A Coding Scheme Development Methodology Using Grounded Theory for Qualitative Analysis of Pair Programming

Stephan Salinger, Laura Plonka, Lutz Prechelt

Institut für Informatik, Freie Universität Berlin
Joensuu, 4 July 2007
Contents

• We do talk about:
  • In brief:
    • research in Pair Programming (PP) done up to now
    • our research
    • basics of (plain) Grounded Theory (GT)
    • our data (video)
  • Problems with plain GT on rich video data
  • Four practices to avoid those problems:
    • Practice 1: Perspective on the data
    • Practice 2: Concept name syntax rules
    • Practice 3: Meta-model
    • Practice 4: Pair coding
  • Use of the practices: an example
• We do not (really) talk about:
  • Concrete results of our PP research
    • Only to demonstrate the usefulness of our practices
Pair Programming: Description

- In PP
  - two programmers jointly produce one artefact (design, algorithm, code, etc.).
  - [...] 
  - One partner is the *driver* and has control of the pencil/mouse/keyboard [...] 
  - The other person (*observer*) continuously and actively observes the work of the driver ... 
  - The roles of driver and observer are [...] switched between the pair [...] 
  - [...] 

(Williams, Kessler, and Cunningham (2000))
Pair Programming: research done up to now

- PP has been subject of many empirical investigations
- Most of them are quantitative. They regard the underlying process of PP as a kind of *black box*
  - Performance
  - Error rate
  - Programmer satisfaction etc.
- The results of this research are often contradictory
  - Most likely these differences are caused by differences in moderator variables (*type of task* etc.)
Pair Programming: our research

• Our overall goal is to understand PP in such a way that we can advice practitioners how to use it most efficiently
• The only way to obtain this understanding is *to look into the black box*
  • Understanding the mechanisms at work in the PP process
  • This understanding must first be gained in qualitative form
  • Since we do not know much yet, the investigation has to start exploratorily.
• We started with an investigation based on GT
• We used rich sets of data (audio, video, screen capture)
Basics of Grounded Theory (1)

- **GT (Glaser/Strauss 1967):**
  - Qualitative data analysis approach
  - Uses hardly any prior assumptions nor predefined terminology
  - Produces a *theory* that describes interesting relationships between *phenomena* (situations, events etc.)
    - The phenomena are represented by abstract *concepts/codes*
  - Some central aspects of GT according to Strauss (1995):
    - **Theoretical coding:** Codes are theoretical, not just descriptive; they have explanatory value for the phenomena
    - **Constant comparison:** Observed phenomena are compared many times in order to create concepts that are precise and consistent
Basics of Grounded Theory (2)

- We use a variant of GT described by Strauss/Corbin
  - It suggests three (partially parallel) activities. Two of them are:

  - **Open coding**: Describing the data (phenomena) by means of conceptual codes (derived directly from the data)
  - **Axial coding**: Identifying relationships between the concepts
    - Strauss/Corbin suggest a concrete set of relationships to be checked for. They call it a *paradigmatic model*:
      
      $\text{context} \rightarrow \text{causal condition} \rightarrow \text{phenomena} \rightarrow \text{participant's strategies} \rightarrow \text{consequence}$
Data used for our analysis of PP (1)

• Origin of the Data
  • Experiment at the end of a course on enterprise information systems and J2EE
  • Participants: seven pairs of graduate students
  • Task: Extension of an existing web shop application
    • Required broad passive J2EE knowledge about JMS, JNDI, and JBoss application server
    • Non-trivial: only three pairs were completely successful

• Data capturing procedure
  • Audio recording
  • Frontal-perspective video of the programmers
  • Full-resolution screen recording capture (all computer activities of the programmers)
Data used for our analysis of PP (2)

- All three recordings are unified into a single, fully synchronized video:
- We decided to work on video directly (transcription was not practical)
  - Too much potentially relevant information (e.g. in the screen recording)
  - We chose a QDA-Software that allows creating annotations to video.
Our initial attempt of using GT (1)

- Initial attempt: Open coding in the manner suggested by Strauss/Corbin
- Short-term goal: Characterizing the activities occurring during PP

- This approach generated 194 different concepts, but
  - almost complete confusion and
  - hardly any results

- We recognised the following problems:
  1. No predefined focus
     - No criteria for selecting which observations to code or to ignore
     - We were overwhelmed by the data
  2. No predefined granularity
     - No prior decision on the level of detail
     - We produced codes on different levels of detail, which were difficult to delineate against one another subsequently
       - E.g.: handle problem vs. test defect fix
  3. No predefined level of acceptable subjectivity
     - GT does not provide a criterion for deciding where „grounded in data“ ends and „wishful thinking“ begins
     - We mixed objective-descriptive and subjective-evaluative attitudes for selecting codes
       - E.g.: uses documentation vs. gains knowledge of detail
       - Hard to decide which one to use in a particular case
  4. Lack of concept grouping
Our initial attempt of using GT (2)

- With this understanding we stopped this mode of investigation completely
- We redesigned the coding procedure.
- The result of the redesign were a number of heuristic and intertwined practices (the heart of this presentation)
  - Practice 1: Perspective on the data
  - Practice 2: Concept name syntax rules
  - Practice 3: Meta-model
  - Practice 4: Pair coding
Practice 1: Perspective on the data

- To start the analysis, Strauss/Corbin only suggest to formulate an open and wide question
  - They provide no guidelines/no criteria
- In contrast, we suggest to formulate a so called perspective
  - This perspective can be defined by answering the following questions:
    - Q1: In which respects do you expect the data to provide insight?
    - Q2: What kinds of phenomena do the researchers allow themselves to identify in the data?
    - Q3: What type of result do you want the analysis to bring forth?
- Three reasons why a perspective used for analysis be defined before starting:
  - To avoid drowning in detail
  - To provide constancy in the criteria used for creating and assigning concepts
  - To focus attention on the most relevant aspects
Practice 1: Perspective on the data: Question 1

- **Q1**: In which respects do you expect the data to provide insight?
  - Q1 asks not what you expect to find, only in what respect you expect to find *something*.
    - Acts as an attention filter

- E.g. in the case of our investigation:
  - The data/results should help understanding what activities dominate PP and how they are relate
Practice 1: Perspective on the data: Question 2

- **Q2**: What kinds of phenomena do the researchers allow themselves to identify in the data?
  - Q2 provides the mechanism for systematically bounding the nature and amount of subjectivity during the process of coding
    - Strongest restriction: only concepts that express directly observable phenomena (behaviouristic (stimulus/response) research perspective)
    - Weaker restrictions allow:
      - concepts referring to unobservable processes (e.g. attitudes or thinking processes)
      - concepts that involve prediction (such as “helpful for reaching goal X”)
      - concepts expressing moral judgement (good, bad)
  - **E.g. in the case of our investigation:**
    - We were convinced that only the behaviourist perspective would enable us to trust our results
Practice 1: Perspective on the data: Question 3

- **Q3:** What type of result do you want the analysis to bring forth?
  - Do you want to produce a full conceptual theory?
  - Or just a conceptual structure (system of categories)?
  - Or even simply a coding scheme?
  - E.g. in our case, the goal was just to produce a coding scheme
    - We felt we knew too little about the internals of PP
Practice 2:
Concept name syntax rules (1)

• Strauss/Corbin: No guidelines for concept names
• Our experience: freely chosen concept names turned out to be highly variable and hence
  • difficult to understand
  • difficult to remember
  • difficult to compare
• As a remedy we developed a structured naming scheme
Practice 2: Concept name syntax rules (2)

code = <actor>.<description>
description = <verb>_<object>[_<criterion>]

- Examples:
  - “P1.ask_knowledge“, “P2.explain_knowledge“
  - Other investigations potentially need other schemes

- Note for GT experts: In plain GT, finding relationships involves axial coding. By practice 2, recording at least some relationships became a benefit of open coding
Practice 2: Concept name syntax rules (3)

- When working with the scheme, we observed the following benefits:
  - Concepts will be better understood right at introduction time
  - The syntax facilitates handling and overlooking a large set of concepts
  - Some relationships between concepts are implicitly recorded as well (simplifying axial coding)
  - A concept name *explicitly* represents several aspects at once (simplifying „constant comparison“)
  - It becomes easier to understand where difficulties in delineating one concept against another come from
By practicing GT, we found some of the terminology and concepts confusing:

1. GT and our QDA-Software have their own terminology

<table>
<thead>
<tr>
<th>GT</th>
<th>ATLAS.ti</th>
</tr>
</thead>
<tbody>
<tr>
<td>phenomena</td>
<td>quotations</td>
</tr>
<tr>
<td>conceptualization</td>
<td>annotation</td>
</tr>
<tr>
<td>concepts</td>
<td>concept/code</td>
</tr>
<tr>
<td>properties</td>
<td>concept/code</td>
</tr>
<tr>
<td>categories</td>
<td>families</td>
</tr>
<tr>
<td>relationships</td>
<td>relationships/relations</td>
</tr>
</tbody>
</table>

2. Some of the differences were subtle enough that we misapplied them every once in a while
   - We became confused when we tried to reconstruct what we had meant to express
Practice 3: Meta-model (2)

- These issues prompted us to formulate an explicit meta-model
  - A model of the concepts that describes the structure of analysis results
  - A model that acts as a repository of ideas for the analysis process
Practice 3: Meta-model (3)

- The meta-model formulated as UML class model:

```
conceptional world
(axial and selective coding)

organize hierarchically

ConceptClass
+Name

subsume

ConceptRelation
+Name

Concept
1

characterise

Property
+Name

real world
(open coding)

AnnotationRelation
+Name

Focus
+Name

real sorting

temporal sorting

Track
+Name

Annotation

Quotation
+begin
+end

Value

Annotation
```
Practice 4: Pair coding (1)

- Pair coding: all coding work is done by two people working together at one computer
  - Key idea: requiring a consensus of two people for all important decisions. E.g.:
    - Single out phenomena for coding
    - Decide which existing concept to use for coding or when to create a new concept
Practice 4: Pair coding (2)

- We found a number of benefits of a pair compared to a single researcher:
  - Concept definitions become more exact
  - The differentiation between similar concepts becomes more precise
    - It is less likely that a concept slips in that is on a much different level of granularity
  - Remaining concept differentiation problems will not be ignored but rather discussed
  - The perspective on the data (practice 1) is maintained more consistently
  - A larger number of relevant phenomena are detected and encoded
Short example (1)

- **Evolutionary history of four concepts of the *think aloud* ConceptClass**

<table>
<thead>
<tr>
<th>History of observed phenomena/Actions of the coders</th>
<th>New Concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>• We recognized that the driver verbalized what he was doing</td>
<td><strong>thinkaloud_activity</strong> (explains a current computer-operating activity)</td>
</tr>
<tr>
<td>• We made two decisions:</td>
<td></td>
</tr>
<tr>
<td>• Development of two ConceptClasses HCI and HHI <em>(meta-model)</em></td>
<td></td>
</tr>
<tr>
<td>• Postulation of a new concept</td>
<td></td>
</tr>
<tr>
<td>• By the concept naming syntax structure (practice 2) this concept generates a whole ConceptClass &quot;to think aloud&quot;</td>
<td></td>
</tr>
<tr>
<td>• We found a phenomenon that was obviously thinking aloud, but that did not explain computer activity</td>
<td><strong>thinkaloud_finding</strong> (states a newly won insight)</td>
</tr>
<tr>
<td>• Postulation of a new concept</td>
<td></td>
</tr>
<tr>
<td>• Discussion of the pair coder (practice 4): <em>thinkaloud_activity</em> can be used only for the driver and has priority where the other might also be applicable</td>
<td></td>
</tr>
<tr>
<td>• We encountered a programmer's explanation of the state of affairs</td>
<td><strong>thinkaloud_state</strong> (reflects on the current state of work)</td>
</tr>
<tr>
<td>• Postulation of a new concept</td>
<td></td>
</tr>
<tr>
<td>• Soon we found <strong>thinkaloud_state</strong> to exhibit two problems (pair coding):</td>
<td></td>
</tr>
<tr>
<td>1. We had a case where it collided with <strong>thinkaloud_finding</strong> (the finding concerned the state of work)</td>
<td></td>
</tr>
<tr>
<td>2. It designates statements on different levels of abstraction and granularity.</td>
<td></td>
</tr>
</tbody>
</table>
Short example (2)

History of observed phenomena/Actions of the coders

- We solved both problems by using the meta-model (practice 3):
  - Introduction of the ConceptRelation „is precondition-of“ from the existing codes
    - propose step (suggesting the next step)
    - propose_strategy (suggesting many future steps)
  - We postulated:
    - think aloud_state had to refer to a previous propose_strategy
    - A new concept thinkaloud_completion had to refer to a propose_step
- We could now discriminate large and small granularity (strategic and tactical)
- We gained a criterion for when not to use thinkaloud_finding (through demarcation to the other two)

New Concept

think aloud_completion

The example illustrates

- how open coding naturally leads into axial coding and
- how the combination of the paradigmatic model with the concept naming syntax (practice 2) can show a way back to open coding
Conclusion

- Plain GT on rich video data of PP sessions is not likely to be successful
- A set of four analysis practices provides a systematic way to hold the analysis problems at bay
Further work

Some further work:

- Validation of the coding scheme: encoding of different sessions (participants, tasks,...)
- Qualitative and quantitative evaluation of the coding process itself
- Refinement of the coding scheme with respect to particular research applications (e.g. adding properties)
- Application of the coding scheme to produce actual grounded theories of several aspects
Hypotheses based on the coding scheme

• Some Hypotheses based on the coding scheme:
  • No clues that driver and observer so indeed work on different levels of abstraction
  • We have observed what we call pair phases, characterized by a high density of communication acts referring to just one narrow issue
  • We believe that PP is not driven by strategic planning and monitoring.
  • Besides the unavoidable roles of driver and observer, PP sessions apparently tend to implicitly produce a leader role as well