Knowledge transfer in pair programming: an in-depth analysis

How to cite:

For guidance on citations see FAQs

© 2014 Elsevier Ltd
Version: Accepted Manuscript
Link(s) to article on publisher’s website:
http://dx.doi.org/doi:10.1016/j.ijhcs.2014.09.001

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online’s data policy on reuse of materials please consult the policies page.

oro.open.ac.uk
Title: Knowledge Transfer in Pair Programming: An In-depth Analysis

Article Type: Original Article

Keywords: pair programming; knowledge transfer; interaction analysis; cognitive apprenticeship; qualitative analysis

Corresponding Author: Prof. Helen Sharp,
Corresponding Author's Institution: The Open University

First Author: Laura Plonka

Order of Authors: Laura Plonka; Helen Sharp; Janet Van der Linden; Yvonne Dittrich

Abstract: Whilst knowledge transfer is one of the most widely-claimed benefits of pair programming, little is known about how knowledge transfer is achieved in this setting. This is particularly pertinent for novice–expert constellations, but knowledge transfer takes place to some degree in all constellations. We ask "what does it take to be a good "expert" and how can a "novice" best learn from a more experienced developer?". An in-depth investigation of video and audio excerpts of professional pair programming sessions using Interaction Analysis reveals six teaching strategies ranging from giving direct instructions to subtle hints, and challenges and benefits for both partners. These strategies are instantiations of some but not all teaching methods promoted in cognitive apprenticeship; novice articulation, reflection and exploration are not seen in the data. The context of pair programming influences the strategies, challenges and benefits, in particular the roles of driver and navigator and agile prioritisation which considers business value rather than educational progression. Utilising these strategies more widely and recognizing the challenges and benefits for both partners will help developers to maximise the benefits from pairing sessions.
25 July 2014

Dear Editor-in-Chief

I wish to submit this revision of the paper Knowledge Transfer in Pair Programming: An In-depth Analysis by Laura Plonka, Helen Sharp, Janet Van der Linden and Yvonne Dittrich.

I trust everything is in order and look forward to hearing from you.

Yours sincerely

Professor Helen Sharp

helen.sharp@open.ac.uk
Comments:

Reviewer #1: This review is provided under a Creative Commons Attribution license (CC-BY).
This means the author must be mentioned when the text is distributed.
The author of this review is Lutz Prechelt.

Author Response: I understand from the IJHCS editorial office that we do not need to respond to this, e.g. by including the reviewer in acknowledgements

Judgment:
This work asks three research questions the answers of which have practical applicability and manages to provide answers for all of them that are both understandable and tangible.

The research uses a qualitative method based on a rather small amount of data (less than 30 minutes of live action) and so the completeness of the results is unlikely and the generalizability is unclear. However, the approach and discussion are very convincing and the validity and credibility of the results are very high.

The article is nicely readable overall. The writeup has many small problems (see the detailed comments), but no large ones. I nevertheless ask the authors to consider my suggestion ##E##.

My only two criticisms of substantial weight concern one conclusion that I find unwarrantedly negative (see detailed comment ##A##) and a terminological problem: Although the whole article revolves around expert-novice relationships, these two concepts are defined only in a highly fuzzy manner (see detailed comment ##N##, also ##S##).

My overall perception is very positive and I am sure that this work ought to be published eventually. However, as the number of issues is large and at least the terminological problem is important, I still suggest to ask the authors for a major revision first.

Detailed comments (formulated chronologically while reading):

<table>
<thead>
<tr>
<th>Comment</th>
<th>Author Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract: &quot;reveals six teaching strategies ranging from giving direct</td>
<td>Punctuation changed to clarify meaning</td>
</tr>
<tr>
<td>instructions to subtle hints, and challenges and benefits for both</td>
<td></td>
</tr>
<tr>
<td>partners&quot; The last part of this sentence does not quite fit in.</td>
<td></td>
</tr>
<tr>
<td>1 Introduction:</td>
<td>References added to introduction</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>- &quot;Several benefits of PP have been claimed including improved understandability and maintainability of code and design [34, 36], decreased defect rates [24, 18, 23, 10, 25] and knowledge transfer.&quot;</td>
<td></td>
</tr>
<tr>
<td>Oops! The abstract called knowledge transfer &quot;one of the most widely-claimed benefits of pair programming&quot; and now there is not a single literature reference for it?</td>
<td></td>
</tr>
<tr>
<td>- &quot;Although knowledge transfer is widely reported as a benefit of PP,&quot;</td>
<td>References added above so not repeated at this point.</td>
</tr>
<tr>
<td>Again, no literature reference?</td>
<td></td>
</tr>
<tr>
<td>- &quot;RQ3: What challenges are faced when pairing developers with different skill levels?&quot;</td>
<td>Rephrased as “What challenges do developers with different knowledge levels face when pairing together?”</td>
</tr>
<tr>
<td>Are faced? Strange wording. What perspective is this going to be? I guess that of the developers themselves? Then please phrase it like that.</td>
<td></td>
</tr>
<tr>
<td>2 Knowledge Transfer in Pair Programming:</td>
<td></td>
</tr>
<tr>
<td>- &quot;They found that PP reduced the mentoring needed per day from 37% to 26%&quot;</td>
<td>Modified to “They found that PP reduced the mentoring needed per day from 37% of a developer's time to 26% of their time”</td>
</tr>
<tr>
<td>Percent of what? Time? And does that include the time used for PP?</td>
<td></td>
</tr>
<tr>
<td>- &quot;some developers in an industrial team had issues when they worked with a partner with similar expertise&quot;</td>
<td>Modified to “Jensen [18] found that pairing developers with similar expertise was counter-productive,&quot;. This is the way it is described in the reference.</td>
</tr>
<tr>
<td>The term &quot;issues&quot; is rather too vague here. Did not like it? Quarreled? Felt inefficient? Were inefficient?</td>
<td></td>
</tr>
<tr>
<td>3 Research design:</td>
<td></td>
</tr>
<tr>
<td>- &quot;full-resolution screen shot&quot;</td>
<td>Changed to: “and a full-resolution recording of the screen, showing the</td>
</tr>
</tbody>
</table>
video, as most people will consider a screen shot to be a still image and may get confused.

| - Table 1: In line 1, the minimum PP experience is larger than the minimum programming experience. |
| - 3.1: Did those teams use PP always, often, or only occasionally? |
| - 3.2: "reviews observations and hypotheses that emerge through the group viewing sessions" |

| - Table 1: In line 1, the minimum PP experience is larger than the minimum programming experience. |
| - 3.1: Did those teams use PP always, often, or only occasionally? |
| - 3.2: "reviews observations and hypotheses that emerge through the group viewing sessions" |

| code and capturing the developers' computer activities." |

| This is a typo. Changed to 0-20. |

| Sentences added “There was no set pattern for pairing. Developers decided themselves when it was appropriate to pair, and with whom.” |

| Sentence deleted “During this procedure the researcher reduces the data to be analysed and reviews observations and hypotheses that emerge through the group viewing sessions.” |

| Modified to “We identified five suitable exemplars, each between 4 and 6 minutes in length.” |

| This has been changed to “different levels of knowledge”. Throughout, we now refer only to knowledge and not skills, unless we are referring to another reference in which ‘skill’ is used. |

| - Table 1: In line 1, the minimum PP experience is larger than the minimum programming experience. |
| - 3.1: Did those teams use PP always, often, or only occasionally? |
| - 3.2: "reviews observations and hypotheses that emerge through the group viewing sessions" |

| code and capturing the developers' computer activities." |

| This is a typo. Changed to 0-20. |

| Sentences added “There was no set pattern for pairing. Developers decided themselves when it was appropriate to pair, and with whom.” |

| Sentence deleted “During this procedure the researcher reduces the data to be analysed and reviews observations and hypotheses that emerge through the group viewing sessions.” |

| Modified to “We identified five suitable exemplars, each between 4 and 6 minutes in length.” |

| This has been changed to “different levels of knowledge”. Throughout, we now refer only to knowledge and not skills, unless we are referring to another reference in which ‘skill’ is used. |

| - Table 1: In line 1, the minimum PP experience is larger than the minimum programming experience. |
| - 3.1: Did those teams use PP always, often, or only occasionally? |
| - 3.2: "reviews observations and hypotheses that emerge through the group viewing sessions" |

| code and capturing the developers' computer activities." |

| This is a typo. Changed to 0-20. |

| Sentences added “There was no set pattern for pairing. Developers decided themselves when it was appropriate to pair, and with whom.” |

| Sentence deleted “During this procedure the researcher reduces the data to be analysed and reviews observations and hypotheses that emerge through the group viewing sessions.” |

| Modified to “We identified five suitable exemplars, each between 4 and 6 minutes in length.” |

| This has been changed to “different levels of knowledge”. Throughout, we now refer only to knowledge and not skills, unless we are referring to another reference in which ‘skill’ is used. |

| - Table 1: In line 1, the minimum PP experience is larger than the minimum programming experience. |
| - 3.1: Did those teams use PP always, often, or only occasionally? |
| - 3.2: "reviews observations and hypotheses that emerge through the group viewing sessions" |

| code and capturing the developers' computer activities." |

| This is a typo. Changed to 0-20. |

| Sentences added “There was no set pattern for pairing. Developers decided themselves when it was appropriate to pair, and with whom.” |

| Sentence deleted “During this procedure the researcher reduces the data to be analysed and reviews observations and hypotheses that emerge through the group viewing sessions.” |

| Modified to “We identified five suitable exemplars, each between 4 and 6 minutes in length.” |

| This has been changed to “different levels of knowledge”. Throughout, we now refer only to knowledge and not skills, unless we are referring to another reference in which ‘skill’ is used. |
- "We used the background questionnaires and the interviews with the developers to identify sessions in which a more knowledgeable developer (expert) worked with a less knowledgeable developer (novice)"
  Is the expert status always general (senior, vs. junior) or is it sometimes topic-specific?

  This is not generic but is always subject-specific. Earlier in that paragraph, we say "Given that developers never have the exact same knowledge, it can be assumed that knowledge transfer takes place in every PP session.", hence acknowledging this matter.
  We have included a more detailed description of "novice versus expert " in the exemplars used for this analysis, in section 3.2.1 under “Step 1”, as follows: “In the context of this study, the definition of an expert and a novice is based on the developers’ perceptions of their knowledge for a particular pair programming session. We chose excerpts from PP sessions for which developers explicitly stated that the aim of the session was to transfer knowledge from the more knowledgeable developer (expert) to the less knowledgeable developer (novice) for the topic covered in this session. In addition we only chose sessions in which both developers agreed who is the expert and who is the novice in this particular session.”

- Is there any difference between "excerpt" and "exemplar"?
  If so, please explain it.
  If not, please unify the terminology.

  Step 3 in the procedure was modified as follows “Firstly, the researcher extracted exemplar excerpts of the data (this process is referred to as “cannibalising” [19, p. 46]). An excerpt is regarded as an “exemplar” if it is a typical or representative example of the data being studied.”

- Step 2: "all video and audio data was in German"
  All_video_ was in German??

  “video” removed

- Step 3: Very good description. And the different focus of the non-German speakers is a beautiful argument!

  😊

- Step 4: "The findings from the interviews were then compared to the findings from the group viewing session."
  What happened if they disagreed?

  Modified to “The findings from the interviews were then used to contextualise the findings from the group viewing session”
4 Findings:

- 4.1.1: "Other subtle guidance can include physical hints such as pointing to something on the screen"
  I am not sure if the adjective "subtle" is appropriate here. (meta-remark: The above remark is about as subtle as the pointing, I guess. Does it feel subtle to you?) The same holds for some of the uses of "subtle" further down.

  The point being made here is that the *guidance* is subtle rather than that the *pointing* is subtle.
  Further down, one ‘subtle’ is removed and another is modified to ‘gentle’

- "In this excerpt, the expert places the cursor"
  This should say "In the _following_ excerpt".

  Modified to say “Example 2” not “this excerpt”

- Example 3: "Twice statement.close"
  Huh? Is "statement.close" a quote from the source code?
  Then please typeset it differently.
  (There are more such cases below, e.g. in Example 5.
   Example 9 uses quotes.)

  Typeface changed to consolas for code statements

- Example 4: Line 2 is missing its quotes.

  Modified

- "ALT N" should probably be spelled "Alt-N".

  Modified

- There appears to be no clear criterion for the position of the examples with respect to the text discussing them. For instance, example 5 appears far below its discussion, whereas example 4 appears before even the section heading under which it will be discussed.
  (Examples 9 and 12 appear not just outside their subsubsection, but even outside their subsection.)
  For the reader, seeing the example _just_ before the discussion provides the best readability in my opinion, at least in most cases.

  Tables moved to be closer to their first reference point
- The discussions of the examples are a bit tiring to read. Have you considered adding the discussions’ _local_ commentary (that which is referring to only one specific statement) directly into the example’s table and keeping the discussion in the text short and focused on the _global_ characterization of the example?
  (Yes, a bit of work. But the result will be so much nicer.)

  The commentary in the example tables describes what the participants are doing, while the descriptions in the text are our interpretation of their activity.
  The current presentation keeps the data and our interpretation separate. We believe that it would not be appropriate to put them both into the tables.

- Example 5: In the unnumbered lines, an annotation such as "[2sec]" indicates the duration of what appears before. In contrast, in the numbered lines I _guess_ it indicates the duration of a pause.
  (Also applies in other examples.) Please make sure your notation has an unambiguous meaning.

  We have added an explanation about the notation of pauses in section 4.1.2, and have made some changes to ensure consistency of use.

- 4.2: Example 7:
  The topic here is explanation (as opposed to verbalization). However, line 4 is explicitly called verbalization.
  Should we take all the rest to be explanation?
  And _one_ explanation (because there was only one question)?
  The discussion text in 4.2.1 is not explicit in this regard, but should be.

  The expert only says these few words, and then stops, which does not correspond to the verbalise strategy.
  This sentence has been modified to "Expert keeps typing for 5 sec without any comments and then briefly verbalises what he is typing", and the following one has been modified to "Expert stops verbalizing and keeps typing for the next 10 sec without saying anything"

- 4.3: "In line 11, the expert mentions that it is time to switch roles."
  The "Haha" uttered at that point deserves an explanation/interpretation.

  This piece of dialogue is not relevant to the discussion here and so we have removed the “Haha”.

- Example 11 appears to be identical to Example 5.
  Please prepare the reader for that or even use a cross reference to Example 5

  Sentence added “Example 11 is the same as Example 5, but it is repeated below to emphasise that it is a continuation of Example 10 in the PP session.”
instead of repeating the whole thing. (You are using this approach already below in 4.4.2.)

- Also, Example 11 is only _almost_ identical to Example 5. For example, 11 ends with "interfere" where 5 has "intervene".

Examples 5 and 11 are the same. The wording has been changed for both examples to use the word "intervene".

- "This example illustrates the novice's change of behaviour from a passive listener [...]
  I know what you mean here, but still: It is not helpful that you have called the "passive listener" behavior of the novice "active listener" above...
  "passive listener" has been changed to "passive participant"

This comment is not relevant to the discussion of this example and so it has been removed.

- 4.4: Example 13:
  "The transcription here does not follow immediately after the transcription in Example 9." This statement should not be about "transcription". And it could tell us, specifically, how many seconds or minutes later in the session the scene occurred.

This has been clarified by adding "immediately" and replacing "eventually" with "then": "The novice has a different problem solving approach and tries to communicate that idea to the expert but is not immediately successful. The expert reacts to the novice's comments but does not really take the novice's suggestions into account. However, then the expert realises that the novice had the right idea"

- 4.4.2: "This indicates that explanations that are not related to the current activity lead to an additional cognitive effort."
  This is a hasty conclusion: First, you

We changed the sentence to phrase this more carefully by saying that "This indicates that explanations that are not related to the current activity can lead to an additional cognitive effort."
<table>
<thead>
<tr>
<th>have no clear notion of relatedness. Second, it may simply be that explaining the use of the debugger is the much simpler task, no matter what context.</th>
</tr>
</thead>
</table>
| The sentence “These quotes from two of the interviews illustrate the developers’ perspectives:” has been added

<table>
<thead>
<tr>
<th>- 4.4.3: Please mention that the second quote is from a different pair, not the novice of the same pair. (For some reason that was my initial assumption.)</th>
</tr>
</thead>
</table>
| 5 Discussion:
| - Please repeat the identifiers RQ1 etc. in the headings for clarity. |
| RQ numbers added. Heading for section 5.3 modified to reflect RQ3.

<table>
<thead>
<tr>
<th>- &quot;These strategies are particular instantiations of the teaching methods suggested in cognitive apprenticeship&quot; They are specializations, not instantiations, right?</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1: Cognitive apprenticeship is now taking a prominent role. Please shortly explain why you are using it (rather than something else or nothing at all).</td>
</tr>
</tbody>
</table>
| Further explanation of cognitive apprenticeship and its applicability to software development has been added at the beginning of section 5.1

<table>
<thead>
<tr>
<th>- &quot;Strategies (5) and (6) are examples of modeling&quot; I suggest to treat the names of the strategies as identifiers (with capitalization) and refer to them by name to make reading easier. Use mnemonic abbreviations if that gets too cumbersome. If you don't like this, we need at least a line-itemized enumeration of the strategies that is prominent and easy to find.</th>
</tr>
</thead>
</table>
| Capitalisation applied

| #AAA#
| - Articulation: I find your interpretation overly pessimistic. That expressing one's thoughts "was

| We have changed this sentence to “In this case, it seemed that articulation was used...” |
used as a method to get reassurance" does not rule out that it is also proper articulation and has all the usefulness expected of articulation. The point of articulation is that it is done, not that the teacher asks for it. Please go back to your data, consider this view, and possibly modify your conclusion.

We also note that the novice’s body language indicated that she sought reassurance, but as we have not analysed body language in any detail this is not included in the paper.

- Exploration: You may not have seen any exploration in your data, but I wouldn’t say that choosing follow-up tasks that foster learning is "not decided during PP sessions". Please go back to your data, consider this view, and possibly modify your conclusion.

Modified to “and identifying such tasks would not normally be decided during PP sessions”

- You say that in the context of PP "tasks are chosen according to agile prioritisation which considers business value, not educational progression". I find this statement overly general. (For instance consider Brooks' Law situations such as those in [41] is may make sense to behave otherwise, in particular if PP is not used throughout.) Please insert a "typically" at least.

“typically” inserted. Note that this review comment may have some words missing, but this is how it appears in our communication from IJHCS

- 5.3: "We did not observe challenges for the novice in this study," Is this also due to your selection of sessions, which is guess was more interested in ones that ran smoothly? (And then perhaps add a bullet for this in 3.2.1’s step 1)

Changed to “We do not report on challenges for the novice in this paper, but novice challenges such as social pressure are described in [29].”

6 Limitations:

- I agree with almost everything of what you say, but overall the section feels unsatisfactory. This is because the limitations you discuss lack clear categorization, which in turn happens because you often only discuss an issue, The limitations section has been modified to include statements that this limitation “potentially affects generalizability/completeness” accordingly. In addition, the sentence below has been added to the limitations
but not its likely effects for the validity of your results. As far as I can see, the limitations of your study concern three types of threat: to validity, to completeness, and to generalizability: 6.1 concerns completeness, 6.2 concerns generalizability, 6.3 concerns completeness and generalizability. None of them concerns validity -- and you ought to say so. Loudly.

7 Conclusions and future work:

- Some of my above remarks apply here again.

   Capitalisation of teaching strategies, insertion of 'typically' regarding agile prioritization

   - "However, novice articulation, where a novice verbalizes their own thought process, was not encouraged" This _is_ probably a result of the small size of your data sample, don't you think so? (Or do you intend to claim no expert has ever done this?)

   Added "Not only was novice Articulation not encouraged in any of the exemplars we analysed in detail, we can also claim, due to step 1 in our procedure, that it also did not feature in any of the sessions we recorded."

References:

- At least the following entries in the references list need checking and potentially correction: 3, 5, 6, 10, 21, 23, 24, 27, 38, 41 (many have _only_ capizalization and hyphenation issues, but some have them mixed with other problems)

   References have been checked and modified.

Global:

---

**Note**

None of these limitations affect the validity of the findings

Thank you for pointing this out

---

The definition of novice and expert for this study was added to section 3.2.1 as described above. The article does recognize that all PP sessions will involve some level of knowledge transfer and that we chose ‘extreme’s in order to
As all your subjects are software professionals, case (b) would be a much more appropriate reason for calling someone a novice (w.r.t. to the one topic only), but most of the text sounded a lot more like case (a).

I have two requests:
- Please explain in Section 1 or 2 whether you have (a) in mind or rather (b).
- If it is (a), please change the term "novice" into the more appropriate term "junior" (Latin for "younger") throughout.

- The article could use a round of proofreading.
  There are a number of issues for instance with
  - singular vs. plural,
  - third-person "s",
  - adverb "ly",
  - "err" vs. "er",
  - missing quotes,
  - "seem" vs. "appear",
  - "another" vs. "a different",
  - hyphenation issues,
  - missing words,
  and the like.

- Please indicate the identity of the session from which each example is taken.

We have not added this information. The excerpts chosen were exemplars of the data set, as explained above. Adding which session each example came from would emphasise an aspect of the analysis that is not relevant, and indeed would be distracting from the main findings.

<table>
<thead>
<tr>
<th>Knowledge wrt a single, specific topic. As all your subjects are software professionals, case (b) would be a much more appropriate reason for calling someone a novice (w.r.t. to the one topic only), but most of the text sounded a lot more like case (a).</th>
<th>be able to identify clearly the strategies being used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>I have two requests: - Please explain in Section 1 or 2 whether you have (a) in mind or rather (b). - If it is (a), please change the term &quot;novice&quot; into the more appropriate term &quot;junior&quot; (Latin for &quot;younger&quot;) throughout.</td>
<td></td>
</tr>
<tr>
<td>- The article could use a round of proofreading. There are a number of issues for instance with - singular vs. plural, - third-person &quot;s&quot;, - adverb &quot;ly&quot;, - &quot;err&quot; vs. &quot;er&quot;, - missing quotes, - &quot;seem&quot; vs. &quot;appear&quot;, - &quot;another&quot; vs. &quot;a different&quot;, - hyphenation issues, - missing words, and the like.</td>
<td>Proofread and changes made.</td>
</tr>
<tr>
<td>- Please indicate the identity of the session from which each example is taken.</td>
<td></td>
</tr>
</tbody>
</table>
Reviewer #2: Review of "Knowledge Transfer in Pair Programming: An in-depth Analysis"

This article describes an interaction analysis of pair programming sessions, focusing on knowledge transfer between experts and novices. The research questions are 1) What teaching strategies do developers use in pair programming, 2) In which ways do the roles of driver and navigator influence knowledge transfer in pair programming, and 3) What challenges are faced when pairing developers with different skill levels.

I find knowledge transfer in pair programming a very interesting topic, pair programming has been one of the agile practices that has received most attention, but most studies focus on other aspects than knowledge transfer. Thus, a study on knowledge transfer is very valuable both to academia and to practice, and I see a clear contribution in this work. The main strength of this study is the micro-level analysis of the videorecorded pair programming sessions, that results in a very detailed analysis. The article is very well written, and I found the mixture of tables with data material and text interpreting the data to make the study very readable. Research questions are clear, and the research method sufficiently described. Limitations are discussed, and I see the limited number of sessions as the main limitation.

In total, I found this a very interesting article, but have some comments that I think need to be adressed before this article should be accepted for publication:

<table>
<thead>
<tr>
<th>Comment</th>
<th>Author Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Method: I am not familiar with interaction analysis, so if there are common criteria for evaluating studies within this domain, I think these should be used. Otherwise, I think the arguments for a &quot;sufficient&quot; sample could be strenthened by explicitly arguing for saturaturaton.</td>
<td>The notion of 'saturation' is not used in interaction analysis, but the method itself aims to ensure that the samples chosen are typical of all the data, e.g. developing a content log and checking back with the participants. We have modified the limitations section to be clearer about the type of limitations in the study.</td>
</tr>
<tr>
<td>2) Method: Would it be possible to state more about the difference between the &quot;novice&quot; and the &quot;expert&quot; in your study? Would an &quot;expert&quot; also be senior in age, or just be more skilled on the topic under discussion in the session?</td>
<td>We have added clarification to this issue, as described above.</td>
</tr>
<tr>
<td>3) Theory: Most of the theory part is now devoted to rather shallow prior studies on other aspects of pair</td>
<td>The response to this comment is broken into 3 parts: 1. Existing studies on pair programming</td>
</tr>
<tr>
<td>Programming. I would rather have referred to an overview article on pair programming, and examined more in detail the ones that actually address knowledge transfer. However, what I think could lift the theory part is to provide more details on relevant &quot;knowledge transfer&quot; from work situations similar to pair programming. For example von Krogh et al.'s book on knowledge creation focuses on tacit knowledge and the role of conversations. I would also have liked more information on the model by Collins et al. described in the theory part.</td>
<td>are generally rather shallow, which is why we have conducted more in-depth analyses. 2. We have considered the knowledge creation reference provided and believe that this would change the paper considerably and direct the work in a completely different way 3. We have added some more information about why Collins was used in our study in response to Reviewer 1, however including it in the initial theory section may imply that we started with this theory in mind but we did not.</td>
</tr>
<tr>
<td>I was surprised not to see references to very similar work by Lutz and Zieris, who also have done very detailed analysis of pair programming sessions. It would be very interesting to see a comparison between their findings and the ones in this study.</td>
<td>This reference has been added to the introduction section of the paper: &quot;Indeed, Salinger et al (2013) have identified PP roles other than driver and navigator, including one called task expert which brings in task expertise relevant to the session. This acknowledges the fact that all sessions include some level of knowledge transfer&quot;.</td>
</tr>
<tr>
<td>4) Discussion: What is really of interest for practitioners, is how effective is pair programming as a mechanism for knowledge transfer. I know that this is very difficult to measure, and that you probably do not have measures of the learning effect. But given the strategies for learning that you found in the study, would it be possible to compare this technique to other techniques in order to assess the likelihood that the technique will lead to genuine learning? Would for example a retrospective be likely to generate the same type of learning in shorter time?</td>
<td>This paper does not attempt to measure the learning effect and the data does not support this kind of work. This could potentially be pursued in further work, but it would require a different and quite extensive new study.</td>
</tr>
<tr>
<td>Other aspects that would be interesting is what kind of learning a technique can generate. Argysis and Schön distinguish between single-loop and double-loop</td>
<td>This could potentially be pursued in further work, but focusing on single or double-loop learning in a PP context would require a different and quite</td>
</tr>
</tbody>
</table>

| [90x708]programming. I would rather have referred to an overview article on pair programming, and examined more in detail the ones that actually address knowledge transfer. However, what I think could lift the theory part is to provide more details on relevant "knowledge transfer" from work situations similar to pair programming. For example von Krogh et al.'s book on knowledge creation focuses on tacit knowledge and the role of conversations. I would also have liked more information on the model by Collins et al. described in the theory part. | are generally rather shallow, which is why we have conducted more in-depth analyses. 2. We have considered the knowledge creation reference provided and believe that this would change the paper considerably and direct the work in a completely different way 3. We have added some more information about why Collins was used in our study in response to Reviewer 1, however including it in the initial theory section may imply that we started with this theory in mind but we did not. |
| I was surprised not to see references to very similar work by Lutz and Zieris, who also have done very detailed analysis of pair programming sessions. It would be very interesting to see a comparison between their findings and the ones in this study. | This reference has been added to the introduction section of the paper: "Indeed, Salinger et al (2013) have identified PP roles other than driver and navigator, including one called task expert which brings in task expertise relevant to the session. This acknowledges the fact that all sessions include some level of knowledge transfer". |
| 4) Discussion: What is really of interest for practitioners, is how effective is pair programming as a mechanism for knowledge transfer. I know that this is very difficult to measure, and that you probably do not have measures of the learning effect. But given the strategies for learning that you found in the study, would it be possible to compare this technique to other techniques in order to assess the likelihood that the technique will lead to genuine learning? Would for example a retrospective be likely to generate the same type of learning in shorter time? | This paper does not attempt to measure the learning effect and the data does not support this kind of work. This could potentially be pursued in further work, but it would require a different and quite extensive new study. |
| Other aspects that would be interesting is what kind of learning a technique can generate. Argysis and Schön distinguish between single-loop and double-loop | This could potentially be pursued in further work, but focusing on single or double-loop learning in a PP context would require a different and quite |
learning, would pair programming primarily function as a mechanism for single-loop learning? I think these topic could be interesting to raise in the discussion.

**extensive new analysis**

Figure 1: Great that you include this figure, but both elements should be enlarged to increase readability.

The image is included to illustrate the recording set up and what the data looked like for the researcher. A higher quality version can be submitted with the final manuscript, should the paper be accepted, but it is not necessary for the reader to be able to read the code on the screen.

**Section 3.2.1: Great if you could be more precise some places, like stating why you chose 5 exemplars of PP sessions, and how many illustrated RQ1 and RQ2?**

The wording was modified to be: “Five suitable exemplars were identified, each between 4 and 6 minutes in length, that together represented the set of typical situations in the data.” There was no particular mapping between the RQs and the exemplars; all of the exemplars illustrated all of the RQs

"ex-pert" -> "expert"

The article has been fully proofread

**Reviewer #3:** The paper is interesting and deals with a very popular set of problems that need to be investigated deeply. The study leverages on a wide set of previous investigations performed in the area and focuses on specific aspects that are investigated in depth.

There are some aspects of the paper that need improvement:

<table>
<thead>
<tr>
<th>Comment</th>
<th>Author Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>- it is not clear for how long the different couples have been recorded</td>
<td>Section 3.2.1 states that the data was 37 hours of video recording of 21 PP sessions. We have added “each PP session lasted between one and a half and three and a half hours.”</td>
</tr>
<tr>
<td>- it is not clear how the authors have identified the 6 groups</td>
<td>There were 5 exemplars and they were chosen according to the criteria in section 3.2.1. There were six strategies which emerged from the data, so they were not ‘identified’ by the authors.</td>
</tr>
<tr>
<td>- it is not clear if the analysis describes</td>
<td>The analysis identifies certain</td>
</tr>
<tr>
<td>just single behaviors or it groups together similar cases (the 6 strategies identified by the paper)</td>
<td>behaviours that we observe participants performing. The examples given are used to illustrate the behaviour that is typical of the whole data set.</td>
</tr>
<tr>
<td>- the paper could be improved providing a more detailed analysis of the single experiments and trying to group them in clusters to identify common behaviors keeping also the connection between the single experiments and the grouped ones</td>
<td>These were not single experiments but a set of recordings from a field study involving four companies, as set out in the paper.</td>
</tr>
</tbody>
</table>
Knowledge Transfer in Pair Programming: An In-depth Analysis

Laura Plonka, Helen Sharp, Janet van der Linden,
Centre for Research in Computing
The Open University Milton Keynes, UK
(Laura.Plonka; Helen.Sharp; Janet.VanderLinden@open.ac.uk)

Yvonne Dittrich
Software Development Group IT University of Copenhagen Copenhagen, Denmark (ydi@itu.dk)

Abstract

Whilst knowledge transfer is one of the most widely-claimed benefits of pair programming, little is known about how knowledge transfer is achieved in this setting. This is particularly pertinent for novice–expert constellations, but knowledge transfer takes place to some degree in all constellations. We ask “what does it take to be a good “expert” and how can a “novice” best learn from a more experienced developer?” An in-depth investigation of video and audio excerpts of professional pair programming sessions using Interaction Analysis reveals: six teaching strategies, ranging from “giving direct instructions” to “subtle hints”; and challenges and benefits for both partners. These strategies are instantiations of some but not all teaching methods promoted in cognitive apprenticeship; novice articulation, reflection and exploration are not seen in the data. The context of pair programming influences the strategies, challenges and benefits, in particular the roles of driver and navigator and agile prioritisation which considers business value rather than educational progression. Utilising these strategies more widely and recognizing the challenges and benefits for both partners will help developers to maximise the benefits from pairing sessions.
1 Introduction

“Two heads are better than one” is a common idiom referring to the advantages of collaborative work.

The value of collaboration is explicitly encouraged in software development through a practice known as pair programming. Pair programming (PP) is a software development technique where two developers work closely together to solve a development problem [41, 1].

Several benefits of PP have been claimed including improved understandability and maintainability of code and design [35, 37], decreased defect rates [24, 18, 23, 10, 25] and knowledge transfer [21, 20, 23, 32, 34, 35, 36, 38, 40]. This paper focuses on knowledge transfer in PP. Indeed, Salinger et al [31] have identified PP roles other than driver and navigator, including one called task expert which brings in task expertise relevant to the session. This acknowledges the fact that all sessions include some level of knowledge transfer.

Most software development teams are composed of developers with different knowledge levels of some kind, including different programming experience, different domain expertise and knowledge about different technologies. PP is one way to share their knowledge with other team members while also achieving meaningful work. In some cases, knowledge transfer is the explicit goal of a PP session [29]. This is common when a more experienced developer teaches a less experienced developer, for example, to bring new staff up to speed [3, 42]. However, given that developers never have identical knowledge, a certain degree of knowledge transfer would be expected within every PP constellation.

Pairing with someone who has a different knowledge level can be problematic [2, 7] and developers tend to interact differently in this situation in comparison to pairing with other developers with similar knowledge levels [9, 7]. For example, Plonka et al. [29] showed that less knowledgeable developers (novices) can disengage in PP sessions and can sometimes not follow their more knowledgeable partner (expert).
Although knowledge transfer is widely reported as a benefit of PP, there is currently not much insight into how developers approach this in practice nor how knowledge transfer can be improved. What does it take to be a good “expert” and how to learn best as a “novice”? What are the challenges? Here, we present an in-depth investigation of knowledge transfer in professional PP sessions to address the following research questions:

- RQ1: What teaching strategies do developers use in pair programming?
- RQ2: In which ways do the roles of driver and navigator influence knowledge transfer in pair programming?
- RQ3: What challenges do developers with different knowledge levels face when pairing together?

These three questions are addressed through a qualitative analysis (using Interaction Analysis [19]) of video recordings of professional developers working together on their day to day tasks. As a result, we identified a set of teaching strategies and behaviours that are related to the roles of driver and navigator and influence teaching and learning, together with associated challenges and benefits for both pairing partners. An increased awareness of working practices for knowledge transfer in PP will help developers to maximise the benefits from such sessions.

The remainder of the paper is organized as follows. Section 2 overviews existing research on knowledge transfer in PP. In section 3, we present the research methodology including data collection and analysis approach, followed by the findings of this study (section 4). In section 5, the findings are discussed with respect to existing literature and section 6 discusses the limitations of the study. The last section 7 presents conclusions and implications for developers.

## 2 Knowledge Transfer in Pair Programming

The positive effect of PP on knowledge transfer, no matter what may be the knowledge levels of the developers, is widely acknowledged across a range of studies in industry [21, 20], [38, 35, 23] and
Knowledge transfer is also one of the main perceived benefits according to two surveys: Schindler [33] surveyed developers and managers in 42 Austrian companies; and Begel and Nagappan [2] conducted a web-based survey of 487 Microsoft developers. Three industrial case studies [21, 35, 38] report more detail on developers’ perceptions. In [21], developers report that PP increased their knowledge of the code and in [35], developers report increased knowledge of the software system. Gaining knowledge about development tools, work practices, refactoring old code, new technologies and programming languages are all perceived benefits reported in [38].

Belshee [3] suggested very frequent changes of the pair constellation to promote fast knowledge transfer and to spread knowledge among different team members. Pandey et al. [23], suggests that this can reduce project risk because multiple developers are familiar with the code and there is less reliance on one individual. Increased flexibility also means that developers can pick up a variety of different tasks. For example, Hodgetts [17] reports on one team that had only one database expert, but too much work for one expert. When this caused a bottleneck, the team decided to use PP to spread the database knowledge among developers. They learned quickly through pairing with the database expert and were then able to do database tasks by themselves.

PP has also been studied in the context of training and mentoring, but not always with a positive effect. For example, in the context of developing firmware for processors, Greene [16] found that the training effect of PP was not as high as expected, which may be due to the very specialized and complex domain knowledge needed in that context. On the other hand, Williams et al. [42] investigated PP for mentoring and hence focused on pair constellations with different levels of expertise. They examined the relationship between PP and Brooks’ Law\(^1\) based on a survey and a case study. They found that PP reduced the mentoring needed per day from 37% of a developer’s time to 26% of their time, and that PP reduced the time for a developer to be independently productive from 27 to 12 days.

\(^1\) Brooks Law says that “adding manpower to a late software project makes it later” [6]
The developers’ view of combining different knowledge levels when pairing was investigated by Jensen [18] and Vanhanen et al. [38, 37]. Jensen [18] found that pairing developers with similar expertise was counter-productive, while Vanhanen and Lassanias [37] found two good partner combinations: when the pair consists of a senior and a junior developer; or partners have complementary knowledge.

When asked about the challenges of PP, developers surveyed by Begel and Nagappan [2] perceived working with someone with different skills as one of the main challenges. Williams and Kessler [41] also point out that pairing experts and novices can be problematic. Novices can slow down experts and some experts might not have a mentoring attitude.

One study by Cao and Xu [7] examined the interactions of pairs in more detail according to their expertise. They assigned students according to expertise and found that the expert asked for the novices’ opinions frequently at the beginning of the session but stopped asking after realising that they did not get valuable information.

Although there is some evidence that pairing developers with different knowledge is useful but challenging, there is currently a lack of understanding about what interactions take place to achieve knowledge transfer and what challenges developers face.

3 Research Design

It is known that people working jointly on a computer use a combination of gesture, language and screen object manipulation to construct an understanding of the problem (see [30] for example). In PP developers work closely together on one computer and all these aspects needs to be considered when analysing knowledge transfer between the developers. For this study, we chose a data gathering approach that captures rich data about the PP sessions and an analysis approach that allows for a detailed investigation of how human beings interact with each other, and with objects in their environment (both verbally and non-verbally) [19].
3.1 Data gathering

Different aspects of the PP session were captured by using a combination of data gathering methods:

- Audio and video recordings were used to record the developers’ interactions during their PP sessions: audio recordings of all verbal communication; a video of the programmers; and a full-resolution recording of the screen, showing the code and capturing the developers’ computer activities. These were fully synchronized into a single video file (see figure 1).

- Questionnaires were used to gather background information about the developers, the aim of the session, and their experience in programming and PP.

- Interviews were conducted with both developers one day after the session to capture the developers’ account of their session.

<table>
<thead>
<tr>
<th>Industry</th>
<th>Company size</th>
<th>Team size</th>
<th>Programming experience</th>
<th>PP experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographic information systems</td>
<td>30-50</td>
<td>8</td>
<td>0.9-20 years</td>
<td>0-20 years</td>
</tr>
<tr>
<td>Traffic, logistic and transport</td>
<td>&lt;500</td>
<td>2 teams, 5 developers each</td>
<td>0.4-13 years</td>
<td>0-3 years</td>
</tr>
<tr>
<td>Email marketing</td>
<td>50-100</td>
<td>8</td>
<td>1.3-10 years</td>
<td>0-5 years</td>
</tr>
<tr>
<td>Estate CRM Software</td>
<td>50-100</td>
<td>10</td>
<td>1.5-12 years</td>
<td>0-3.6 years</td>
</tr>
</tbody>
</table>

We recorded PP sessions from four different companies. All companies used agile approaches and all companies belonged to different industries. None of the companies provided developers with PP
training. Table 1 provides background about the companies and the developers. During our studies, the developers worked on their day-to-day tasks in their usual working environment. There was no set pattern for pairing. Developers decided themselves when it was appropriate to pair, and with whom. See figure 1 for an example of a recording set-up in one of the participating companies.

### 3.2 Analysis

To analyze the data we used Interaction Analysis [19] which focuses on social interactions (verbal and non-verbal) as they take place in their natural settings through analysing everyday interactions. Video data supports Interaction Analysis because it captures the minutiae of interactions. Moreover, video data allows sequences of interactions to be replayed which is crucial for re-examining and understanding what happens in the session.

![Figure 1: Left: Screenshot of a fully synchronized video showing Eclipse IDE and the developers. Right: Recording setup in one of the companies.](image)

Jordan and Henderson [19] originally published their description of Interaction Analysis in the context of learning sciences. In recent years Interaction Analysis has been used in software engineering research. For example, Børte et al. [5] successfully used Interaction Analysis to study software effort estimation by investigating different types of knowledge, reasoning and decision-making in group based estimation sessions, while Dittrich and Giuffrida [15] used the method to investigate the role of instant messaging in a global software development project.
Our analysis is based on the core procedures of Interaction Analysis but was tailored to the context of this study. The following summarises the six core procedures and how they were tailored, see [26] for details.

1. **Procedure: Ethnographic context** In order to provide context for the video data, Interaction Analysis suggests to capture the ethnographic context in which the recordings take place. For the data gathering of this study, the main researcher spent time at the organisations and in addition to the video data three other types of data (questionnaires, interviews, and field notes) were gathered during that stay.

2. **Procedure: Content logs** The intention of this procedure is to obtain an overview of the data through an initial viewing and annotation to create content logs. In this study, the video data was annotated through previous analyses [29, 27] and these annotations were used as content logs for this study.

3. **Procedure: Individual researcher’s work** In this study, this procedure was divided into two separate steps. Firstly, the researcher extracted exemplar excerpts of the data (this process is referred to as “cannibalising” [19, p. 46]). An excerpt is regarded as an “exemplar” if it is a typical or representative example of the data being studied. Secondly, after the group work procedure (described below) the researcher reviewed the group viewing notes and analysed the interview data.

4. **Procedure: Transcription** The video data to be analysed is transcribed. This procedure was followed in this study and is described below in the second step of our analysis.

5. **Procedure: Group work** Group work is fundamental to Interaction Analysis. Group members discuss observations and hypotheses, searching for “distinguishing practices” and “identifiable regularities” in the interactions. In this study, this procedure was used in a slightly adapted form which is described in section 3.2.1.
6. **Procedure: Video review sessions** In this procedure the video segments are played back to the participants. The intention of this step is to include the participants’ perspective for the analysis to gather further insights. This procedure was not practical in the context of this study and was replaced by analysing the interviews that were conducted after the sessions and which provide the developers’ perspectives on their PP sessions.

3.2.1 **Analysis steps**

In the following sections the specific analysis steps in the order in which they were conducted are described.

**Step 1: Sampling of relevant video exemplars**

Interaction analysis is a very detailed and time-consuming analysis procedure which means that sampling of the relevant video exemplars is a crucial step. A detailed analysis of a 3-5 minute exemplar can take about 2 hours. At the start of this analysis, we had about 37 hours of video recordings of 21 PP sessions with 31 developers from four different companies. Each session lasted between one and a half and three and a half hours. In previous studies [29, 27] we had analysed the full 37 hours using different analysis methods. Hence, we had excellent knowledge of the data before the Interaction Analysis started. This allowed us to effectively identify exemplars that exhibit typical situations in which knowledge transfer between developers takes place. Five suitable exemplars were identified, each between 4 and 6 minutes in length, that together represented the set of typical situations in the data. The following criteria guided our selection.

- **Developers with different levels of knowledge are aiming to transfer knowledge.** Developers never have exactly the same knowledge, and so it can be assumed that knowledge transfer takes place in every PP session. However, in PP sessions where developers have similar levels of expertise, knowledge transfer and the strategies used to achieve it might be difficult to identify. In contrast, where pair constellations have an explicit aim to transfer knowledge from one
developer to another, knowledge transfer and associated strategies can be observed more easily and explicitly. Moreover, in these sessions it is clear who is teaching whom. Hence, we used the background questionnaires and the interviews with the developers to identify sessions in which a more knowledgeable developer (expert) worked with a less knowledgeable developer (novice).

In the context of this study, the definition of an expert and a novice is based on the developers’ perceptions of their knowledge for a particular pair programming session. We chose excerpts from PP sessions for which developers explicitly stated that the aim of the session was to transfer knowledge from the more knowledgeable developer (expert) to the less knowledgeable developer (novice) for the topic covered in this session. In addition we only chose sessions in which both developers agreed who is the expert and who is the novice in this particular session.

- **Experts are trying to teach novices.** To address RQ1 excerpts were selected in which experts tried to teach the novices rather than excerpts in which no communication, explanations or verbalisations took place.

- **Both expert and novice are driving within the excerpt.** To answer RQ2 some excerpts were chosen in which the novice was driving and some in which the expert was driving or where the developers switched roles.

- **Behaviour is not unique to one pair constellation.** Although we only selected a small number of exemplars to conduct the in-depth interaction analysis the first author watched all video recordings to ensure that the selected excerpts represent behaviours observed in multiple pair constellations.

**Step 2: Transcription**

The conversations of the developers together with timestamps and pauses were transcribed. Short pauses in the communication are marked as […] and for longer pauses the number of seconds [sec] is provided in the transcriptions. The collaborator companies were based in Germany and all audio data was in
German. The transcription was based on the original data. The main researcher who conducted the data gathering is a native German speaker and the other researchers have different language skills (see table 2).

**Step 3: Group viewing**

The video excerpts were analysed collaboratively during group viewing sessions. The group size varied between two and five members. In each group viewing session at least one German and one non-German speaker were present. Table 2 provides an overview of the members of the group, their language skills and their relevant experience. Each group member had a different background. This strengthened the analysis as each member provided a different perspective on the data. For example, the different language skills of the group members influenced the initial focus when watching the excerpts; usually, the non-German speakers focused first on the non-verbal communication and on the computer activities of the developers while the German-speaking members focused on the conversation first. To allow every member to understand the full picture of the events the transcripts were translated during the group viewing session.

For each video excerpt (4-6 min) the group session took approximately 2 hours in order to accommodate intense discussions. During the whole group viewing session the first author took detailed notes of the discussion. The group viewing session had the following activities (steps 3 and 4 were iterated):

1. **Watching the whole video excerpt** The group watched the excerpt from the beginning to the end without stopping. The group was not provided with the context of the excerpt before watching it, in order to counter researcher bias.

2. **Discussing initial observations** Each team member shared their initial observations with the group, drawing on their specific expertise.

3. **Providing context for the segment** After the initial discussion, the group members were told the context of the excerpt: who is the expert and who is the novice; is this excerpt from a beginning,
middle or end of a session; and what developers reported in the interviews about their knowledge transfer experience.

4. **Stop-and-go watching of the video excerpt** The excerpt was watched again but this time, team members stopped the video to discuss different dimensions of the interaction in detail. During this, certain events or the whole excerpt were re-played several times.

5. **Finding themes** The results of group viewing sessions were brought together and the emerging themes were discussed.

<table>
<thead>
<tr>
<th>Research experience</th>
<th>Relevant languages</th>
<th>Relevant experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experienced researcher</td>
<td>English</td>
<td>Empirical software engineering, qualitative research, research on agile methods</td>
</tr>
<tr>
<td>Experienced researcher</td>
<td>German (intermediate), English</td>
<td>Empirical research, qualitative research, gesture analysis, advanced programming</td>
</tr>
<tr>
<td>Experienced researcher</td>
<td>German, English</td>
<td>Empirical research on cooperative and human aspects of software engineering, interaction analysis</td>
</tr>
<tr>
<td>PhD student</td>
<td>English</td>
<td>Empirical research, qualitative research</td>
</tr>
<tr>
<td>PhD student</td>
<td>German, English</td>
<td>Empirical research, qualitative research, pair programming research</td>
</tr>
</tbody>
</table>

The analysis was exploratory and did not follow a pre-defined coding scheme. During the group viewing, observations were not restricted to the research objectives because a restriction early on in the
Step 4: Analysis of group viewing notes and interview data

The themes that emerged during the group viewing were reviewed with respect to the research questions. The first author reviewed the extensive notes from the group session and revisited the data for each theme for validation. The results of this procedure were discussed with other group members. In addition, the interviews from the three selected PP sessions were analysed focusing on developers’ statements related to their experience of knowledge transfer within their sessions. The findings from the interviews were then used to contextualise the findings from the group viewing session.

4 Findings

This section presents the findings focusing on the teaching strategies used by developers (RQ1) the roles of driver and navigator (RQ2) and the challenges when pairing developers with different knowledge levels (RQ3). Our findings are illustrated through examples from the selected transcriptions and descriptions of what developers were doing. The analysis highlighted that experts use a combination of different strategies to teach novices and that each expert uses a variety of strategies even within the same pair programming session.

4.1 What teaching strategies do developers use when a novice is driving?

We identified four different strategies that experts use when the novice is driving: Verbal nudging and physical hints, pointing out problems, gradually adding information, and giving clear instructions.

4.1.1 Verbal nudging and physical hints

Verbal nudging and physical hints are teaching strategies that provide directions without providing the solutions for the novice. For example, using verbal nudging an expert will make suggestions rather than
explicitly tell the novice what to do. Other subtle guidance can include physical hints such as pointing to something on the screen or placing the cursor in a particular location. These different types of behaviours are illustrated in Example 1 and Example 2.

In Example 1, the novice is driving and has just finished writing a line of code that checks an entry in a list. The novice suggests that they can now move on (line 1) but the expert proposes (using the word “could” in line 2) that they should test this code first. The novice agrees without any resistance and instead of moving on, the novice starts testing the code.

**Example 1 Nudging**

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“Ok, so this is done now, so we can move on to the next bit.”</td>
</tr>
<tr>
<td>2</td>
<td>Expert:</td>
<td>“We could also test that first.”</td>
</tr>
<tr>
<td>3</td>
<td>Novice:</td>
<td>“Yes, ok.”</td>
</tr>
</tbody>
</table>

In Example 2, the expert places the cursor in a certain position before handing the keyboard over to the novice. Later in the excerpt, it becomes apparent that this is the point in the code where the problem should be addressed. The fact that the expert switched to this test class after finishing his explanations and while preparing the handover, indicates his intention to provide a hint for the novice.

**Example 2 Indirect hint: Preparing the environment**

The expert is driving and explains a problem to the novice. The expert opens different Java classes and test files to illustrate his explanations. He finishes his explanations by pointing something out in a Java class. Afterwards he says with a smile on his face: “OK, so now it is time to switch driver.” While saying this, he switches from the Java class to a test class, looks for a specific location in this test class and moves the cursor there before he hands the keyboard to the novice.
The excerpts illustrate that verbal nudging and physical hints are used by experts to provide a learning opportunity for the novice. In Example 1, the expert uses a gentle form of verbal nudging which is immediately picked up by the novice. Without telling the novice how exactly to solve the problem, the expert provides successful directions. Following the excerpt in Example 1, the novice starts writing a test. In addition to verbal hints, we also observed indirect non-verbal hints where an expert physically moves the cursor around the programming environment to nudge the novice towards the right place. Example 2 illustrates such a situation in which the expert sets up the environment for the novice before handing over the keyboard to provide the novice with a starting point. These subtle strategies were observed in pairs where the expert seemed to be patient and the novice had some initial knowledge of the task at hand.

4.1.2 Pointing out problems

Experts point out problems for novices without suggesting how the problem should be solved. This is illustrated in Example 3.

**Example 3 Pointing out a problem**

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“I see at least three mistakes.”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“You see three mistakes?”</td>
</tr>
<tr>
<td>3</td>
<td>Expert:</td>
<td>“I see three mistakes.”</td>
</tr>
<tr>
<td>4</td>
<td>Novice:</td>
<td>“OK.”</td>
</tr>
<tr>
<td>5</td>
<td>Expert:</td>
<td>“Twice <code>statement.close</code> and one uninitialized member variable.”</td>
</tr>
<tr>
<td>6</td>
<td>Novice:</td>
<td>“Yes.”</td>
</tr>
</tbody>
</table>
In Example 3, the novice has finished writing some code. The expert points out that there are at least three mistakes in it without explicitly explaining how to address them. The expert thus gives the novice the space to think about how to solve the problem without simply solving it for him and the novice starts suggesting how to address the problems. In comparison to verbal nudging and physical hints, in the example above the novice receives no direction on what to do next. This strategy prompts the novice to suggest how to address the problems, giving the novice the opportunity to think the problems through by herself/himself. We observed that this strategy can be time-consuming because it might take the novice some time to identify solutions to the problem. In cases where this strategy does not work, we observed that experts use a follow-up strategy (gradually adding information). This strategy is presented next and is also used independent of the pointing out problems strategy.

### 4.1.3 Gradually adding information

Gradually adding information means that the expert supports the novice in finding a solution for a problem on an “as needed” basis. Instead of suggesting how to solve an issue experts wait and see whether novices are capable of solving a problem by themselves with a certain amount of information given. If the novice is not capable of solving the task, the expert gradually adds more information in order to help the novice (as illustrated in Examples 4 and 5).

**Example 4** Gradually adding information (1) and Giving clear instructions

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“We would maybe need a constructor here, wouldn’t we?”</td>
</tr>
<tr>
<td>2</td>
<td>Expert:</td>
<td>“Right. That would be good.”</td>
</tr>
</tbody>
</table>
No communication, no typing [3sec].

**Expert:** “At the top.”

Novice is moving the cursor to the top of the class.

**Expert:** “ALT-N.”

Novice presses ALT-N.

---

**Example 5** Gradually adding information (2)

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“Ok, technically, we’d have to call a [...] job instead of calling a PreSQLStatements.”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No communication/no driving [3sec].</td>
</tr>
<tr>
<td>2</td>
<td>Expert:</td>
<td>“In our case, we’d call a [...] job instead now. Right.”</td>
</tr>
<tr>
<td>3</td>
<td>Novice:</td>
<td>“Right. Ok, that means we somehow build us a class now that then just gets the job and executes it.”</td>
</tr>
<tr>
<td>4</td>
<td>Expert:</td>
<td>“Eventually, that would be the implementation. Yes.”</td>
</tr>
<tr>
<td>5</td>
<td>Novice:</td>
<td>“Good.”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No communication/no driving [2sec].</td>
</tr>
<tr>
<td>6</td>
<td>Expert:</td>
<td>“We just have to tell the SQL FilterStatement somehow [...] that it executes it at the right time.”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No communication/no driving [3sec].</td>
</tr>
<tr>
<td>7</td>
<td>Expert:</td>
<td>“Instead of using PreSQLStatement; [...] use PreSQL or PreListener or PostListener or all in one. I don’t know.”</td>
</tr>
<tr>
<td>8</td>
<td>Novice:</td>
<td>“Yes. Ok, but we would want to do that then by using the Config-class.”</td>
</tr>
</tbody>
</table>

Novice puts his hands on the keyboard and starts typing.
In Example 4, the novice is driving and suggests creating a constructor. He phrases this suggestion as a question (line 1). The expert confirms that the novice’s suggestion is right without adding more information (line 2). The novice hesitates to start typing. After a short pause, the expert gives the novice additional instructions by telling him that the constructor should be written at the top of the class and that he should use the shortcut “ALT-N” to do that.

This can be interpreted as the expert waiting to see whether the novice can create the constructor without more information and only giving instructions when the novice hesitates.

Example 5 presents a second example of this strategy. Just before the start of this excerpt, the expert had explained the code and the problem and then made it verbally explicit that it’s the novice’s turn to drive. The novice does not immediately take the keyboard or mouse. Instead the novice starts suggesting what to do next (line 1 and 3). In lines 2 and 4, the expert repeats and confirms the novice’s suggestion without adding new information. When it is the novice’s turn to talk and type, the novice hesitates again. The expert starts adding further information about how to solve the problem. After adding new information (line 6) the expert waits. The novice does not react and after another moment of silence the expert starts again to add new information (line 7). This time (line 7) the expert finishes his statement by saying that he does not know what the best approach would be. In line 8, the novice suggests how to address the problem and subsequently takes the keyboard and starts implementing the solution. The expert confirms the novice’s suggestion in line 9.

Later on in the session (not provided as a transcript), it becomes clear that the expert knew how to approach that problem, and that the “I don’t know” statement in line 7 was a hint that the novice should solve the problem.

Both examples 4 and 5 show an expert not giving all the information to the novice at once but gradually adding more information when necessary. This suggests that the expert wants the novice to actively
think about the problem rather than for him to just explain each step immediately. This also means that the problem solving process might take more time in comparison to having the expert telling the novice explicitly how to solve the problem (as described in the next section).

4.1.4 Giving clear instructions

Experts also use clear and direct instructions, both with and without additional explanations, to help guide the novices. These instructions include dictating what to type, which shortcuts to use and where the changes have to be made in the code. In some cases, experts explain their instructions thereby providing the novices with the reasoning behind those instructions. In both cases, it was observed that the novice follows the instructions of the expert immediately.

Looking back at example 4, this acts as a good example of an expert giving clear instructions when noticing that the novice knows what to do (line 1), but hesitates about how to go about it (line 3). The expert gives the novice clear instruction where to create it (line 4) and how to do it (“ALT-N” in line 5) without providing any additional explanation.

A similar example of providing clear instructions on what to do is shown in example 6. However, here we also note the expert providing an explanation on why the code had to be placed elsewhere (line 2).

In contrast to the three previous strategies, in this strategy the experts solves the problem for the novice. This strategy doesn’t encourage the novice to solve the problem for himself/herself. This might be less time-consuming but also provides the novice with less opportunity to explore different approaches. However, in some cases this might be the only sensible strategy. For example, there is little value in asking a novice to look up a shortcut rather than telling the novice what the shortcut is.

Example 6 Giving direct instructions with explanations

<table>
<thead>
<tr>
<th></th>
<th>Novice:</th>
<th>“Yes, so then ... we can create a nice method now.”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice start to create a new method.</td>
<td></td>
</tr>
</tbody>
</table>
Novice deletes the already written code and creates the new method outside of the inner class.

### 4.2 What teaching strategies do developers use when expert is driving?

Focusing now on strategies where it is the expert who is driving, rather than the novice as in the previous section, we identified two strategies: explanations and verbalisations. It is difficult to differentiate between explaining to oneself (verbalisation) and explaining interactively [14]. In this paper, verbalisation refers to verbalising while performing activities without being asked for it, while explanations are statements triggered by a question or a comment that makes it clear that the other person needs additional information.

#### 4.2.1 Explanations

When the novice asks for an explanation, experts can address the question verbally (Example 7) or by showing the novice on the computer how to do certain steps (Example 8).

**Example 7 Explanation**

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“So, and now we create some methods and call them test. <code>BasicFilter</code>, isn’t it?”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“Was the style back then so different that, hm […] , that err... all tests were in one method?”</td>
</tr>
</tbody>
</table>

Expert keeps typing and replies while typing.

| 3    | Expert: | “Err... no in terms of style not but err that happened quite often back then just because err...” |

Expert keeps typing for 5 sec without any comments and then briefly verbalises what he is typing

<p>| 4    | Expert: | “<code>PreparedStatement</code> with S, erm” |</p>
<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“So, now you can go over it and see whether it finds it. Have a look whether the activities exist. So, you can [...]”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“The activities?”</td>
</tr>
</tbody>
</table>

Example 8 Explanation by showing

Example 7 illustrates how the expert addresses the novice questions verbally. In the excerpt, the developers are amending tests that have been written before. The expert is driving and the novice notes that the structure of the old tests does not conform to the current coding policy. He asks (line 2) the
expert about the different styles. The expert starts explaining how the coding standards evolved and at
the same time expresses his displeasure about the old structure of the existing code.

In Example 8 the expert initiates a short role switch to show something to the novice. The novice is
driving and the expert suggests what the novice should do next, which requires knowledge about the
debugger. The novice does not seem to know how to do that. The novice leans back, looks at the expert
and repeats the keyword “activities” as a question (line 2). The expert takes the mouse, shows the novice
what he meant and takes his hands immediately back from the mouse.

In both cases, the expert addresses the novice’s request for information. Depending on the information
request it seemed to be easier to explain the knowledge verbally (for example, how do certain processes
work, why have specific decisions been made) while in other cases it seemed to be easier, more
convenient and maybe less time consuming to show steps rather than explain them.

4.2.2 Verbalisation

Experts verbalise their activities and thoughts while driving. Verbalisation is not necessarily directed to
the novice, but might help the expert to structure his/her own thoughts; as a result, it may help the
novice to understand the expert’s thought process.

**Example 9 Verbalisation**

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“Now, let’s have a look. I just open the tests. This is where I had started already. I just wanted to show that to you and then you may (drive) as well.”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Expert opens the test and then starts browsing through the code.</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“And this (test) always has everything in it? It will get big.”</td>
</tr>
<tr>
<td>3</td>
<td>Expert:</td>
<td>“Right, yes, that will get a bit bigger. I leave it all in here for now and then we can think about whether we should create a second test class.”</td>
</tr>
<tr>
<td></td>
<td>Novice:</td>
<td>“And for this test, do all the other tests have to be run through before? Or could we get a specific one?”</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>5</td>
<td>Expert:</td>
<td>“So, I thought, we have this “before” here. Let’s go through it again. It creates one CommissionCalculation. That is here and it ... the DNS. This [experts navigates with mouse through the code] is adding a hint, this is adding the “invoices”, and this one is adding a First-WorkFlow among others. It sets the status in created, added and sets this as current Workflow.</td>
</tr>
<tr>
<td>6</td>
<td>Novice:</td>
<td>“Yes, this is what it just did.”</td>
</tr>
<tr>
<td>7</td>
<td>Expert:</td>
<td>“And we also made sure it synchronises this automatically. Add the “recipients”, err [...] maybe that is not as important at the moment.”</td>
</tr>
</tbody>
</table>

In Example 9, the expert had worked a particular part of the code before and now goes through it again in order to explain the code to the novice (line 1). After some browsing through the code, the novice asks a question about the size of the test (line2) and while the expert responds (line 3) he indicates that they should not focus on the size issue. During this exchange the expert is very focused on the screen (his eyes never leave the screen) and on understanding the code that he had written before. The subsequent question by the novice (line 4) is in fact ignored by the expert (line 5) who proceeds to verbalize his thought process about how the code works.

The fact that he does not react to her comments indicates that he verbalises his thoughts for himself rather than to provide explanations for her, because if his main interest was to explain the code to her, he would pay attention to her comments. However, he stated in line 1 that he does want to explain the code to her, so it may be that he is concentrating on verbalising his thoughts and can’t take her viewpoint into account. This stresses that there might be a conflict between self-verbalisation and verbalising for the partner (this is discussed in detail in section 4.4).
4.3 In which ways do the roles of driver and navigator influence knowledge transfer in pair programming?

The role being taken had a clear influence on the developers’ interactions, and on the novice’s engagement. In a previous study [29], we have shown that novices are at risk of disengaging from the PP session when navigating and when there is a lack of communication. In contrast here we focus on excerpts where experts and novice communicate and on excerpts when the novice is driving. We observed that the novices tend to be more active when driving. This shift in behaviour is illustrated in the next section. It also became apparent that novices are articulating what they are doing when driving to get reassurance from the expert as described in section 4.3.2.

4.3.1 From a listener to an active participant

We observed that novices act more like listeners when navigating and turn into active participants that verbalise their behaviours and ask questions when driving. Examples 10 and 11 describe role switches, demonstrating the contrast between the novice’s behaviour when navigating with that of being the driver. Example 11 is the same as Example 5, but it is repeated below to emphasise that it is a continuation of Example 10 in the PP session.

**Example 10 Role switch**

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line 1-10: Expert is driving. He keeps moving the cursor to the code fragments that he explains. The novice has his hands under the table.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“And that of course, leads to the point that not all entries are used, but […]”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“Mhm, ok.”</td>
</tr>
<tr>
<td>3</td>
<td>Expert:</td>
<td>“Or if you call execute twice at the same Prepared-Statement […]”.</td>
</tr>
<tr>
<td>4</td>
<td>Novice:</td>
<td>“OK.”</td>
</tr>
</tbody>
</table>
Expert: “And that of course leads to the fact that even when the fields are not used [...]”

Novice: “OK.”

Expert: “So and we do the same somehow when the method is close and we [...]”

Novice: “Mhm.”

Expert: “That is the code that we have [...] . What we want to do now - aside from getting the big learning effect - err, is of course, eh, to instead of just using SQL, to ehm, it would be easier [...]”

Novice: “Mhm.”

Expert: “OK? So then we can change driver now..”

Example 11 Role switch (continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Novice:</td>
<td>“Ok, technically, we’d have to call a [...] job instead of calling a PreSQLStatement.”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No communication/no driving [3sec].</td>
</tr>
<tr>
<td>13</td>
<td>Expert:</td>
<td>“In our case, we’d call a [...] job instead now. Right.”</td>
</tr>
<tr>
<td>14</td>
<td>Novice:</td>
<td>“Right. Ok, that means we somehow build us a class now that then just gets the job and executes it.”</td>
</tr>
<tr>
<td>15</td>
<td>Expert:</td>
<td>“Eventually, that would be the implementation. Yes.”</td>
</tr>
<tr>
<td>16</td>
<td>Novice:</td>
<td>“Good.”</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No communication/no driving [2sec]</td>
</tr>
</tbody>
</table>
| 17   | Expert: | “We just have to tell the SQLFilterStatement somehow [...] that it
executes it at the right time.”

<table>
<thead>
<tr>
<th>18</th>
<th>Expert:</th>
<th>“By instead of doing PreSQLStatement [...], PreSQL or PreListener or PostListener or all in one. I don’t know.”</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>Novice:</td>
<td>“Yes. Ok, but we would want to do that then by using the Config-class.”</td>
</tr>
</tbody>
</table>

Novice puts his hands on the keyboard and starts typing.

<table>
<thead>
<tr>
<th>20</th>
<th>Expert:</th>
<th>“Right. That is our only chance to intervene there.”</th>
</tr>
</thead>
</table>

In the first 11 lines, the expert is driving and explains the code to the novice, using the mouse to highlight code snippets that are relevant for his explanation. While the expert is in control of the mouse, the novice keeps his hands under the table indicating that he is not trying to take on the role of driver. The novice appears to be an active listener but he is not asking any questions. In line 11, the expert mentions that it is time to switch roles. This is when the body language and the verbal involvement of the novice changes. The novice puts his hands on the desk but does not take the mouse or keyboard. He starts asking questions and makes suggestions about the actual implementation (Example 11 lines 12, 14, 19). The expert initially repeats his suggestions without adding new information and then adds information gradually (section 4.1.3). Eventually, the novice takes the keyboard and starts typing.

This example illustrates the novice’s change of behaviour from being a passive participant (as navigator) to a more active participant (as driver). As a navigator it might be enough to understand the underlying concepts but as a driver detailed knowledge is required of the code and the next steps are required to perform the implementation. This means that letting the novice drive can be useful to ensure that detailed knowledge is transferred. However, it also means that the process of solving the task at hand might be slower as the novice will need detailed information about the task and possible solutions.
### 4.3.2 Articulation and re-assurance

Novices tend to articulate their thoughts and verbalise their steps, their suggestions for solutions and reasoning behind them. A closer look at the verbalisation shows that novices seek for re-assurance about their actions from the experts. This is illustrated in Example 12. The novice verbalises her plans for the next steps (lines 1 and 3) and also asks for reassurance (line 5). The comments and questions are very detailed. This shows that letting the novice drive can encourage both expert and novice to be actively involved in discussing, understanding and solving the task at hand.

**Example 12 Novice driving**

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Developers just switched roles and the novice takes the keyboard and starts creating a for-loop to check entries in a list.</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“So, now here we have a “subject”. [...] “From”. [...] So, I just go through [the list] now and when I find one [activity], what am I doing with it then?”</td>
</tr>
<tr>
<td>2</td>
<td>Expert:</td>
<td>“Technically, it should only find one, if we were good, then it should have only one “activity” with that “subject”.”</td>
</tr>
<tr>
<td></td>
<td>Novice deletes the for–loop.</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Novice:</td>
<td>“Then, then I say “assert”.”</td>
</tr>
<tr>
<td>4</td>
<td>Expert:</td>
<td>“Right.”</td>
</tr>
<tr>
<td>5</td>
<td>Novice:</td>
<td>“So we just check afterwards? ”</td>
</tr>
<tr>
<td>6</td>
<td>Expert:</td>
<td>“Technically yes. Just, technically, it would be enough that there is exactly one “activity”.”</td>
</tr>
<tr>
<td>7</td>
<td>Novice:</td>
<td>“Ah, ok, in the test I can see, that, what is in there. I’m still with my old <em>Sys .Out.</em>”</td>
</tr>
<tr>
<td>8</td>
<td>Expert:</td>
<td>“No, we can debug that.”</td>
</tr>
</tbody>
</table>
What challenges do developers with different knowledge levels face when pairing together?

In a previous study [29], we analysed and presented challenges faced by novices in expert–novice constellations. For this paper, we focus on the challenges that experts face when trying to transfer knowledge. Experts seem to face challenges when they are driving and guiding the novice at the same time.

4.4.1 Conflict between self-verbalisation and communication with partner

Example 9 illustrates the conflict of self-verbalisation and communication with partner. It is particularly interesting because the expert was intending to explain the code to the novice (line 1) but ends up focusing on understanding the code himself.

As the session continues (shown in Example 13) the expert still focuses on the code and on his approach to solving the problem. The novice has a different problem solving approach and tries to communicate that idea to the expert but is not immediately successful. The expert reacts to the novice’s comments but does not really take the novice’s suggestions into account. However, then the expert realises that the novice had the right idea (Example 13, lines 7 and 10).

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“The first checkpoint should have gotten an email already. Just because I saved it here in the status edit [...] because it is now in status created”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“None.”</td>
</tr>
<tr>
<td>3</td>
<td>Expert:</td>
<td>“Created and None.”</td>
</tr>
<tr>
<td>4</td>
<td>Novice:</td>
<td>“Yes, Created and None.”</td>
</tr>
</tbody>
</table>
5  Expert: “That means, if our logic is correct, what we just programmed, then we could indeed already search for an “activity”, that has the subject “from”, err, err from a work-flow that goes to the first checkpoint.”

6  Novice: “Ah, but I understood that at the interface, the creator, so that the calculation will be delivered to the creator. Hence, the interface, the “wizard” sets the “workflow” to “first prove”. And we are at “first prove”. That means it must have passed it on.

7  Expert: “You’re right. I think this is how we did it here, didn’t we? I have to have a look again.”

Expert navigates to another window.

8  Expert: “So, at the moment, here I can orientate myself better.”

9  Novice: “Yes, yes.”

10  Expert: “You’re right. For sure. That means that nothing has happened here yet. You’re right, completely right.”

The behaviour of the expert indicates that he is not able to cope simultaneously with the novice’s suggestions, with understanding the code and structuring his thoughts. However, neither of the developers seems to realise that; the expert does not ask for time to finish his thoughts nor does the novice stop making suggestions.

This issue of thinking and communicating at the same time was also identified by experts during the interviews: “...complex, analytical and problems related to architecture. I prefer to work those kind of problems through in my head first. [...] My point is I cannot communicate and share my thoughts when I have to think about complicated problems.”

4.4.2 The effort needed to explain

Providing explanations for the novice can be an additional effort for the expert.
In Example 7 the expert explains former coding conventions while working on the current tests. Hence, the expert has to switch context between his current activity of working on tests and his explanation about former coding conventions. In this case, the explanation is not directly related to his current activity. The expert’s explanations about the former coding conventions are frequently interrupted by breaks (err or pauses) and by verbalising what he is currently typing (line 4). This indicates the effort of switching the context between his task and his explanation. Furthermore, his typing slows down while he is explaining.

In contrast, Example 8 provides an episode in which the explanation appears to be effortless for the expert. In that situation, both developers focus on the same problem and the expert explains the use of the debugger which is directly related to his current activity. This indicates that explanations that are not related to the current activity can lead to an additional cognitive effort.

4.4.3 Benefits of verbalisation and explanation

Analysis of the interviews showed that novices can help experts to challenge their assumptions and reflect on the existing code and that working with a novice provides them with opportunities to learn themselves. These quotes from two of the interviews illustrate the developers’ perspectives:

“\textit{It is good to work with her. She is always asking the right questions. I don’t perceive that as slowing me down [...] so she is asking automatically the right questions and that forces me to think about what I’m actually doing.}” and

“\textit{It is really helpful to work with him. When I work with someone who is already familiar with the code, the risk is that we overlook things because we have always done it like this [...] That happens less when working with a newbie because he does not know these parts of the code and so he asks questions about it. And then I feel like I’m being forced to reflect and to explain what the software is doing.}”
5 Discussion

This section returns to the three research questions.

5.1 RQ1: What teaching strategies do developers use in pair programming?

Six strategies which experts combine to guide and teach novices in PP sessions emerged from our data analysis: (1) indirect hints, (2) pointing out problems, (3) gradually adding information, (4) giving clear instructions, (5) explanation, and (6) verbalisation.

The knowledge transfer aspect of pair programming can be viewed as a kind of apprenticeship, but traditional apprenticeship involves learning a physical activity [11, 13] through social interactions while focusing on a task. Cognitive apprenticeship, on the other hand, focuses on learning “cognitive and meta-cognitive, rather than physical skills and processes” [12, p. 3]. Collins et al. [12, 11] stress the importance of making tacit processes visible for learners by making thinking visible. Given that software development requires cognitive and meta-cognitive skills and that the key to understanding software development is the reasoning and concepts behind it, rather than the physical act of typing, cognitive apprenticeship shares the same characteristics, and is therefore relevant to learning and improving software development skills.

Comparing our strategies with the teaching methods suggested in cognitive apprenticeship [12], they can be viewed as specialisations of the teaching methods described there. This then provides further insights into strategies that might be used in PP as described below where we systematically relate the knowledge transfer strategies that emerged from our data with the teaching methods put forward by Collins:

Comparing our strategies with the teaching methods suggested in cognitive apprenticeship [12], and viewing them as specialisations of the teaching methods described there provides insight into strategies that might be used in PP. The six teaching methods suggested by Collins et al. and how they relate to the six knowledge transfer strategies are described below.
**Modeling** refers to the expert demonstrating a task and verbalising his or her thoughts at the same time to make the process of thinking visible. Strategies (5) and (6) are examples of Modeling: experts verbalised their thoughts while the expert is driving, and explained more when asked to.

**Coaching and Scaffolding** describes the process of observing the learner while solving a task and providing support. Strategies (1)-(4) are examples of Coaching and Scaffolding used when the novice was driving. Strategies (1) and (2) make the novice solve the problem themselves. Strategy (3) helps to identify how much help the novice needs. Scaffolding should consider the current skill level of the learner as described by Vygotsky’s “Zone of Proximal Development”. He says that learners might be able to solve a task in collaboration with a more capable peer that they would not be able to solve independently. In contrast, strategy (4) does not enforce the same level of novice engagement. ”Giving clear instruction” is a less time-consuming Coaching approach but learning may suffer as the novice is not thinking for themselves.

**Articulation** refers to learners being encouraged to articulate their knowledge, reasoning and problem-solving processes as Articulation refines the learner’s understanding. In none of the sessions did the expert explicitly encourage the novice to articulate their knowledge. Some form of Articulation was observed when the novice was driving (4.3.2). In this case, it seemed that articulation was used as a method to get reassurance from the expert rather than to make thinking visible.

**Reflection** means that learners compare their own problem-solving process with those of others. This behaviour was not observed during our study.

The method “exploration” is not included in the discussion because this method refers to helping a novice to choose suitable follow-up tasks to foster and advance their learning, and identifying such tasks would not normally be decided during PP sessions.

---

2 The strategies Coaching and Scaffolding were merged because Scaffolding is one form of Coaching and no clear delineation is provided
So without any prior training, experts in a PP session are using cognitive apprenticeship teaching methods for knowledge transfer. But why don’t pair programmers use all of these strategies? Firstly, developers may just not be aware of cognitive apprenticeship. Secondly, Collins describes the methods in the context of education which differs from the context of PP. Although developers work together with the explicit aim to transfer knowledge from an expert to a novice, tasks are typically chosen according to agile prioritisation which considers business value, not educational progression. Moreover, knowledge transfer is not the exclusive aim of the session because developers work on their real-world tasks with the focus of finishing the project on time. This means that developers have to balance knowledge transfer and getting the task done effectively.

One strategy from cognitive apprenticeship that could be applied to improve knowledge transfer in PP is that of articulation. Encouraging the novice to articulate was not observed in our PP sessions, yet it could expose novice’s thoughts. Vygotsky’s [39] “Zone of Proximal Development” could be used to identify suitable tasks and pair constellations to ensure that the task is manageable for the novice and that developer skill levels are not too far apart.

5.2 RQ2: In which ways do the roles of driver and navigator influence knowledge transfer in pair programming?

Experts adapt their teaching strategies according to their role: they engage the novice through explanations and verbalisations while driving and guide the novice with instructions while navigating. Novices’ behaviours are also influenced by whether they are driving or navigating.

Novices become more active and engage on a more detailed level when driving than they do when navigating. Novices are encouraged to think through, understand and solve parts of the problems by themselves when using strategies (1)-(3) and to perform the necessary steps by following the expert’s instructions when using strategy (4). This means that novices “learn by doing” when driving rather than observing. Example 11 shows that the novice asked more questions as soon as the expert suggested a
role switch, thus seeming to engage more with the task. In the interviews, experts and novices agreed that if the novice drives it is beneficial for the novice learning but acknowledged that having the novice drive takes more time to solve the task than when the expert drives. This is in line with existing research that an “extremely effective style of learning by doing” occurs when learners solve as much of the work as possible guided by a tutor [22].

The novice’s behaviour in Example 11 indicates that novices hesitate when it is their turn to drive (in line with the findings in [29]) even though it is supposed to be beneficial for their learning process. Plonka et al. [28] showed that often the expert dominates the driving in expert–novice constellations. This emphasises how important it is for the expert to encourage role switches, and for the novice to be prepared to drive.

5.3 RQ3: What challenges do developers with different knowledge levels face when pairing together

Expert–novice constellations provide learning opportunities for both partners, and experts also face challenges working with a novice. We do not report on challenges for the novice in this paper, but novice challenges such as social pressure are described in [29].

5.3.1 Opportunities for expert learning

Expert–novice constellations focus mainly on knowledge transfer from the expert to the novice, but this research identified learning opportunities for experts as well. Novices are usually less familiar with the code than the expert, so they can provide a different perspective on the problem and existing code. Novices might ask “simple questions” (see Example 7) that force the expert to rethink previously-held ideas thereby uncovering problems and leading to code improvements. The novice might also suggest solutions that an expert had not seen or considered (see Example 13).

Novices have a “beginner’s mind”, which refers to people who are unfamiliar with a situation and who might consider more possibilities than an expert. Belshee [3] points out the positive effect of the
beginner’s mind on knowledge transfer in PP and in the context of requirements engineering. Berry [4]
points out that novices might ask questions that help to expose assumptions by experts.

In the context of expert–novice constellations, this means that experts can benefit and learn from the
fresh perspective of the novice and that simple questions might help the expert to challenge their own
assumptions about the existing code.

Another potential for expert learning is the process of verbalisation; talking about a problem might lead
to a better understanding of the problem itself [8]. In the interviews, experts stated that explaining their
thoughts to the novice helped them to structure their thoughts and think them through more thoroughly.

Cao and Xu [7] also found that the process of verbalisation was helpful for the expert in order to readjust
goals and reorganise thoughts even when the novice did not react to the explanation.

5.3.2 Expert challenges

On the other hand, explaining and verbalising can be an additional cognitive effort for the experts. In
Example 9, an expert was seen to struggle while working through a complex problem. The expert was
using verbalisation to assist himself to work through the problem. It was challenging because the expert
was verbalising in order to explain the code to the novice, and so the novice was free to ask questions
and make comments during this time. The expert was not able to deal with both his verbalisation and the
novice’s comments and questions. This stresses the importance of being aware that developers might
verbalise to structure their own thoughts and hence might not be able to react to their partner’s
comments until they have finished this process.

PP has been described by some as an exhausting practice [41, 37]. While [41] ascribes this to developers
focusing more on the task due to the pair pressure, our research shows that the effort of verbalization
and explanation could be another reason why PP is more tiring than solo programming.
6 Limitations

Whilst great care was taken in designing and conducting the study, we acknowledge that some issues in the data gathering, research method and the analysis could be viewed as limitations of the research. In particular we identify the following issues: Voluntary participation (potentially affecting completeness), developers being observed and recorded (potentially affecting generalizability), and detailed qualitative analysis based on a relatively small amount of data (potentially affecting completeness and generalizability). Note that none of these limitations affect the validity of the findings.

6.1 Voluntary participation

Participation was voluntary for all developers. That is, developers from the identified companies were invited to participate, but no pressure was used to attempt to get all developers from each company to be involved. This means that developers who are confident in their pair programming skills or have a positive attitude towards pair programming might have been more likely to participate. This in turn may have affected the findings because less confident developers or developers with a negative attitude towards pair programming might exhibit different behaviours. This limitation potentially affects the completeness of our results.

6.2 Developers being observed and recorded

Video and audio data were gathered during the pair programming sessions. Developers were informed about the recording before they started their sessions and it is possible that they modified their behaviour due to the fact that they knew they were being studied. To minimise the effect of participants feeling observed the recording setup was integrated in the developers workplace (using a webcam and wireless microphones that do not restrict any kind of movement, for example getting up from the chair). Additionally, in the interviews, developers were asked whether they had felt conscious of being observed. Some developers stated that they were aware of the webcam, others stated that they
completely forgot about it once they started working on the task. Some developers even replied that they
felt more observed by their partner than by our recording setup. This last statement suggests that pair
programming sessions are situations in which developers might feel observed even without being
recorded. This limitation potentially affects the generalisability of our results.

6.3 Detailed qualitative analysis based on a small amount of data

In this paper, we used a detailed and qualitative analysis approach rather than a quantitative perspective
on knowledge transfer in PP. This means that the very time-consuming steps (for example group
viewing) of the interaction analysis were conducted on only a small subset of the overall data. However,
all 21 sessions had been analysed previously to ensure that the excerpts present typical pair
programming episodes. Therefore, these findings, while not generalisable, do capture a good degree of
the variability faced within pair programming sessions.

7 Conclusions and Future Work

So, what does it take to be a good “expert” and to learn best as a “novice”?

We have shown that developers, without any explicit training, use strategies to transfer knowledge
between expert and novice developers that are examples of some of the teaching methods used in
cognitive apprenticeship – in particular using forms of Modeling, where the expert verbalizes their
thought process, and Coaching and Scaffolding where novice developers are supported while they take
an active part in the task. Not all of the teaching methods from cognitive apprenticeship were seen, and
two of them (Reflection and Exploration) may not be easily transferrable to the specific context of agile
software practice, where business value is typically prioritised over training needs, and the main focus is
on producing code ready for integration. However, novice Articulation, where a novice verbalizes their
own thought process, was not encouraged, yet this would enhance the novice’s learning experience. Not
only was novice Articulation not encouraged in any of the exemplars we analysed in detail, we can also
claim, due to step 1 in our procedure, that it did not feature in any of the sessions we recorded.
Although experts need to make an effort to transfer knowledge, they regard sessions in which novices ask questions as rewarding since it helps them to reflect on their own practices and thus learn from the experience. An increased awareness of working practices for knowledge transfer in PP will help developers to maximise the benefits from such sessions, and provide learning opportunities even for the expert.

This study focused on expert–novice constellations in order to highlight knowledge transfer activities, but a certain amount of knowledge transfer takes place in all PP sessions. One future direction will be to investigate which (if any) of these findings are evident in other PP sessions where knowledge transfer is not the main purpose.

**Acknowledgements**

We would like to thank all the developers and companies who made this research possible by participating in the studies.

**References**


### Table 1: Companies’ and developers’ background

<table>
<thead>
<tr>
<th>Industry</th>
<th>Company size</th>
<th>Team size</th>
<th>Programming experience</th>
<th>PP experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographic information systems</td>
<td>30-50</td>
<td>8</td>
<td>0.9-20 years</td>
<td>0-20 years</td>
</tr>
<tr>
<td>Traffic, logistic and transport</td>
<td>&lt;500</td>
<td>2 teams, 5 developers each</td>
<td>0.4-13 years</td>
<td>0-3 years</td>
</tr>
<tr>
<td>Email marketing</td>
<td>50-100</td>
<td>8</td>
<td>1.3-10 years</td>
<td>0-5 years</td>
</tr>
<tr>
<td>Estate CRM Software</td>
<td>50-100</td>
<td>10</td>
<td>1.5-12 years</td>
<td>0-3.6 years</td>
</tr>
</tbody>
</table>

### Table 2: Members of the interaction analysis group

<table>
<thead>
<tr>
<th>Research experience</th>
<th>Relevant languages</th>
<th>Relevant experience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experienced researcher</td>
<td>English</td>
<td>Empirical software engineering, qualitative research, research on agile methods</td>
</tr>
<tr>
<td>Experienced researcher</td>
<td>German (intermediate), English</td>
<td>Empirical research, qualitative research, gesture analysis, advanced programming</td>
</tr>
<tr>
<td>Experienced researcher</td>
<td>German, English</td>
<td>Empirical research on cooperative and human aspects of software engineering, interaction analysis</td>
</tr>
<tr>
<td>PhD student</td>
<td>English</td>
<td>Empirical research, qualitative research</td>
</tr>
<tr>
<td>PhD student</td>
<td>German, English</td>
<td>Empirical research, qualitative research, pair programming</td>
</tr>
</tbody>
</table>
Example 1 Nudging

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“Ok, so this is done now, so we can move on to the next bit.”</td>
</tr>
<tr>
<td>2</td>
<td>Expert:</td>
<td>“We could also test that first.”</td>
</tr>
<tr>
<td>3</td>
<td>Novice:</td>
<td>“Yes, ok.”</td>
</tr>
</tbody>
</table>

Example 2 Indirect hint: Preparing the environment

The expert is driving and explains a problem to the novice. The expert opens different Java classes and test files to illustrate his explanations. He finishes his explanations by pointing something out in a Java class. Afterwards he says with a smile on his face: “OK, so now it is time to switch driver.” While saying this, he switches from the Java class to a test class, looks for a specific location in this test class and moves the cursor there before he hands the keyboard to the novice.

Example 3 Pointing out a problem

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>The novice has just finished writing some code.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“I see at least three mistakes.”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“You see three mistakes?”</td>
</tr>
<tr>
<td>3</td>
<td>Expert:</td>
<td>“I see three mistakes.”</td>
</tr>
<tr>
<td>4</td>
<td>Novice:</td>
<td>“OK.”</td>
</tr>
<tr>
<td>5</td>
<td>Expert:</td>
<td>“Twice statement.close and one uninitialized member variable.”</td>
</tr>
<tr>
<td>6</td>
<td>Novice:</td>
<td>“Yes.”</td>
</tr>
</tbody>
</table>
This is followed by the novice suggesting how to address the problems.

**Example 4** Gradually adding information (1) and Giving clear instructions

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“We would maybe need a constructor here, wouldn’t we?”</td>
</tr>
<tr>
<td>2</td>
<td>Expert:</td>
<td>“Right. That would be good.”</td>
</tr>
<tr>
<td>3</td>
<td>No communication, no typing [3sec].</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Expert:</td>
<td>“At the top.”</td>
</tr>
</tbody>
</table>

Novice is moving the cursor to the top of the class.

| 5    | Expert: | “ALT-N.” |

Novice presses ALT-N.

**Example 5** Gradually adding information (2)

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“Ok, technically, we’d have to call a [...] job instead of calling a PreSQLStatements.”</td>
</tr>
<tr>
<td></td>
<td>No communication/no driving [3sec].</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Expert:</td>
<td>“In our case, we’d call a [...] job instead now. Right.”</td>
</tr>
<tr>
<td>3</td>
<td>Novice:</td>
<td>“Right. Ok, that means we somehow build us a class now that then just gets the job and executes it.”</td>
</tr>
<tr>
<td>4</td>
<td>Expert:</td>
<td>“Eventually, that would be the implementation. Yes.”</td>
</tr>
<tr>
<td>5</td>
<td>Novice:</td>
<td>“Good.”</td>
</tr>
<tr>
<td></td>
<td>No communication/no driving [2sec]</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Expert:</td>
<td>“We just have to tell the SQL FilterStatement somehow”</td>
</tr>
</tbody>
</table>
"Instead of using PreSQLStatement; [...] use PreSQL or PreListener or PostListener or all in one. I don't know."

"Yes. Ok, but we would want to do that then by using the Config-class."

"Right. That is our only chance to intervene there."

**Example 6** Giving direct instructions with explanations

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“Yes, so then ... we can create a nice method now.”</td>
</tr>
<tr>
<td>2</td>
<td>Expert:</td>
<td>“You better do that above because this here is an inner class.”</td>
</tr>
</tbody>
</table>

**Example 7** Explanation

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“So, and now we create some methods and call them test. BasicFilter, isn’t it?”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“Was the style back then so different that, hm [...], that err... all tests were in one method?”</td>
</tr>
</tbody>
</table>
| 3    | Expert: | “Err.. no in terms of style not but err that happened quite
Expert keeps typing for 5 sec without any comments and then briefly verbalises what he is typing

| 4 | Expert: | “PreparedStatement with S, erm

Expert stops verbalizing and keeps typing for the next 10 sec without saying anything

| 5 | Expert: | “Because one was lazy and just wrote it like this and the strict policy did not exist either.”

| 4 | Novice: | “OK.”

| 6 | Expert: | “The number of parameters for the methods did not exist like today either.

Expert keeps writing for the next 9 sec without saying anything

| 7 | Expert: | “Oh wow, they [methods] do all build upon each other.”

Expert keeps writing for the next 9 sec without saying anything

| 8 | Expert: | “ Ah that is stupid, that is no fun.”

### Example 8 Explanation by showing

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“So, now you can go over it and see whether it finds it. Have a look whether the activities exist. So, you can [...]”</td>
</tr>
</tbody>
</table>

Novice leans a little bit back, takes her hand from the mouse, turns around, looks at the expert and says:

| 2    | Novice: | “The activities?” |

Expert leans forward, takes the mouse and shows the novice how to use the debugger to check the values for the list entries. He removes his hand from the mouse, leans back again and puts his hand under the table.
### Example 9 Verbalisation

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“Now, let’s have a look. I just open the tests. This is where I had started already. I just wanted to show that to you and then you may (drive) as well.”</td>
</tr>
</tbody>
</table>

Expert opens the test and then starts browsing through the code.

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“And this (test) always has everything in it? It will get big.”</td>
</tr>
<tr>
<td>3</td>
<td>Expert:</td>
<td>“Right, yes, that will get a bit bigger. I leave it all in here for now and then we can think about whether we should create a second test class.”</td>
</tr>
<tr>
<td>4</td>
<td>Novice:</td>
<td>“And for this test, do all the other tests have to be run through before? Or could we get a specific one?”</td>
</tr>
<tr>
<td>5</td>
<td>Expert:</td>
<td>“So, I thought, we have this “before” here. Let’s go through it again. It creates one CommissionCalculation. That is here and it ... the DNS. This [experts navigates with mouse through the code] is adding a hint, this is adding the “invoices”, and this one is adding a First-WorkFlow among others. It sets the status in created, added and sets this as current WorkFlow.”</td>
</tr>
<tr>
<td>6</td>
<td>Novice:</td>
<td>“Yes, this is what it just did.”</td>
</tr>
<tr>
<td>7</td>
<td>Expert:</td>
<td>“And we also made sure it synchronises this automatically. Add the “recipients”, err [...] maybe that is not as important at the moment.”</td>
</tr>
</tbody>
</table>

### Example 10 Role switch
Expert is driving. He keeps moving the cursor to the code fragments that he explains. The novice has his hands under the table.

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“And that of course, leads to the point that not all entries are used, but [...]”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“Mhm, ok.”</td>
</tr>
<tr>
<td>3</td>
<td>Expert:</td>
<td>“Or if you call execute twice at the same Prepared-Statement [...]”</td>
</tr>
<tr>
<td>4</td>
<td>Novice:</td>
<td>“OK.”</td>
</tr>
<tr>
<td>5</td>
<td>Expert:</td>
<td>“And that of course leads to the fact that even when the fields are not used [...]”</td>
</tr>
<tr>
<td>6</td>
<td>Novice:</td>
<td>“OK.”</td>
</tr>
<tr>
<td>7</td>
<td>Expert:</td>
<td>“So and we do the same somehow when the method is close and we [...]”</td>
</tr>
<tr>
<td>8</td>
<td>Novice:</td>
<td>“Mhm.”</td>
</tr>
<tr>
<td>9</td>
<td>Expert:</td>
<td>“That is the code that we have [...]. What we want to do now - aside from getting the big learning effect - err, is of course, eh, to instead of just using SQL, to ehm, it would be easier [...]”</td>
</tr>
<tr>
<td>10</td>
<td>Novice:</td>
<td>“Mhm.”</td>
</tr>
<tr>
<td>11</td>
<td>Expert:</td>
<td>“OK? So then we can change driver now..”</td>
</tr>
</tbody>
</table>

Example 11 Role switch (continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
</table>
| 12   | Novice: | “Ok, technically, we’d have to call a [...] job instead of calling a
No communication/no driving [3sec].

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Expert:</td>
<td>“In our case, we’d call a [...] job instead now. Right.”</td>
</tr>
<tr>
<td>14</td>
<td>Novice:</td>
<td>“Right. Ok, that means we somehow build us a class now that then just gets the job and executes it.”</td>
</tr>
<tr>
<td>15</td>
<td>Expert:</td>
<td>“Eventually, that would be the implementation. Yes.”</td>
</tr>
<tr>
<td>16</td>
<td>Novice:</td>
<td>“Good.”</td>
</tr>
</tbody>
</table>

No communication/no driving [2sec]

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Expert:</td>
<td>“We just have to tell the SQLFilterStatement somehow [...] that it executes it at the right time.”</td>
</tr>
</tbody>
</table>

No communication/no driving [3sec]

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>Expert:</td>
<td>“By instead of doing PreSQLStatement [...], PreSQL or PreListener or PostListener or all in one. I don’t know.”</td>
</tr>
<tr>
<td>19</td>
<td>Novice:</td>
<td>“Yes. Ok, but we would want to do that then by using the Config-class.”</td>
</tr>
</tbody>
</table>

Novice puts his hands on the keyboard and starts typing.

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Expert:</td>
<td>“Right. That is our only chance to intervene there.”</td>
</tr>
</tbody>
</table>

**Example 12 Novice driving**

Developers just switched roles and the novice takes the keyboard and starts creating a for-loop to check entries in a list.

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Novice:</td>
<td>“So, now here we have a “subject” [...]. “From” [...]. So, I just go through [the list] now and when I find one [activity], what am I doing with it then?”</td>
</tr>
</tbody>
</table>
**Example 13** Verbalisation (continued)

<table>
<thead>
<tr>
<th>Line</th>
<th>Speaker</th>
<th>Talk</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Expert:</td>
<td>“The first checkpoint should have gotten an email already. Just because I saved it here in the status edit [...] because it is now in status created”</td>
</tr>
<tr>
<td>2</td>
<td>Novice:</td>
<td>“None.”</td>
</tr>
<tr>
<td>3</td>
<td>Expert:</td>
<td>“Created and None.”</td>
</tr>
<tr>
<td>4</td>
<td>Novice:</td>
<td>“Yes, Created and None.”</td>
</tr>
<tr>
<td>5</td>
<td>Expert:</td>
<td>“That means, if our logic is correct, what we just programmed, then we could indeed already search for an “activity”, that has the subject “from”, err, err from a work-flow that goes to the first checkpoint.”</td>
</tr>
</tbody>
</table>
| 6 | Novice: | “Ah, but I understood that at the interface, the creator, so that the calculation will be delivered to the creator. Hence, the interface, the “wizard” sets the “workflow” to “first prove”. And we are at “first
“prove”. That means it must have passed it on.

<table>
<thead>
<tr>
<th></th>
<th>Expert:</th>
<th>“You’re right. I think this is how we did it here, didn’t we? I have to have a look again.”</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Expert:</td>
<td>“So, at the moment, here I can orientate myself better.”</td>
</tr>
<tr>
<td>8</td>
<td>Novice:</td>
<td>“Yes, yes.”</td>
</tr>
<tr>
<td>10</td>
<td>Expert:</td>
<td>“You’re right. For sure. That means that nothing has happened here yet. You’re right, completely right.”</td>
</tr>
</tbody>
</table>
Figure 1: Left: Screenshot of a fully synchronized video showing Eclipse IDE and the developers. Right: Recording setup in one of the companies.
International Journal of Human - Computer Studies

AUTHOR AGREEMENT FORM

Manuscript Title: Knowledge Transfer in Pair Programming: An In-depth Analysis

List of All Authors: Laura Plonka, Helen Sharp, Janet Van der Linden, Yvonne Dittrich

Corresponding Author: Helen Sharp

This statement is to certify that that the author list is correct, all Authors have seen and approved the manuscript being submitted and agree to its submission to the International Journal of Human - Computer Studies. The Authors also confirm that this research has not been published previously and that it is not under consideration for publication elsewhere. On behalf of all Co-Authors, the Corresponding Author shall bear full responsibility for the submission.

All authors agree that the author list is correct in its content and order and that no modification to the author list can be made without the formal approval of the Editor-in-Chief. All authors accept that the Editor-in-Chief's decisions over acceptance, rejection or retraction (the latter in the event of any breach of the Principles of Ethical Publishing in the International Journal of Human - Computer Studies being discovered) are final.
Whilst knowledge transfer is one of the most widely-claimed benefits of pair programming, little is known about how knowledge transfer is achieved in this setting. This is particularly pertinent for novice-expert constellations, but knowledge transfer takes place to some degree in all constellations. We ask “what does it take to be a good “expert” and how can a “novice” best learn from a more experienced developer?” An in-depth investigation of video and audio excerpts of professional pair programming sessions using Interaction Analysis reveals six teaching strategies ranging from giving direct instructions to subtle hints, and challenges and benefits for both partners. These strategies are instantiations of some but not all teaching methods promoted in cognitive apprenticeship; novice articulation, reflection and exploration are not seen in the data. The context of pair programming influences the strategies, challenges and benefits, in particular the roles of driver and navigator and agile prioritisation which considers business value rather than educational progression. Utilising these strategies more widely and recognizing the challenges and benefits for both partners will help developers to maximise the benefits from pairing sessions.
Knowledge Transfer in Pair Programming: An In-depth Analysis

- We analyse professional pair programming sessions to investigate knowledge transfer
- We identify 6 knowledge transfer strategies
- Both experts and novices learn, face challenges, and benefit from knowledge transfer
- Driver and navigator roles influence these strategies, challenges and benefits
- Agile prioritisation influences these strategies, challenges and benefits