Part 1 is a course, not just a lecture

Grounded Theory Methodology (1)

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 (1)

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

What is Grounded Theory Methodology (GTM)?
What is a Grounded Theory (GT)?

- GTM:
  - dt. "gegenstandsverankerte Theoriebildung"
  - A qualitative research method
  - Aim: Explain a phenomenon of interest at a conceptual level
    - as opposed to: describe, categorize, count
  - Often exploratory
  - Often starts from a general research interest, not a specific research question

- GT:
  - The output of a full GTM study
    - partial applications of GTM are common and do not create GTs
  - A story that uses abstract concepts to explain what elements and aspects the phenomenon has and how they interact

- "grounded":
  - Each concept and each statement of the theory can be traced back to specific observations
    - No extrapolation, very limited interpolation.
    - If the theory is complete, any observation can also be traced forwards to some theoretical statement.

What does a Grounded Theory (GT) look like? Example: "Quality Experience"

- Lutz Prechelt, Holger Schmeisky, Franz Zieris: "Quality Experience: A Grounded Theory of Successful Agile Projects Without Dedicated Testers", Int'l. Conf. on SW Engineering 2016

- Research question: Why are there successful SW development teams both with and without testers?
  - Suggested by Holger Schmeisky, mostly done as a Master thesis

https://www.schmeisky.com/static/pic.png

- Data: field observations and interviews in 3 agile teams
  - that each build a part of a web portal
  - A1 with testers (company A)
  - A2 without testers (company A)
  - B1 without testers (company B)

- The GT is summarized in the following diagram:
  - (The diagram is not the GT!)

What does a GT look like? Example: "Quality Experience"

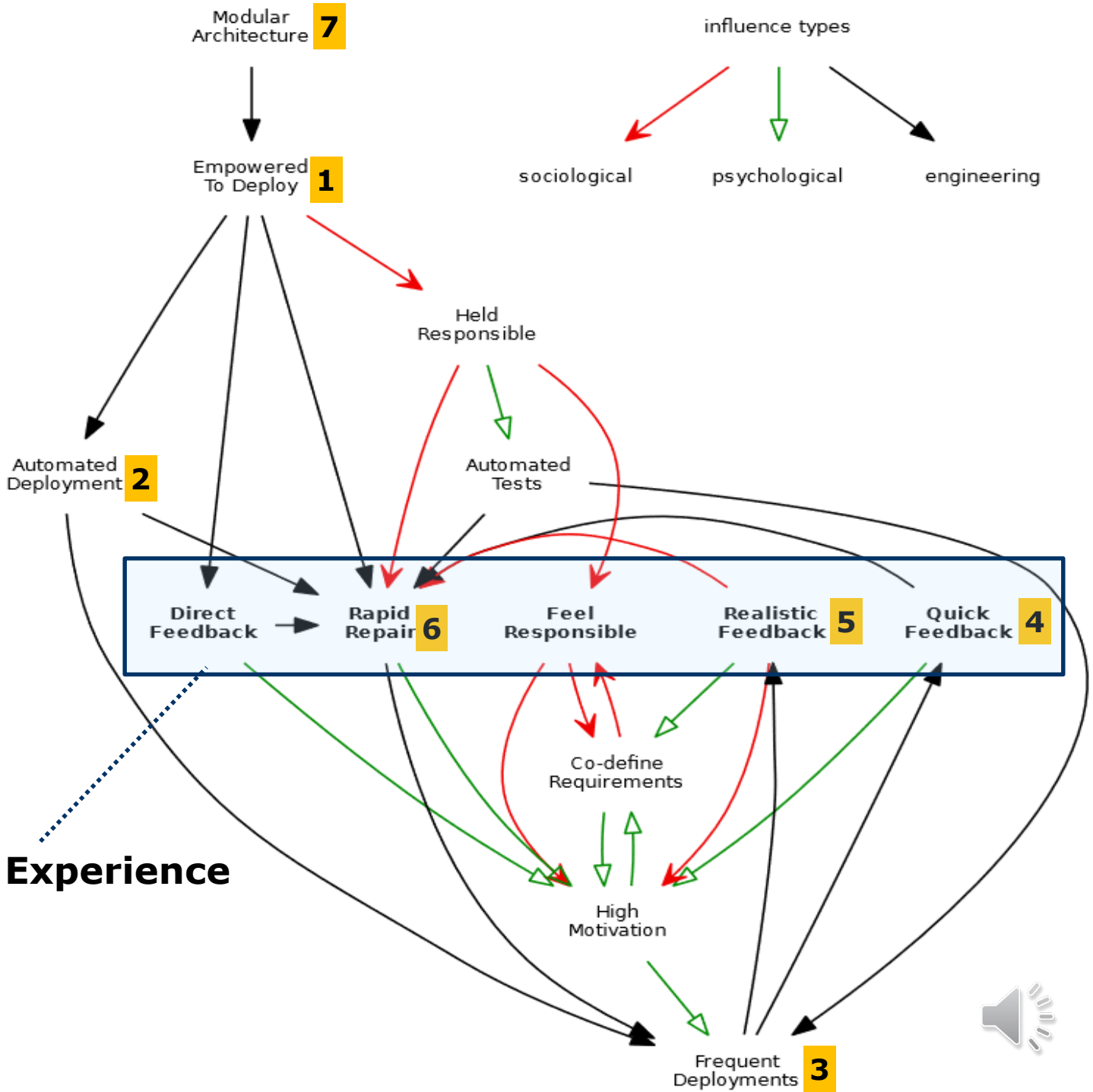
Starting point:

- Team gets empowered to deploy versions **1**

Outcomes:

- Fast and relevant feedback allows teams to repair problems so fast that dedicated testers are not needed
- (Also, teams become more careful and produce fewer defects)

(some elements and relationships are not shown)



Quality Experience

an influence/consequence diagram

How to do GTM:
Types of data for GTM

- Interviews:
  - very convenient and efficient
  - but not fully reliable:
    - memory gaps and distortions
    - post-hoc rationalizations
- Documents:
  - e.g. version archives, bug tracker, email and chat traffic archives, etc.
  - Also convenient and efficient
  - but no way to close any gaps
    - → resort to interviews or direct observations
- Direct observations:
  - often harder to obtain
  - much harder to analyze
    - see below
  - often much less condensed
    - → more data needed
  - but reflect reality
    - much fewer gaps and distortions
    - extreme case: participant observation
      - required (for also other reasons) by Ethnography

3 schools of GTM: Glaser, Strauss/Corbin, Charmaz

GTM was originally described in 1967 by Barney Glaser and Anselm Strauss

- Glaser school:
  - Emphasizes creativity and freedom
    - Glaser is a crypto-positivist:
      He talks as if GTM was an objective procedure
      - but GTM is of course deeply interpretivist
  - *"All is data"*
  - "Beware of forcing"

- Strauss/Corbin school:
  - Emphasizes systematic and teachable procedure **[→we use this the most]**
  - Axial Coding, paradigm model

- Charmaz school:
  - Emphasizes constructivism:
    - an epistemological stance, saying:
  - All conceptualization will reflect the researcher's background and perception
    - so beware of your biases!

- All three emphasize a strong need for *Theoretical Sensitivity*

GTM overview:
Key **activities** and ***notions***

Freie Universität Berlin

- **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

● : unique core ideas of GTM according to Strauss

Theoretical Sensitivity

- GTM requires a "gut feeling" for what is relevant in the data
  - Discriminate wheat from chaff
  - "What concept provides explanation rather than only description?"
  - "What concepts help satisfy my research interest?"
    - rather than conceptualizing arbitrarily
  - This power of judgment is called Theoretical Sensitivity

- Example: Quality Experience

  An easy GTM study:
  - Schmeisky had a good idea what he was looking for
    - QA-related activities and rationale
    - Could come up with relevant high-level concepts from the start
  - A member of team A1 (with testers!) mentioned "Quality Experience"; Schmeisky recognized this could be a key concept
    - So he searched for its elements
      - very early Selective Coding
    - It later turned out to be the key concept.

- Other cases are much tougher:

Open Coding (Charmaz: **Initial Coding**), **Theoretical Sensitivity**: Pair programming example

Freie Universität Berlin

- AG SE studies Pair Programming (PP) since 2004
  - Stephan Salinger, Franz Zieris, and several others
  - *"Pair programming is a dialog between two people simultaneously programming (and analyzing and designing and testing) and trying to program better."* -- Kent Beck
- Research interests:
  - **How does PP work?**
    - basic research
  - **How to do it well?**
    - engineering perspective: produce advice

- We have collected 60+ recordings of real industrial PP sessions
  - mostly in advance
  - some Theoretical Sampling much later
- We will now look at such PP data
  1. to understand Open Coding
  2. to get an idea of Theoretical Sensitivity
  3. to get an idea of field observation data
- For legal reasons, we will use recordings of *student* pairs
  - for our purposes here, they are equivalent
    - for some parts of PP research, they are not

Hold on!

Scene ZA4-280

Scene ZA4-280: Background

- The pair is building a small algorithmic program for encoding phone numbers as word sequences
- The program is almost complete
  - after 4.5 hours of work
- Now they optimize for speed by adding caching
  - Episode topic:
    What is the thing we found in the cache?
- View the scene again
  - understand the content
- Based on the scene,
  suggest 5-or-so concepts for (eventually) understanding PP
  - apply Open Coding
  - each concept must be grounded in the data.

Scene ZA4-280: Discussion

- The video has many aspects that one can pay attention to, e.g.:
  1. Code (as written down)
  2. IDE handling
  3. Software dialog
     - What the SW is, what it could become
  4. Coordination dialog
     - How to work together: Situation, what to do
  5. Social dialog
     - Getting by with each other as people
  6. Manner of speaking:
     - Voice (e.g. loudness, pitch, emphasis, prosody)
     - Body language (e.g. facial expressions, gestures, posture)

- View the scene 3 more times
  - each time focusing on only one of the aspects on the left
    - Those 3 you expect to be most interesting
    - What concepts do you find?
    - Is that aspect fruitful? Or uninteresting?
    - Will that be similar in other PP scenes?.

- Your Theoretical Sensitivity should now have improved
  - You have some ideas regarding what is important and what is less so

Memoing

During GTM work, one frequently writes or revises memos

- to clarify thoughts, to keep ideas or results

Example (fictitious):

"Most information is in the dialog.
Code or physical interactions are usually only needed for disambiguation.
Exceptions occur(?).

Partition dialog by utterance? Probably.

How to classify utterances? Unclear."

Common types of memo:

1. Code memo: Concept definition
   - a must-have for any non-trivial concept
2. Theory memo:
   Preliminary thoughts about the theory
3. Done/TODO memo:
   Remember work status
4. Do/Don't: Work heuristics
   - e.g. rules for concept names
   - e.g. what to pay attention to
   - e.g. traps to avoid
5. (perhaps personal types)

Transcription

- If all relevant content can well be expressed in writing, one can **transcribe** audio/video material
  - Write down all that happens or is said
  - Makes subsequent work much easier
  - But beware: transcribing is a lot of work
  - Advice: Do it only where needed
    - use different levels of paraphrasing to make transcripts much shorter and only as detailed as required
- View ZA4-280 again
- Transcribe it at a just-detailed-enough level of precision.

Possible transcription of ZA4-280

P2: So, jetzt interessieren wir uns also, ob wir das schon drin haben.
Und zwar unsere aktuelle Position. Also:

[schreibt dabei Code:
if (cache.containsKey(new Integer(pos)))]

Uns interessiert...ob es...unsere Position in dem Cache gibt.
Wenn ja... Jetzt wird's komplizierter.

P1: Drin müsste eigentlich alles gespeichert werden.

P2: Was liegt, was legen wir denn da rein?

P1: Das *result*.

P2: Ein *result* ist ja zu wenig.

P1: Nee, das müssten mehrere sein.
Also *results*. Sorry.

P2: Ja, aber das ist auch...nicht so einfach.

[Fährt mit dem Cursor umher.
Lehnt sich zurück.]

Na, *results* ist ja das Oberteil,

[P1 steht auf und dreht sich um.]

das Gesamtding. Das kannste nicht unten irgendwo mitspeichern.
Und in den Cache[...]

P1: [unverständlich] , was da passiert ist, dass Du ein *result* hast für Teilnummern.

P2: Ja, das müssen wir aber schreiben.
Unser *results* tut das nicht, momentan.

- Where is your transcription not adequate? Why not?

Recap: **Theoretical Sensitivity**, **Memoing**

1. One may have good Theoretical Sensitivity from the start
   - e.g. in the Quality Experience study
2. If it is harder to get, the first step is classification:
   - *"What kinds of things do we have here?"*
3. Transcription can be helpful (e.g. to create a classification, see below)
   - But preferably compactified
4. For video data, we needed to decide how much attention to pay to which types of information
5. Non-trivial work results must be captured
   - either by annotating codes to data (see below)
   - or by writing or revising a memo

GTM overview: Key **activities** and ***notions***

Freie Universität Berlin

- **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

Um….where are we?

A few slides back, this was the mission:

- We will now look at such PP data
  - to understand Open Coding
  - to get an idea of Theoretical Sensitivity
  - to get an idea of field observation data

- The latter two are done.

- So let's look at actual Open Coding:

- Review the transcription two slides up
  - or use your own
- Annotate codes to 4-8 stretches of text
  - perhaps use the PDF of the slides
  - at least one code should occur twice
  - Use Theoretical Codes that *explain*, not only describe.

Example: possible Open Codings for ZA4-280

- quotation {code}
- leftover to-do

P2: So, jetzt interessieren wir uns also, ob wir das schon drin haben. Und zwar unsere aktuelle Position. {suggest-next-goal}

Jetzt wird's komplizierter. {state-difficulty}

P1: Drin müsste eigentlich alles gespeichert werden. {state-sw-property}{uncertain}

P2: Was liegt, was legen wir denn da rein? {ask-sw-property}

P1: Das *result*. {state-sw-property}

P2: Ein *result* ist ja zu wenig. {state-impossibility}

P1: Nee, das müssten mehrere sein. {justification}
Also *results*. {state-sw-property} Sorry.

P2: Ja, aber das ist auch...nicht so einfach. {state-difficulty}

Na, *results* ist ja das Oberteil, das Gesamtding. {justification} Das kannste nicht unten irgendwo mitspeichern. {state-impossibility}
Und in den Cache[...]

P1: was da passiert ist, dass Du ein *result* hast für Teilnummern. {state-sw-property}

P2: Ja, das müssen wir aber schreiben. {suggest-next-goal}
Unser *results* tut das nicht, momentan. {state-sw-property}

- What about your own Open Coding is fundamentally different from this?

Example Open Codings in ZA4-280: Discussion

1. jetzt interessieren wir uns also, ob wir das schon drin haben. Und zwar unsere aktuelle Position. {suggest-next-goal}
   - grammatically, the utterance is a statement of fact
     - the locutionary speech act
   - but its function in the discourse is making a suggestion
     - the illocutionary speech act
     - **Theoretical Coding:**
       Coding illocutionary acts explains more than coding locutionary acts

2. Drin müsste eigentlich alles gespeichert werden.
   {state-sw-property}{uncertain}
   - {uncertain} is a *property* of {state-sw-property} and refers to the latter

- This was an example of (roughly) "**line-by-line coding**"
  - The resulting codes probably have low quality
  - But the approach gets one going quickly.
  - Constant Comparison helps to consolidate and improve such codes into proper concepts
- Expect much iteration

GTM overview:
Key **activities** and ***notions***

Freie Universität Berlin

- **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

Still-harder **Theoretical Sensitivity**: PP scene ZB7-25

- In the next scene, the pair detects misbehavior of their IDE and discusses whether to restart it.

- View it at least twice and list 3-6 reasons why this scene will be even more difficult to conceptualize.

PP scene ZB7-25:
Why it is still harder for **Theoretical Sensitivity**

Some reasons:

1. Both speaking at once
   - Some incomprehensible speaking
2. Relevant, fast activity on screen
   - also in parallel with speaking
3. Larger codebase → Researcher has less understanding of it
4. Utterances rely on context knowledge from outside the scene
5. Targets an infrastructure problem, not PP as such

→ initially overwhelming

If it is hard to understand what's going on, it is hard to judge what is relevant or how to conceptualize it usefully.

(A consolation:

- A lot of context knowledge can be had by viewing the beginning of the session)

What to do?

There are techniques for strengthening Theoretical Sensitivity:

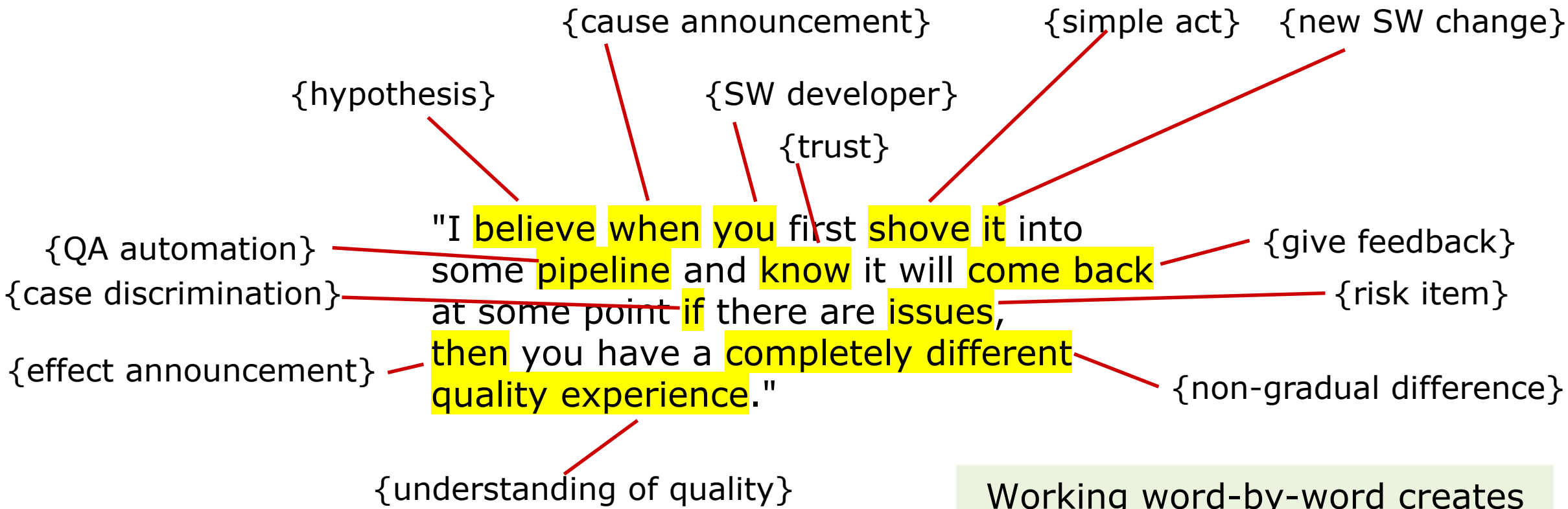
Enhancing **Theoretical Sensitivity** [StrCor90, Chapter 6] (1): Basic questioning

Freie Universität Berlin

Ask

- Who?
- When?
- Where?
- What?
- How?
- How Much?
- Why?

Enhancing **Theoretical Sensitivity** (2): Word-by-word coding (example from Quality Experience study)



Working word-by-word still must use context to understand meaning!

Working word-by-word creates attention to many things that may be relevant

QA: Quality Assurance

fictitious

Theory memo "Quality Experience" (QE)

QE may be a good label for the core category.

Should capture all QA elements that a no-tester team needs to succeed without testers.

Theory must explain how they play together; preconditions (→ why some teams have testers); consequences

Sounds like with-tester teams desire those same things?

→ Theoretical Sensitivity: search for these!

Enhancing **Theoretical Sensitivity** (4): Multiple meanings of a word or phrase

"I believe when you
first shove it into some pipeline
and know it will come back
at some point if there are issues,
then you have a completely different
quality experience."

"Multiple meanings"
emphasizes creativity
and produces ideas

Again: observe context!

1. Canonical order of steps ("first")
2. Simple step for developer (just shove)
3. Developer is powerful (can shove!)
4. Need not think, cannot make mistake
5. Second step will follow soon
6. Pipeline can do different things ("it", "some")
7. Pipeline can be complex
8. Pipeline is a black box.

Enhancing **Theoretical Sensitivity** (5): Consider alternatives: What if…?

Freie Universität Berlin

1. Canonical order of steps ("first")
2. Simple step for developer (just shove)
3. Developer is powerful (can shove!)
4. Need not think, cannot make mistake
5. Second step will follow soon
6. Pipeline can do different things ("it", "some")
7. Pipeline can be complex
8. Pipeline is a black box

Note the example stacks
techniques 3 and 4
on top of each other

1. What if the QA process is unclear?
2. What if QA is difficult to start?
3. What if the developer cannot start it?
4. What if QA-starting mistakes occur?
5. What if QA takes long? (*)
6. What if QA is inflexible? Too limited?
7. (ditto)
8. What if developer needs detailed knowledge about how QA will proceed?

And then go and look for such things in the data.

(*) relevant in Quality Experience study

Enhancing **Theoretical Sensitivity** (6): Watch for red flags

- When a statement is very categorical (does not allow for exceptions), question its validity. Trigger words:
  - "always", "never"
  - "obviously"
  - "must", "cannot", "will"
  - and many cousins of these
- In particular in interviews!

Example:

"I believe when you first shove it into some pipeline and know it will come back at some point if there are issues, then you have a completely different quality experience."

Red flag suggests what?

- Sometimes "it will" *not*
  - although a defect exists
- and this possibility induces a key part of the story

End of part 1

- **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

Thank you!