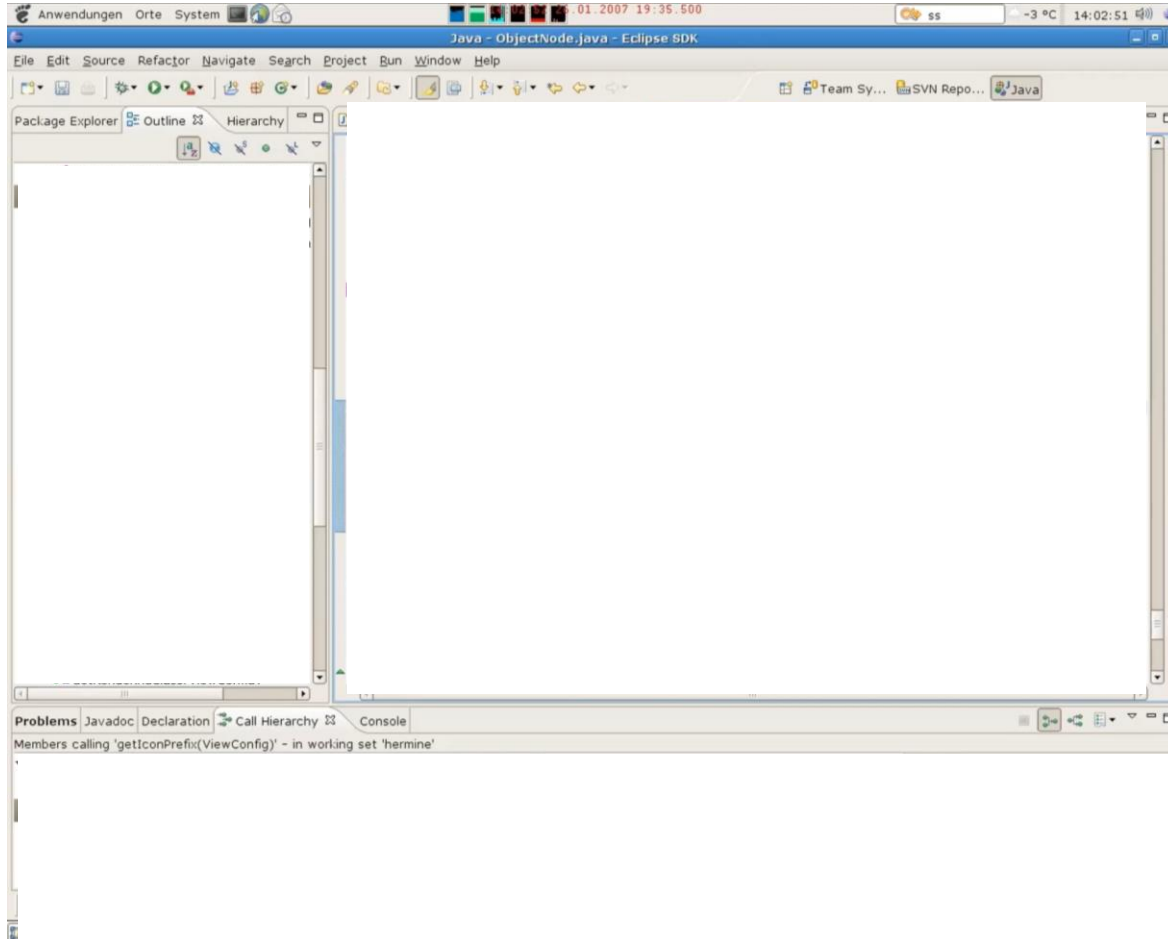


Beseitigen von Inkonsistenzen in 5 unterschiedlichen Listen im Frontend. Arbeiten an dem Anzeigen des richtigen Icons. Dafür wird der Name der Grafik u.a. in Abhängigkeit ob das Objekt aktiv/inaktiv ist zusammengebaut.



Haben die Methode an der sie gerade arbeiten komplett neu geschrieben. Und sich u.a. währenddessen darauf geeinigt, dass sie dafür ein CMMicroObject brauchen. A2 ändert das nun zu einem CMMiniObject.

Beispiele aus Sitzung AA1



Beispiele AA1: Weitere Auffälligkeiten



A1 neigt ähnlich wie P1 zu einem belehrenden Tonfall gepaart mit sehr aggressiven Interventionen.



A1 ist ähnlich wie P1 die ganze Zeit Observer und gibt spezifische Anweisungen.



Anders als P3, lässt sich A2 nicht aus der Ruhe bringen (bis auf 1x innerhalb der Sitzung als er A1 bittet ihn ausreden zu lassen).



Warum hat das Verhalten von A1 hier keinen schädlichen Einfluss?

Beispiele AA1: Weitere Auffälligkeiten



A1 neigt ähnlich wie P1 zu einem belehrenden Tonfall gepaart mit sehr aggressiven Interventionen.



A1 ist ähnlich wie P1 die ganze Zeit Observer und gibt spezifische Anweisungen.



Anders als P3, lässt sich A2 nicht aus der Ruhe bringen (bis auf 1x innerhalb der Sitzung als er A1 bittet ihn ausreden zu lassen).



Warum hat das Verhalten von A1 hier keinen schädlichen Einfluss?

Forschen mit der Grounded Theory Methodology:

Wie entstehen Konzepte & Hypothesen?

Forschen mit der Grounded Theory Methodology:



Entscheidungs-episoden beginnen mit **propose_***, gefolgt von **disagree_***; **challenge_*** ; bis zu einem **agree_***



Es gibt unterschiedliche Arten des propose, challenge, disagree, agree:



Mit oder ohne Begründung



Spezifisch oder Abstrakt



Vorschlag durch „Vormachen“



„Zustimmung“ durch keinen Widerspruch



Die Rollenverteilung (Driver/Observer) hat großen Einfluss auf den Entscheidungsprozess.

product-oriented concepts		process-oriented concepts		
general_design Create a general design regarding the structure and content of the program without specifying the structure of the program.	ask_design Ask for a concrete proposal regarding the structure and content of the program.	assess_step Extend a given proposal regarding the next task/step work step regarding the proposal.	ask_step Ask for a concrete proposal regarding the next task/step work step.	explain_completion Explain a statement regarding the degree of completion of the current task/step work step.
challenge_design Highlight a given proposal regarding the structure and content of the program and make an alternative proposal.	agree_design Agree with a proposal regarding the structure and content of the program.	challenge_step Highlight a given proposal regarding the next task/step work step and make an alternative proposal.	agree_step Agree with a proposal regarding the next task/step work step.	agree_completion Agree with a statement regarding the degree of completion of the current task/step work step.
decide_design Select one from among several proposals regarding the structure and content of the program.	propose_design Make one or several proposals regarding the structure and content of the program.	decide_step Select one from among several proposals regarding the next task/step work step.	propose_step Make one or several proposals regarding the next task/step work step.	challenge_completion Highlight a statement regarding the degree of completion of the current task/step work step and make an alternative proposal.
diagnose_design Reason a given proposal regarding the structure and content of the program as an alternative proposal.		diagnose_step Reason a given proposal regarding the next task/step work step without making an alternative proposal.		explain_state Explain a statement regarding the degree to which the current state of the current task/step work step is reached.
restructure_requirements Extend the scope of a given plan/step without functional or non-functional requirement of the plan/step.		assess_strategy Extend a proposal regarding the strategy to work on a plan/step without reworking.	ask_strategy Ask for a concrete proposal regarding the strategy to work on a plan/step.	agree_state Agree with a statement regarding the degree to which the current state of the current task/step work step is reached.
challenge_requirements Highlight a given plan/step and make an alternative proposal.	agree_requirements Agree with a plan/step of a proposal of the program.	challenge_strategy Highlight a given proposal regarding the strategy to work on a plan/step and make an alternative proposal.	agree_strategy Agree with a proposal regarding the strategy to work on a plan/step.	challenge_state Highlight a statement regarding the degree to which the current state of the current task/step work step is reached and make an alternative proposal.
propose_plan Propose one or several alternative program characteristics that should be in a requirement.		decide_strategy Select one from among several alternative proposals regarding the strategy to work on a plan/step.	propose_strategy Propose one or several alternative proposals regarding the strategy to work on a plan/step.	propose_state Propose one or several alternative proposals regarding the degree to which the current state of the current task/step work step is reached.
evaluate_state Make an uncomprehensible proposal (frequently or occasionally) without.	say_off_topic Make an off-traffic statement that has nothing to do with solving the programming task.	disagree_strategy Highlight a given proposal regarding the strategy to work on a plan/step without making an alternative proposal.		agree_activity Agree with a statement regarding the degree to which the current state of the current task/step work step is reached and need to take one or several actions in the process.
universal concepts				
explain_plan_in_knowledge Explain a plan/step in terms of knowledge to not give plan/step a number in the plan/step.	agree_plan_in_knowledge Agree with a plan/step in terms of knowledge to not give plan/step a number in the plan/step.	challenge_plan_in_knowledge Highlight a plan/step in terms of knowledge to not give plan/step a number in the plan/step.	propose_plan_in_knowledge Propose a plan/step in terms of knowledge to not give plan/step a number in the plan/step.	ask_statement_of_knowledge Ask a statement regarding the degree to which the current state of the current task/step work step is reached.
explain_finding Explain a finding in terms of knowledge to not give finding a number in the finding.	propose_finding Propose a finding in terms of knowledge to not give finding a number in the finding.	challenge_finding Highlight a finding in terms of knowledge to not give finding a number in the finding.	agree_finding Agree with a finding in terms of knowledge to not give finding a number in the finding.	think_about_activity Think about a statement regarding the degree to which the current state of the current task/step work step is reached.
agree_finding Agree with a finding in terms of knowledge to not give finding a number in the finding.	propose_hypothesis Propose a hypothesis in terms of knowledge to not give hypothesis a number in the hypothesis.	challenge_hypothesis Highlight a hypothesis in terms of knowledge to not give hypothesis a number in the hypothesis.	agree_hypothesis Agree with a hypothesis in terms of knowledge to not give hypothesis a number in the hypothesis.	agree_activity Agree with a statement regarding the degree to which the current state of the current task/step work step is reached.
challenge_finding Highlight a finding in terms of knowledge to not give finding a number in the finding.	challenge_hypothesis Highlight a hypothesis in terms of knowledge to not give hypothesis a number in the hypothesis.	challenge_knowledge Highlight a knowledge in terms of knowledge to not give knowledge a number in the knowledge.	challenge_activity Highlight a statement regarding the degree to which the current state of the current task/step work step is reached.	disagree_activity Disagree with a statement regarding the degree to which the current state of the current task/step work step is reached.
disagree_finding Disagree with a finding in terms of knowledge to not give finding a number in the finding.	disagree_hypothesis Disagree with a hypothesis in terms of knowledge to not give hypothesis a number in the hypothesis.	disagree_knowledge Disagree with a knowledge in terms of knowledge to not give knowledge a number in the knowledge.	disagree_activity Disagree with a statement regarding the degree to which the current state of the current task/step work step is reached.	
assess_finding Extend a finding in terms of knowledge to not give finding a number in the finding.	assess_hypothesis Extend a hypothesis in terms of knowledge to not give hypothesis a number in the hypothesis.	assess_knowledge Extend a knowledge in terms of knowledge to not give knowledge a number in the knowledge.	assess_activity Extend a statement regarding the degree to which the current state of the current task/step work step is reached.	

Driver-Observer-Dynamik im Fokus



Der Driver hat mehr Möglichkeiten Entscheidungen zu initiieren und hat automatisch das letzte Wort.



Der Observer kann intervenieren. Und ist gefährdet den Anschluss zu verlieren, wenn er dies nicht tut.



Gleichberechtigung ist für gute Entscheidungsprozesse fundamental (Behauptung!). Driver muss dies aktiv ermöglichen.



Betrachtung von CA1 unter diesen Gesichtspunkten, führte zum Fokus auf autoritäre Partner, die die Togetherness aktiv gefährden.

Autoritäres Verhalten in der PP

Verhalten von autoritären Partnern:

 Unterbrechen (und mit anderem Thema fortfahren)

 Ignorieren

 Belehren

 Aggressiver Tonfall

CA1/PA3/AA1

CA1

CA1/PA3/AA1

CA1/AA1

Auswirkungen auf die Sitzungen:

 Observer wird in die Passivität gedrängt / geringe Togetherness

 Driver reagiert genervt + schlechte Entscheidung

CA1

PA3

Weitere Ideen und Überlegungen


Fallstudie zu JA1 – JA9:

- 9 aufeinander aufbauende Sitzungen ohne Unterbrechung.
- Von Design bis zum finalen Produkt
- Entscheidungen können bewertet werden

Entscheidungsausgänge:

- Ursprünglicher Vorschlag
- Gegenvorschlag
- Gemeinsamer Kompromiss
- Entscheidung vertagt

Entscheidungstreiber

- Wissenstransfer
 - Argumentation
 - Third-Party
- 

Einfluss des Senior-Junior-Verhältnis auf die Entscheidungsfindung:

- Wann setzt sich wer durch?
- 

Fazit



Schädliche Auswirkung von autoritären Verhalten auf die Paar-Programmierung.



Erste Erkenntnisse und Belege



Aber: Nicht eindeutige Auswirkungen → Es fehlen Daten!



Einfluss der Driver-Observer-Dynamik auf die Entscheidungsfindung.



Erste Erkenntnisse und Belege



Aber: Wie geht es damit weiter?

Fazit



Schädliche Auswirkung von autoritären Verhalten auf die Paar-Programmierung.



Erste Erkenntnisse und Belege



Aber: Nicht eindeutige Auswirkungen → Es fehlen Daten!



Einfluss der Driver-Observer-Dynamik auf die Entscheidungsfindung.



Erste Erkenntnisse und Belege



Aber: Wie geht es damit weiter?

Vielen Dank!

Fragen / Anregungen / Kommentare?

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 \(650×826\) \(scitation.org\)](https://scitation.org/1.1570789.figures.f1.jpeg)