

Grounded Theory Methodology (2)

Lutz Prechelt, Freie Universität Berlin

V+Ü "Empirical Methods in Software Engineering"

Part 1:

- Open Coding
 - transcription, memoing, Constant Comparison ●
- Theoretical Coding ●
- Theoretical Sensitivity

Part 2:

- Axial Coding
- Selective Coding
- Theoretical Sampling ●
- Theoretical Saturation

- Other qualitative methods

Gegenstandsverankerte Theoriebildung (2)

Lutz Prechelt, Freie Universität Berlin

V+Ü "Empirische Methoden im Software Engineering"

Teil 1:

- Offenes Kodieren
 - Transkribieren, Memos schreiben, Ständiges Vergleichen ●
- Theoretisches Kodieren ●
- Theorie-Sensibilität

Teil 2:

- Axiales Kodieren
- Selektives Kodieren
- Theoretisches Sampling ●
- Theoretische Sättigung

- Andere qualitative Methoden

- **Theoretical Sampling:**

- Data collection is always driven by the current questions/analysis
 - Do not collect lots of data without analysis

- **Open Coding:**

- Conceptualize the elements of the data
 - "fracture the data": take it apart
 - Concepts are variously called **Codes** (a label only), **Concepts** (label, definition), or **Category** (concept, properties, relationships)

- **Constant Comparison:**

- Frequently compare phenomena to codes and codes to codes to ensure grounding
 - split incoherent concepts into several
 - join too-similar concepts into one

- **Theoretical Coding:**

- Concepts should explain, not describe
- Abduction: Infer the best explanation

- **Theoretical Sensitivity:**

- **Develop a feel for what is relevant**

- **Axial Coding:**

- determine and conceptualize reliable relationships between phenomena

- **Selective Coding:**

- Pick a core concept and arrange a Grounded Theory around it
 - and then "tell the story"

- **Theoretical Saturation:**

- The GT is done if new data exhibits only known phenomena

but we are still not done with Open Coding:

- Initial Open Coding quickly leads to a wild zoo of codes
 - especially when using line-by-line or word-by-word coding
- We need to get to an orderly system of concepts
 - so we can memorize codes
 - so we can make things orthogonal
 - so we can roughly explain many concepts in a small space
- A good approach is often to form codes with a regular structure
- Example: *PP base concepts* [SalPre13]
 - Each HHI base concept name has the form *<verb>_<object>*
 - e.g. *propose_design*, *disagree_step*
 - Each object comes with several verbs, forming a concept group
 - Groups relate to prototypical dialog episode structures: Bring X up, discuss/decide X.
 - Most verbs recur in several groups
 - Verbs and objects can roughly be understood separately
 - which leads to a useful partial understanding of the compound concepts

Example: The PP base concepts

[SalPre13, p.50]; an entire book mostly about defining 60 base concepts.

These descriptions are *not* the concept memos!

E.g. there are almost 2 pages for discriminating *propose_step* from *propose_design*.

Took several years to work out.

product-oriented concepts		process-oriented concepts		
<p><i>amend_design</i></p> <p>Extend a given proposal regarding the structure and content of the program without rejecting the proposal.</p>	<p><i>ask_design</i></p> <p>Ask for a concrete proposal regarding the structure and content of the program.</p>	<p><i>amend_step</i></p> <p>Extend a given proposal regarding the next tactical work step without rejecting the proposal.</p>	<p><i>ask_step</i></p> <p>Ask for a concrete proposal regarding the next tactical work step.</p>	<p><i>explain_completion</i></p> <p>Make a statement regarding the degree of completion of the current tactical work step.</p>
<p><i>challenge_design</i></p> <p>Reject a given proposal regarding the structure and content of the program and make an alternative proposal instead.</p>	<p><i>agree_design</i></p> <p>Signal agreement with a given proposal regarding the structure and content of the program.</p>	<p><i>challenge_step</i></p> <p>Reject a given proposal regarding the next tactical work step and make an alternative proposal instead.</p>	<p><i>agree_step</i></p> <p>Signal agreement with a given proposal regarding the next tactical work step.</p>	<p><i>agree_completion</i></p> <p>Signal agreement with a statement regarding the degree of completion of the current tactical work step.</p>
<p><i>decide_design</i></p> <p>Select one from among several alternative proposals regarding the structure and content of the program.</p>	<p><i>propose_design</i></p> <p>Make one or several alternative proposals regarding the structure and content of the program.</p>	<p><i>decide_step</i></p> <p>Select one from among several alternative proposals regarding the next tactical work step.</p>	<p><i>propose_step</i></p> <p>Make one or several alternative proposals regarding the next tactical work step.</p>	<p><i>challenge_completion</i></p> <p>Reject a statement regarding the degree of completion of the current tactical work step and make an alternative statement.</p>
<p><i>disagree_design</i></p> <p>Reject a given proposal regarding the structure and content of the program without making an alternative proposal.</p>		<p><i>disagree_step</i></p> <p>Reject a given proposal regarding the next tactical work step without making an alternative proposal.</p>		<p><i>explain_state</i></p> <p>Make a statement regarding the degree to which the current strategy or work plan has been worked through.</p>
	<p><i>remember_requirement</i></p> <p>Remind the pair of a given (pre-specified) functional or non-functional requirement of the program.</p>	<p><i>amend_strategy</i></p> <p>Extend a proposed strategy or work plan without rejecting it.</p>	<p><i>ask_strategy</i></p> <p>Ask for a concrete proposal regarding the strategy or work plan to be chosen.</p>	<p><i>agree_state</i></p> <p>Signal agreement with a statement regarding the degree to which the current strategy or work plan has been worked through.</p>
<p><i>challenge_</i></p>	<p><i>agree_</i></p>	<p><i>challenge_strategy</i></p>	<p><i>agree_strategy</i></p>	<p><i>challenge_state</i></p>

Open Coding:

Applying a terminology

- Working out a good terminology can take a long time
- Even using it afterwards can be far from simple

Example:

- View the following PP scene ZB7-135 at least once
- Use the transcript below (from slide PDF)
 - or make your own
- Annotate (most of) the dialog using only the short definitions on p.50/51 of [[SalPre13](#)]
 - P&P concepts have priority over universal concepts





P3: Wir programmieren das mit ganz normalem JMS, würde ich sagen.

P4: [unverständlich]

[öffnet XpetsoreEmailSpy.java]

Nehmen wir das hier als Vorlage? [P1 nickt]
Das Programm hier?

P3: Ja.

P4: Dann mache ich da 'ne Kopie von.
Das wird dann nämlich...

[legt Hand ans Kinn, wendet sich ab,
macht Keuchgeräusch]

Nee

P3: Nee, den brauchen wir ja noch.

P4: Dochdoch, können wir machen.

P3: Müssten dann das ganze Projekt aber...

P4: Nee, wir können das einfach so kopieren.
An dem [Objektname] machen wir nix mehr.
Der ist einmal deployed, der läuft da in Ruhe...

[P3 nickt. P4 kopiert, gibt Zielname ein]

P3: Ja, Du hast recht

P4: So, den nenn ich jetzt... SpySendToTopic.

P3: Nee! Ach so, schon richtig.

P4: Das war der alte. Hier kann man mal wieder.. [schließt Editoren]

P3: Ich denk immer falsch. Aber Du denkst einfach anders. [lacht]



P3: Wir programmieren das mit ganz normalem JMS, würde ich sagen. {propose_design}
 {propose_design}
 P4: Nehmen wir das hier als Vorlage? {propose_step}
 {propose_step}
 [P3 nickt] {agree_step}
 Das Programm hier? {propose_step}
 P3: Ja. {agree_step}
 P4: Dann mache ich da 'ne Kopie von. {thinkaloud_activity}
 {thinkaloud_activity}
 Das wird dann nämlich... {explain_knowledge}
 Nee {disagree_step}
 P3: Nee, {stop_activity, disagree_step} den brauchen wir ja noch. {explain_knowledge}
 {explain_knowledge}
 P4: Doch, können wir machen. {agree_step}
 P3: Müssten dann das ganze Projekt aber... {mend_step*}

P4: Nee, wir können das einfach so kopieren. {agree_step}
 {agree_step}
 An dem JBoss[Objektname] machen wir nix mehr. Der ist einmal deployed, der läuft da in Ruhe... {explain_knowledge}
 {explain_knowledge}
 P3: Ja, Du hast recht {agree_step}
 {agree_step}
 P4: So, den nenn ich jetzt... SpySendToTopic. {propose_design}
 {propose_design}
 P3: Nee! {disagree_design}
 {disagree_design}
 Ach so, schon richtig. {agree_design}
 {agree_design}
 P4: Das war der alte. {agree_design**}
 {agree_design**}
 Hier kann man mal... {thinkaloud_activity}
 {thinkaloud_activity}
 P3: Ich denk immer falsch. Aber Du denkst einfach anders. {explain_finding}
 {explain_finding}

- Understand your coding differences.

*or: propose_strategy, explain_finding, propose_hypothesis

**or: explain_knowledge

1. Context understanding:

- a) We have not seen what happened before
 - this is after 2:15 hours
 - so the pair shares a lot of context we do not know
- b) Our technology knowledge is too low
 - system built with Java2 Enterprise Edition
- c) Both partners think fast
 - they not verbalize many things and still understand each other immediately
 - even when P3 does not yet understand the technical situation.
 - (This is a sign of good *Togetherness*, see below)

2. Base concepts understanding:

- a) The base concepts were developed by somebody else
- b) We have no practice in applying them
- c) They are more subtle than one might think

Problem 2 disappears over time, problem 1 is fundamental and stays



- Due to grounding, GTM work usually starts with "low-level" concepts
 - and later works towards also "higher-level" concepts
- What does "high-level" mean?
 - 1. Less local:**
Instances stretch over more data
(annotated text is longer)
 - or:
 - 2. More general:**
Concept covers a larger class of instances

PP example of "more general":

- We might initially have created *suggest_variablename*, *suggest_classname*, *recommend_designpattern*
- and then decide this is too fine-grained
→
- introduce *propose_design* for all of those
 - and no longer use the more specialized codes

fictitious

PP example of "less local":

- We focused on knowledge creation and transfer
- Creation & transfer happen in episodes
 - some goal (**knowledge need**) is pursued, then reached (or not)
- Episodes are driven in different ways, each leading to a **knowledge transfer mode** (property of an episode)
 - a) pull**: transfer driven by questions
 - b) push**: transfer started by knowledge-possessor

- c) co-produce**: partners produce the knowledge collaboratively
 - lots of dialog, very variable
- d) pioneering**: One partner produces the knowledge alone (mostly by reading)
 - can be **silent** or **talking**

see [ZiePre14,Zieris20]

- Good example of less local, higher-level concepts
 - found only later during Open Coding
 - (but could have been found early with a differently oriented Theoretical Sensitivity)

- Note that being "higher-level" does not imply usefulness
 - we need Theoretical Sensitivity to avoid developing concepts that lead nowhere

PP example:

- The first higher-level concept Franz Zieris found was the *Clarification Cascade*
 - a rare particular ordering of question types in a pull episode where the questions are misunderstood
 - academically perhaps interesting, but
 - of no value for practitioners and
 - irrelevant for our subsequent theory development
 - see [ZiePre14,Zieris20]

- **Theoretical Sampling:**

- Data collection is always driven by the current questions/analysis
 - Do not collect lots of data without analysis

- **Open Coding:**

- Conceptualize the elements of the data
 - "fracture the data": take it apart
 - Concepts are variously called **Codes** (a label only), **Concepts** (label, definition), or **Category** (concept, properties, relationships)

- **Constant Comparison:**

- Frequently compare phenomena to codes and codes to codes to ensure grounding
 - split incoherent concepts into several
 - join too-similar concepts into one

- **Theoretical Coding:**

- Concepts should explain, not describe
- Abduction: Infer the best explanation

- **Theoretical Sensitivity:**

- **Develop a feel for what is relevant**

- **Axial Coding:**

- determine and conceptualize reliable relationships between phenomena

- **Selective Coding:**

- Pick a core concept and arrange a Grounded Theory around it
 - and then "tell the story"

- **Theoretical Saturation:**

- The GT is done if new data exhibits only known phenomena

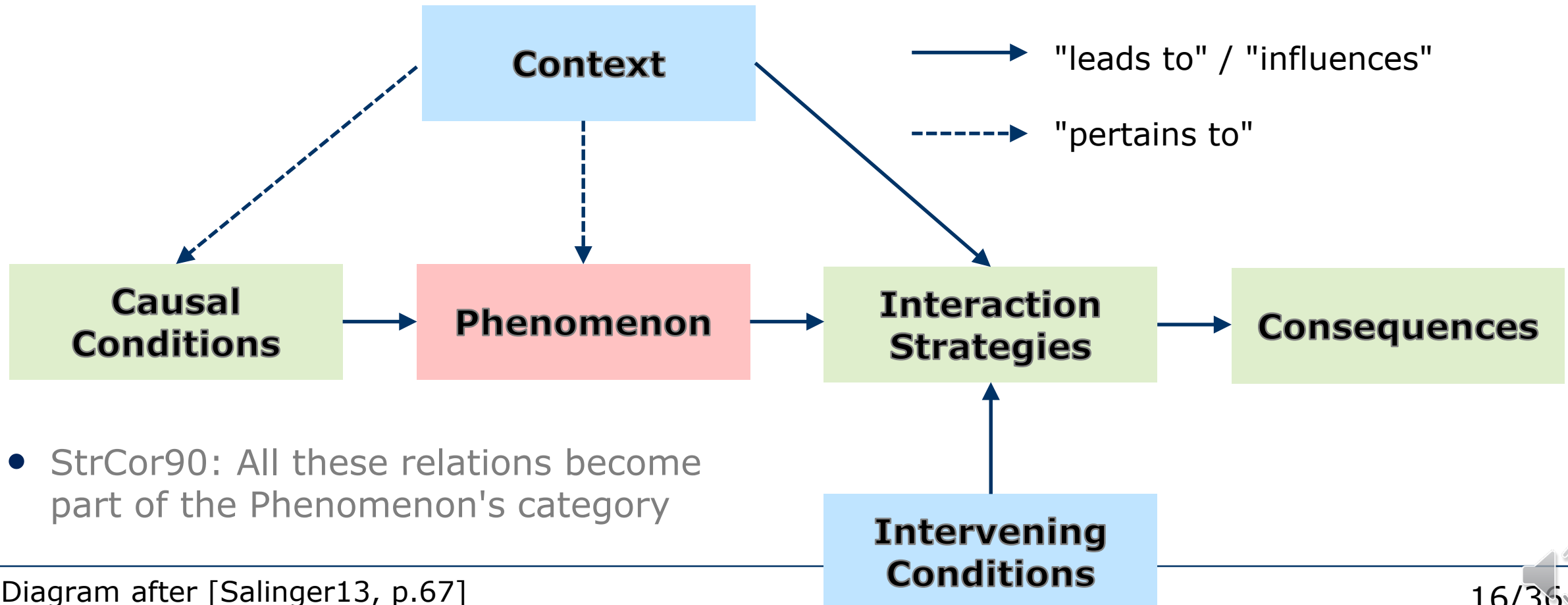


Axial Coding: Putting the data back together again

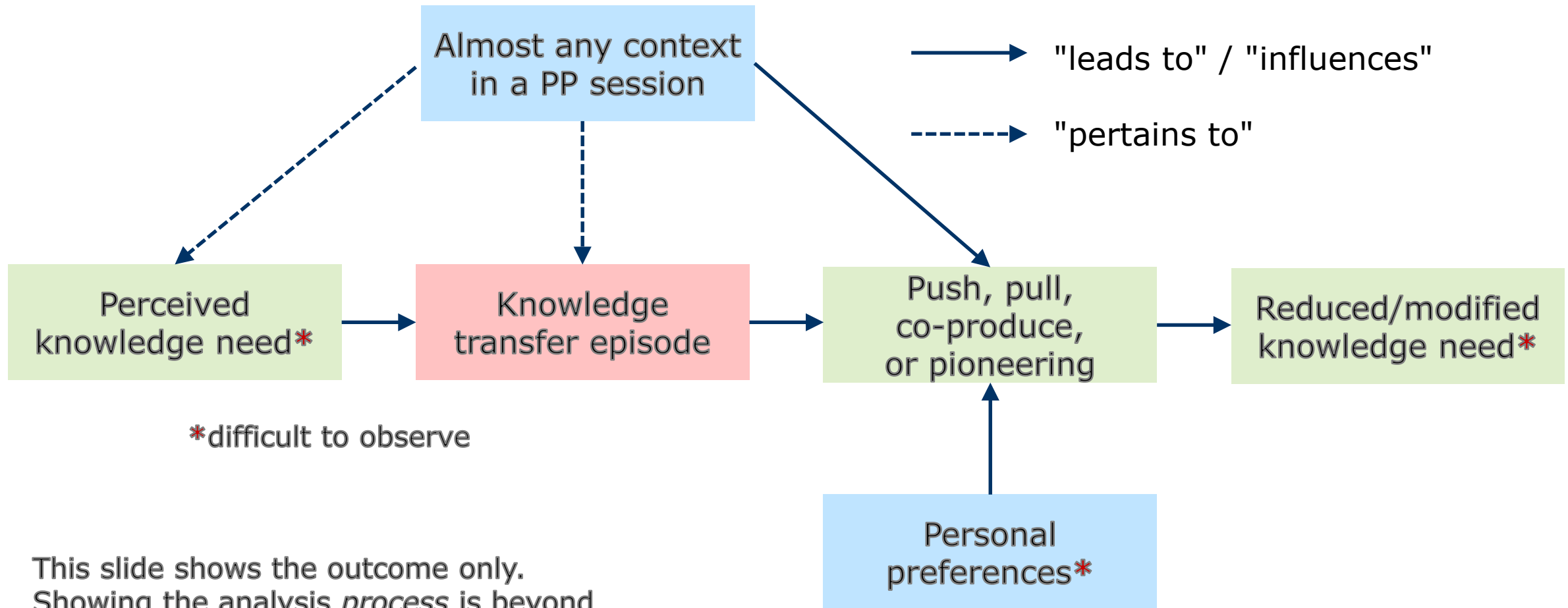
- Initially in GTM, the data is a tangled mass of weird stuff
 - Remember the first time you saw the first PP scene ZA-280?
- Open Coding cuts the data into understandable pieces
 - *"Open coding [...] fractures the data"* [StrCor90, p.97] (dt. "aufbrechen")
- *"Axial coding puts those data back together in new ways by making connections"* [StrCor90, p.97]
 - Alternate smoothly between both!
- Axial Coding is used multiple times
- Each execution focuses on one well-developed concept and extends it
 - Well-developed means:
 - has precise definition (concept memo)
 - perhaps has variants (subclasses)
 - has enough instances
 - its relevant properties are also conceptualized
 - Such concepts are called "categories" [StrCor90, p.61]
 - (not a good name!)
- Axial Coding calls the focus concept the *Phenomenon*
 - StrCor90 do not cleanly distinguish class (concept) from instance (which is what "phenomenon" *should* stand for)

Enhancing **Theoretical Sensitivity** (7): The Paradigm model (dt. "paradigmatisches Modell")

- Pick any concept/Phenomenon to focus on
- Then look in the data for the following relationships and related items:



Axial Coding example: Modes of knowledge transfer in PP



This slide shows the outcome only.
Showing the analysis *process* is beyond our scope.

Axial Coding: Easy or difficult?

In interview data:

- Interviewees may point out and explain relevant relationships directly
 - this happened often in the Quality Experience study
- But beware of lack of grounding: These are opinions or claims, not facts!
 1. Need to validate any statement of fact by triangulation:
 - asking others about the same thing,
 - and cross-check
 2. Need to validate any generalization by
 - asking for specific instances and episodes
 - Watch for red flags! (see above)

With field observations:

- It can take long to obtain enough data about infrequent phenomena
 - Information about Context or Conditions is often incomplete or unreliable
 - There may be combinatorial explosion of possible factors
- ➔ Paradigmatic models can rarely be filled completely at the concept level
- and will talk only about *some* conditions *some* strategies & *possible* consequences

- **Theoretical Sampling:**

- Data collection is always driven by the current questions/analysis
 - Do not collect lots of data without analysis

- **Open Coding:**

- Conceptualize the elements of the data
 - "fracture the data": take it apart
 - Concepts are variously called **Codes** (a label only), **Concepts** (label, definition), or **Category** (concept, properties, relationships)

- **Constant Comparison:**

- Frequently compare phenomena to codes and codes to codes to ensure grounding
 - split incoherent concepts into several
 - join too-similar concepts into one

- **Theoretical Coding:**

- Concepts should explain, not describe
- Abduction: Infer the best explanation

- **Theoretical Sensitivity:**

- **Develop a feel for what is relevant**

- **Axial Coding:**

- determine and conceptualize reliable relationships between phenomena

- **Selective Coding:**

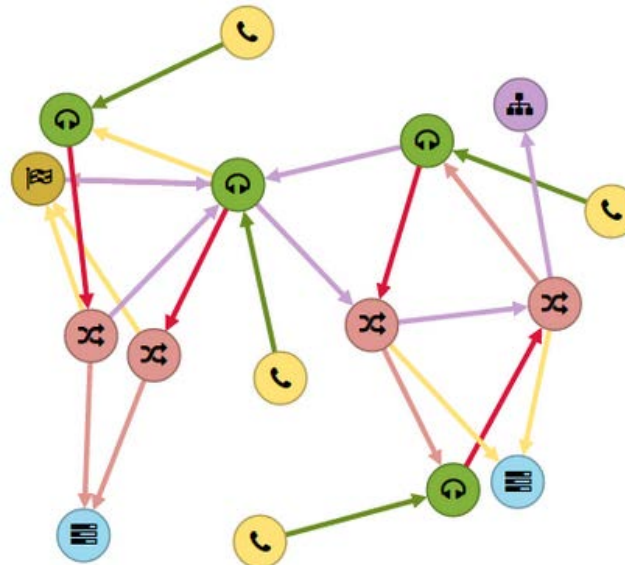
- Pick a core concept and arrange a Grounded Theory around it
 - and then "tell the story"

- **Theoretical Saturation:**

- The GT is done if new data exhibits only known phenomena



- [StrCor90, p.116]
"Selective Coding: The process of
 - selecting the core category,
 - systematically relating it to other categories,
 - validating those relationships, and
 - filling in categories that need further refinement and development."
 - Which means to go back to Axial Coding and perhaps Open Coding.
- If the material is rich, multiple rounds of Selective Coding can be useful
 - Each leads to a separate theory
 - These theories will be densely interconnected



Selective Coding can be easy:

- In the Quality Experience study, the core concept was found and recognized early
 - so little analysis was spent on things not important for the resulting theory

Selective Coding can be difficult:

- In PP research, it took long before we recognized concepts of interest to practitioners.
- So far we have (see [Zieris20]):
 1. A theory of overall session dynamics
 - Fine.
 2. Some elements of PP skill
 - Probably incomplete
 3. The notions of Fluency and Togetherness for pair self-observation and reflection
 - Helpful, but only concepts, not a full theory

Franz Zieris, Lutz Prechelt:
"[Two Elements of Pair Programming Skill](#)",
ICSE NIER 2021

- We had long *felt* there was such a thing as PP skill
 - distinct from programming skill.
 - But what *was* this skill specifically?
- While investigating knowledge transfer modes, we had seen episodes of
 - a) pairs losing track of what they were trying to understand (sub-sub-sub-episodes)
 - b) developers drowning their partner in too much detail (explain too much)
 - c) developers losing their partner in too much Silent Pioneering (explain too little)

- Our best answer to the question what is "good" PP was:
 - Good pairs carefully maintain *Togetherness*

Decision:

Make "PP Skill" the core category →

- One element of PP Skill is Maintaining Togetherness
 - and we found typical patterns of doing so
 - there may be more
- Another is Maintaining Expediency, i.e., avoiding process inefficiencies
 - such as the patterns a), b), c)
 - there may be more
- There may be more elements

- **Theoretical Sampling:**

- Data collection is always driven by the current questions/analysis
 - Do not collect lots of data without analysis

- **Open Coding:**

- Conceptualize the elements of the data
 - "fracture the data": take it apart
 - Concepts are variously called **Codes** (a label only), **Concepts** (label, definition), or **Category** (concept, properties, relationships)

- **Constant Comparison:**

- Frequently compare phenomena to codes and codes to codes to ensure grounding
 - split incoherent concepts into several
 - join too-similar concepts into one

- **Theoretical Coding:**

- Concepts should explain, not describe
- Abduction: Infer the best explanation

- **Theoretical Sensitivity:**

- **Develop a feel for what is relevant**

- **Axial Coding:**

- determine and conceptualize reliable relationships between phenomena

- **Selective Coding:**

- Pick a core concept and arrange a Grounded Theory around it
 - and then "tell the story"

- **Theoretical Saturation:**

- The GT is done if new data exhibits only known phenomena



- At any point during Open Coding, Axial Coding, or Selective Coding, we may ask: *"What if <X>?"*
 - and then find we have no data about this
- This triggers a GTM researcher to get up and collect data expected to have <X>
 - This is Theoretical Sampling:
The theory development suggests what data to look at.
 - Depending on research interest, Theoretical Sampling can be easy or difficult

Example:

Investigating PP Skill:

- We considered all our pairs to have (more or less) medium levels of skill
- We wondered what high PP skill might look like →
- We found a PP-only company ("We use PP for everything") and made some recordings there:
 - No higher-PP-skill pair found
 - But a too-low-PP-skill pair
 - → session state PPbreakdown
 - helped understand PP Skill as well

- Our overview slide:
Theoretical Saturation:
 - The GT is done if new data exhibits only known phenomena
- StrCor90, p.188: "sample until [...]"
 - 1) no new or relevant data seem to emerge regarding a category;
 - 2) the category development is dense, insofar as all of the paradigm elements are accounted for, along with variation and process;
 - 3) the relationships between categories are well established and validated."

But:

- 1) What does it mean that "no new data seem to emerge"?
How much additional data should I collect (and not see anything new) before I stop?
 - 2) Accounting for all variations is very often not feasible
- A questionable concept!

Theoretical Saturation is rarely achieved

• **Theoretical Sampling:** ✓

- Data collection is always driven by the current questions/analysis
 - Do not collect lots of data without analysis

• **Open Coding:** ✓✓

- Conceptualize the elements of the data
 - "fracture the data": take it apart
 - Concepts are variously called **Codes** (a label only), **Concepts** (label, definition), or **Category** (concept, properties, relationships)

• **Constant Comparison:**

- Frequently compare phenomena to codes and codes to codes to ensure grounding
 - split incoherent concepts into several
 - join too-similar concepts into one

• **Theoretical Coding:** ✓

- Concepts should explain, not describe
- Abduction: Infer the best explanation

• **Theoretical Sensitivity:** ✓✓

- **Develop a feel for what is relevant**

• **Axial Coding:** ✓

- determine and conceptualize reliable relationships between phenomena

• **Selective Coding:** ✓

- Pick a core concept and arrange a Grounded Theory around it
 - and then "tell the story"

• **Theoretical Saturation:** ✓

- The GT is done if new data exhibits only known phenomena



The resulting theory: Telling the story

GTs must be presented as a narrative

1. An understandable story:

- Finding a linear order of explanation
 - There are many possibilities
 - Avoid forward references
 - Avoid backward references to unimportant concepts
- Limiting the number of concepts involved
 - Perhaps this should be two theories or three?
- Providing an overview diagram

2. Study quality:

- Show grounding
 - Verbal quotes are great!
- Explain relevance!
 - e.g. member check quotes (see below)

3. Explain the research process:

- Early generations of codes
- Difficulties encountered
- Key insights

4. Explain limitations:

- Uncertain observations or inferences

5. Explain presumed domain of applicability

- If your findings are only existence proofs, applicability (if not relevance) is universal
- If you formulate rules, this gets a lot trickier

Validation: Member check

If GTM is done right, GTM outputs will never be "wrong"

- But they may be useless or incomprehensible
- And we *may* have done GTM wrong!

To enhance Credibility, perform a *member check*:

- Present your outcomes to members of your domain
 - ideally including people from which you collected data and others
- Ask them if they agree
 - whether the outcomes "resonate" with their experience and perceptions
- Ask them if they find the outcomes useful

Don't attempt GTM with only a text editor or PDF tool

- You will need e.g. the following:

Create

- annotating directly in different types of document (Text, PDF, audio, video, image)
- sync transcripts to audios/videos
- recording relationships between concepts
- recording relationships between annotations
- making overlapping annotations

Review

- finding all instances of a concept and reviewing them together
- arranging concepts in a foldable hierarchy
- creating tailored views of just the right parts of your data

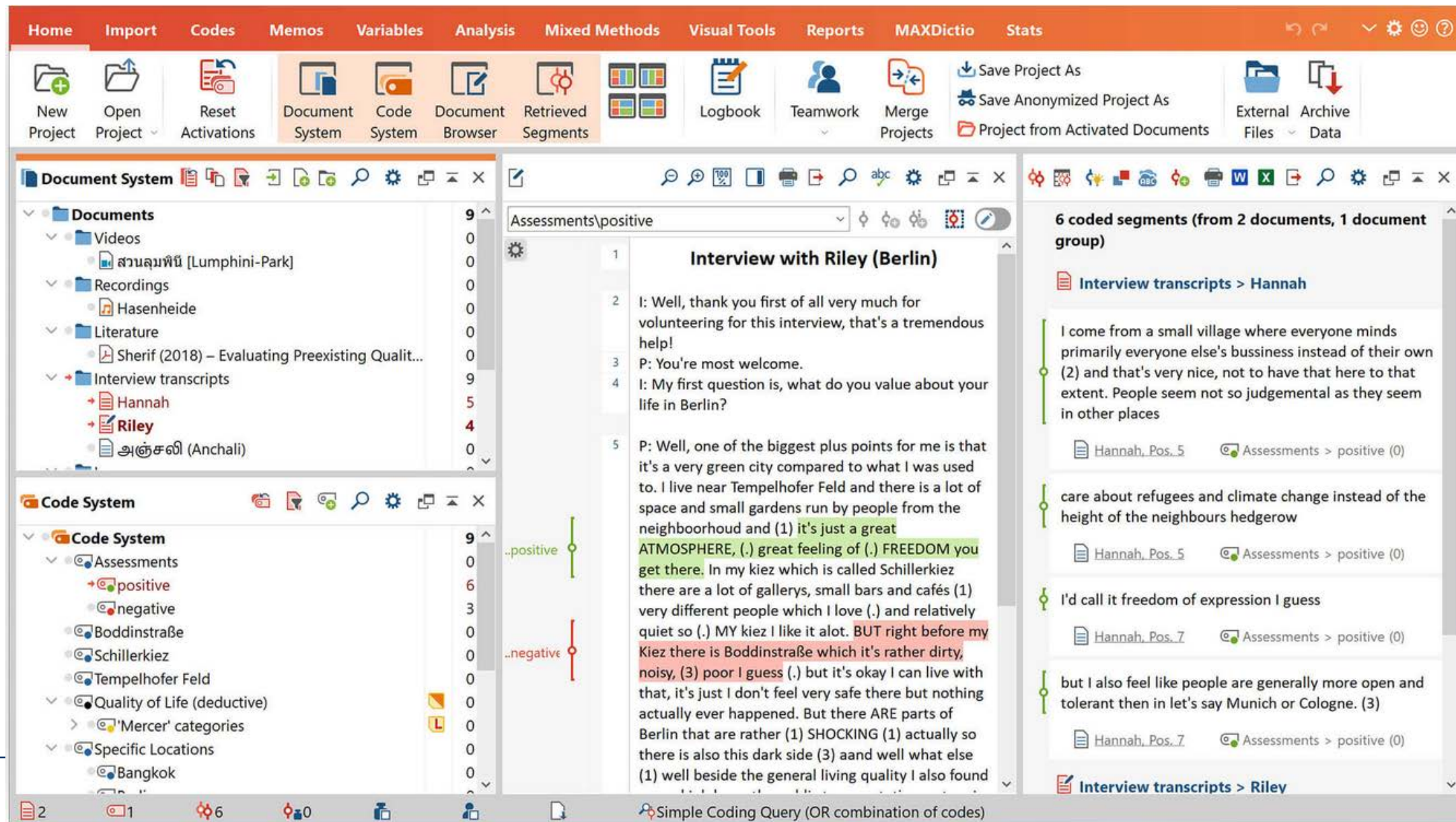
Revise

- renaming a concept reliably
- using color and re-decide which ones
- keeping previous generations of concepts around
- etc.

You need a specialized tool!

- Some leading tools:
 - [MAXQDA](#) (from Berlin, [FU campus license](#))
 - [ATLAS.ti](#) (from Berlin)
 - [NVivo](#)

MAXQDA screenshot



The screenshot displays the MAXQDA software interface with the following components:

- Top Menu Bar:** Home, Import, Codes, Memos, Variables, Analysis, Mixed Methods, Visual Tools, Reports, MAXDictio, Stats.
- Toolbar:** New Project, Open Project, Reset Activations, Document System, Code System, Document Browser, Retrieved Segments, Logbook, Teamwork, Merge Projects, Save Project As, Save Anonymized Project As, Project from Activated Documents, External Files, Archive Data.
- Document System Panel:** Shows a tree view of documents including folders like Videos, Recordings, Literature, and Interview transcripts. Under Interview transcripts, there are sub-items for Hannah (5 documents) and Riley (4 documents).
- Code System Panel:** Shows a tree view of codes. Under Assessments, there are codes for positive (6) and negative (3). Other codes include Boddinstraße, Schillerkiez, Tempelhofer Feld, Quality of Life (deductive), 'Mercer' categories, and Specific Locations (Bangkok).
- Main Text Area:** Displays an interview transcript titled "Interview with Riley (Berlin)". The text includes dialogue between an interviewer (I) and a participant (P). Several segments of text are highlighted with green and red markers, corresponding to the codes in the Code System panel. For example, "it's just a great ATMOSPHERE, (...) great feeling of (...) FREEDOM you get there" is highlighted in green, and "BUT right before my Kiez there is Boddinstraße which it's rather dirty, noisy, (3) poor I guess" is highlighted in red.
- Right Panel:** Shows a list of coded segments. It indicates "6 coded segments (from 2 documents, 1 document group)" and lists segments such as "Interview transcripts > Hannah" and "Interview transcripts > Riley". Each segment is accompanied by a snippet of text and its source (e.g., Hannah_Pos.5, Assessments > positive (0)).
- Bottom Status Bar:** Shows the number of documents (2), codes (1), and segments (6) and provides a "Simple Coding Query (OR combination of codes)" option.

Epilogue: Some other qualitative methods

Ethnography
Thematic Analysis
Content Analysis

(a few key properties of each)



- Aims at description-leading-to-explanation (answering why-questions)
- Uses "the members' point-of-view": perspective of the culture being observed
 - observes "ordinary detail of life" and reports in thick descriptions
 - initially details-oriented (descriptive) where GTM is already explanation-oriented (theoretical coding)
 - Requires field observation, often participant observation
- Employs existing theoretical lenses
- Aims at neutrality, avoids judgment
 - not an obvious choice for engineering research's wish to optimize
 - but useful for e.g. requirements elicitation when designing systems

Thematic analysis (TA)

- Aims at identifying frequent "themes" in material
 - typically interviews or documents called a *corpus* of *items* from which *extracts* are made to identify *themes*
- Essentially a particular form of iterative Open Coding
 - with its own specialized instructions and terminology:
Themes are concepts, *extracts* are annotated stretches, *items* have no name in GTM.
 - Can be applied from constructionist/interpretivist as well as realist/essentialist epistemological stances.
 - May use a theoretical lens (*theoretical* TA) or not (*inductive*, "grounded" TA)
 - Themes can be *semantic* (low-level) or *latent* (high-level)

- Aims at quantifying the frequency of qualitative concepts in material
 - A qualitative quantitative method: qualitative input data, quantitative results
 - Positivist epistemological stance!

Procedure:

- Take (or perhaps derive via Open Coding) a set of concepts
 - Each concept must have a precise definition
- Go through the material and identify each instance of each concept
- Report frequencies
- Judge each item at least twice, independently, and report inter-rater agreement

e.g. Maalej, Robillard: "[Patterns of Knowledge in API Reference Documentation](#)", TSE 2013

- [Charmaz14] Kathy Charmaz: "[Constructing grounded theory](#)", Sage 2014.
- [Salinger13] Stephan Salinger: "[Ein Rahmenwerk für die qualitative Analyse der Paarprogrammierung](#)", Dissertation FU Berlin, 2013
- [SalPre13] Stephan Salinger, Lutz Prechelt: "[Understanding Pair Programming: The Base Layer](#)", BoD 2013
- [StrCor90] Anselm Strauss, Juliet Corbin: "Basics of Qualitative Research: Grounded Theory Procedures and Techniques", 1st ed., Sage 1990
- [ZiePre14] Franz Zieris, Lutz Prechelt: "[On Knowledge Transfer Skill in Pair Programming](#)", ESEM 2014
- [Zieris20] Franz Zieris: "[Qualitative Analysis of Knowledge Transfer in Pair Programming](#)", Dissertation FU Berlin 2020

Thank you!