Freie Universität Berlin

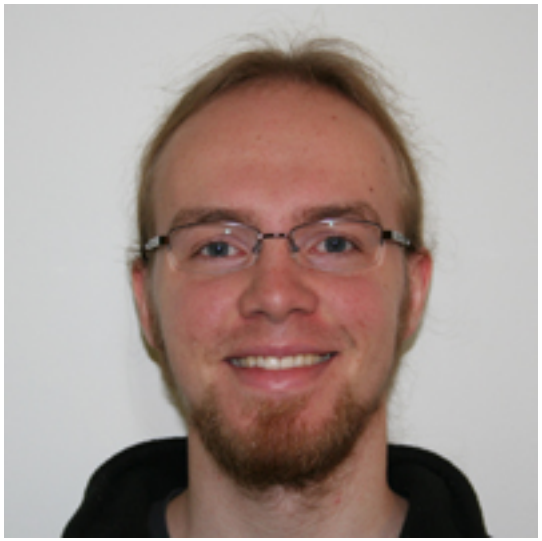
Observations on **Knowledge Transfer** of **Professional** Software Developers during **Pair Programming**

38th Int'l. Conference on Software Engineering (ICSE), 2016

Franz Zieris
zieris@inf.fu-berlin.de

Lutz Prechelt
prechelt@inf.fu-berlin.de

About me



- Franz Zieris
- AG Software Engineering (Lutz Prechelt), since 2012
- What I do:
  - Organize the exercise part of our group's lectures
  - Organize seminars and summer projects
  - Supervise students' theses
  - Administrate the open source project Saros
    - www.saros-project.org
  - Oh wait … research! *I'm also a researcher!*

Our research

- <u>Overall:</u> *"We are concerned with reducing the gap between the* State of the Art *and the average* State of the Practice *in all areas of software engineering."*
- <u>In research:</u> *"Understand what goes on in software development processes and why."*

- <u>In particular:</u>
  - Agile software development methods and practices, *such as*
  - Pair Programming

Pair Programming

Pair Programming: What & Why?

- **Claimed Advantages**
  - Faster than solo
  - Fewer defects
  - Better software design
  - More concentrated work
  - 2 people familiar with code
  - Can employ joint knowledge
  - Faster learning
  - Enjoyable

- **Use it for**
  - Shorter time-to-market
  - Higher quality
  - Reducing code knowledge bottlenecks
  - Knowledge transfer

- **Disadvantages/Issues**
  - Some developers do no like it
  - May require more resources

Usual approach

- Let *n* students solve the same task
  - some work in pairs, others alone
  - measure time to completion
  - publish statistically significant ($p<0.05$) difference between respective means

Problems with usual approach

Long list:

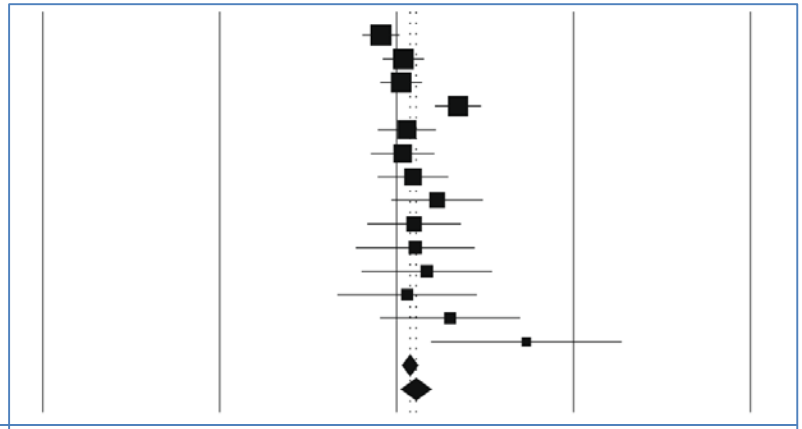
- Did the students know each other?
  Were they allowed to choose their partner?
- Were the students skilled in the relevant areas?
- What was the task's focus? Algorithmic problem?
  New feature? Bug fix? Refactoring? Simple or complex?
- What about internal code quality?
- What about long-term effects?
- What about "soft factors", e.g. confidence or satisfaction?
- Students are not professionals (years of experience, not used to the work environment, ...)
- ...

And besides: The quantitative result does not give any explanations as to *how* the difference comes to be.

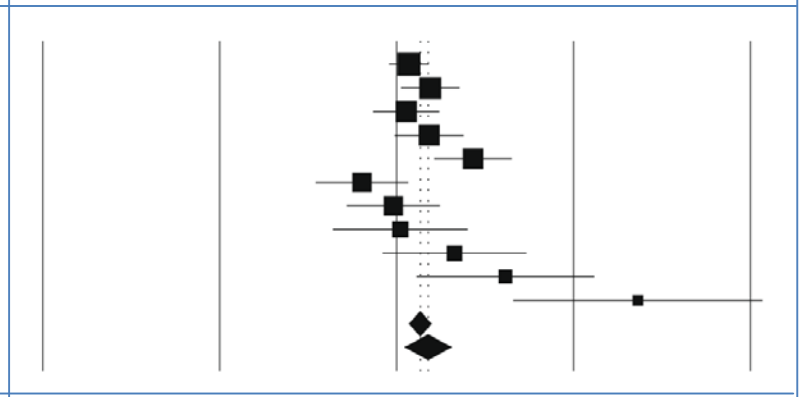
Meta-Analysis

What do you think?

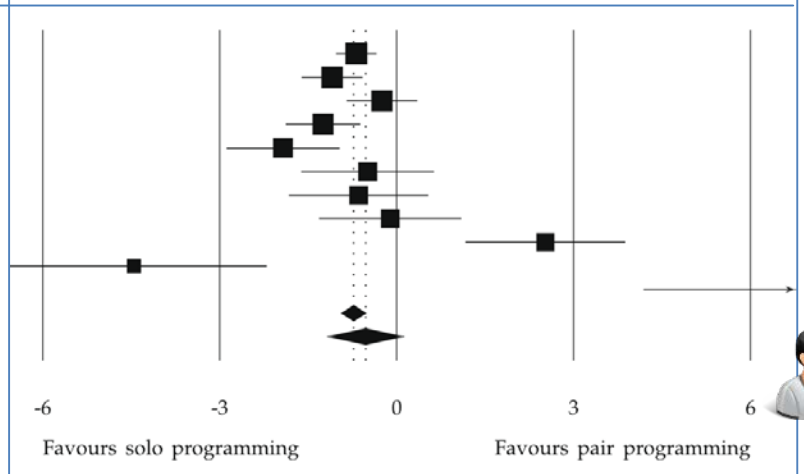
– Quality



– Duration

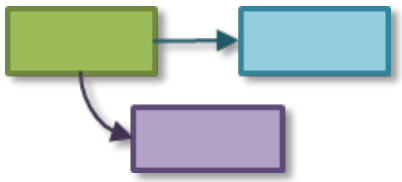


– Effort



Hannay et al., "The effectiveness of pair programming: A meta-analysis", Information and Software Technology 51, 2009

Qualitative Research



- Understand relevant concepts
- Identify mechanisms that explain observable effects

- Without such understanding of a field, quantitative methods will measure just *something*
- Software engineering research: often not mature enough for quantitative methods
  - at least for many interesting questions

We almost arrived at the core topic!

Typical pair programming scenarios

1. Solve a difficult problem

- Combined ideas
- Combined background knowledge

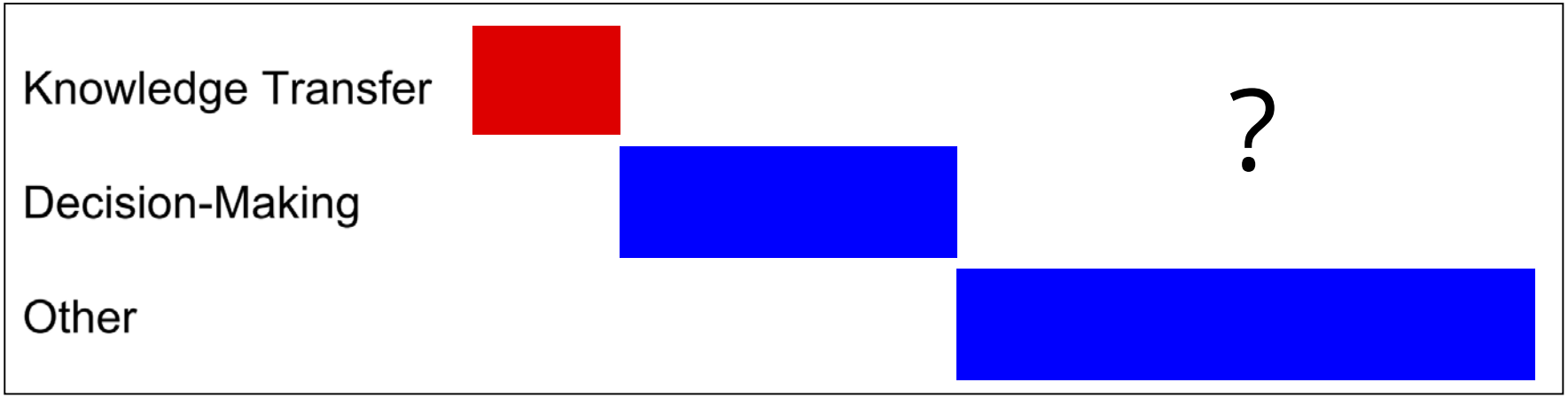
Developer ⟷ knowledge ⟷ Developer

2. Introduce new team member

- System understanding
- Common practices, coding conventions

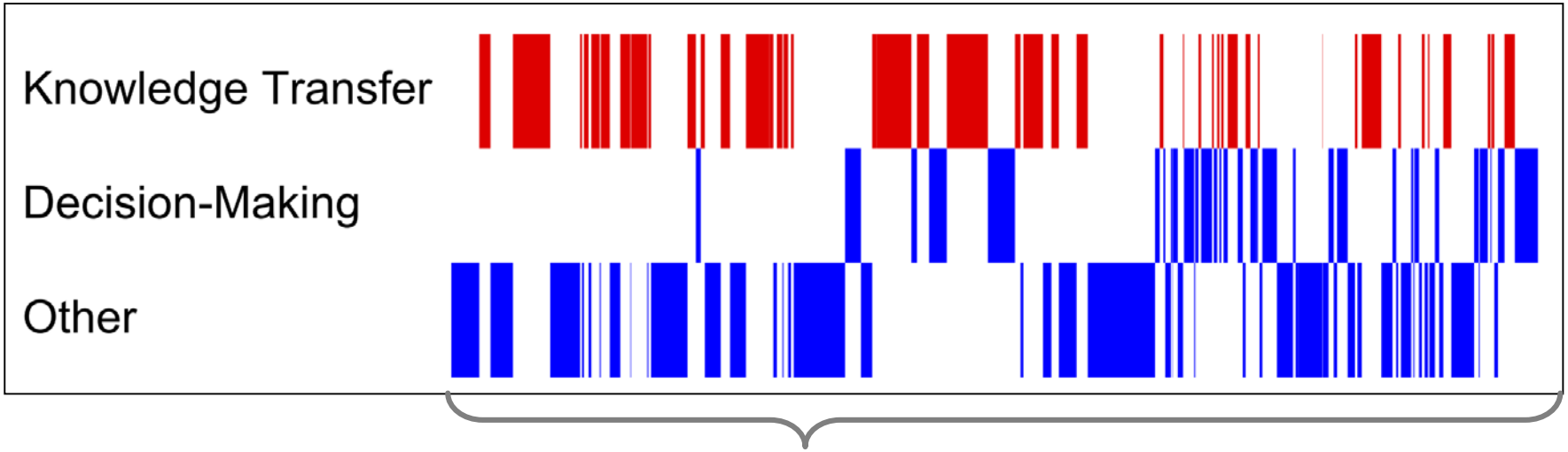
Senior member → knowledge → Newbie

Knowledge Transfer in PP Sessions



- Categories
  - **Knowledge Transfer**: information exchange, clarification, etc.
  - **Decision-Making**: What to do next? How to do it? ...?
  - **Other**: e.g. direct computer-interaction

Knowledge Transfer in PP Sessions

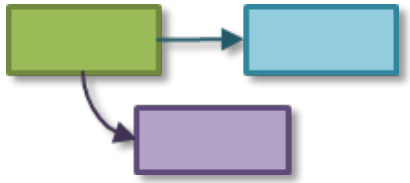


- Session gross length: ca. 2 hours
- Categories
  - **Knowledge Transfer**: information exchange, clarification, etc.
  - **Decision-Making**: What to do next? How to do it? ...?
  - **Other**: e.g. direct computer-interaction
- Knowledge Transfer: **ca. 35%** of the time

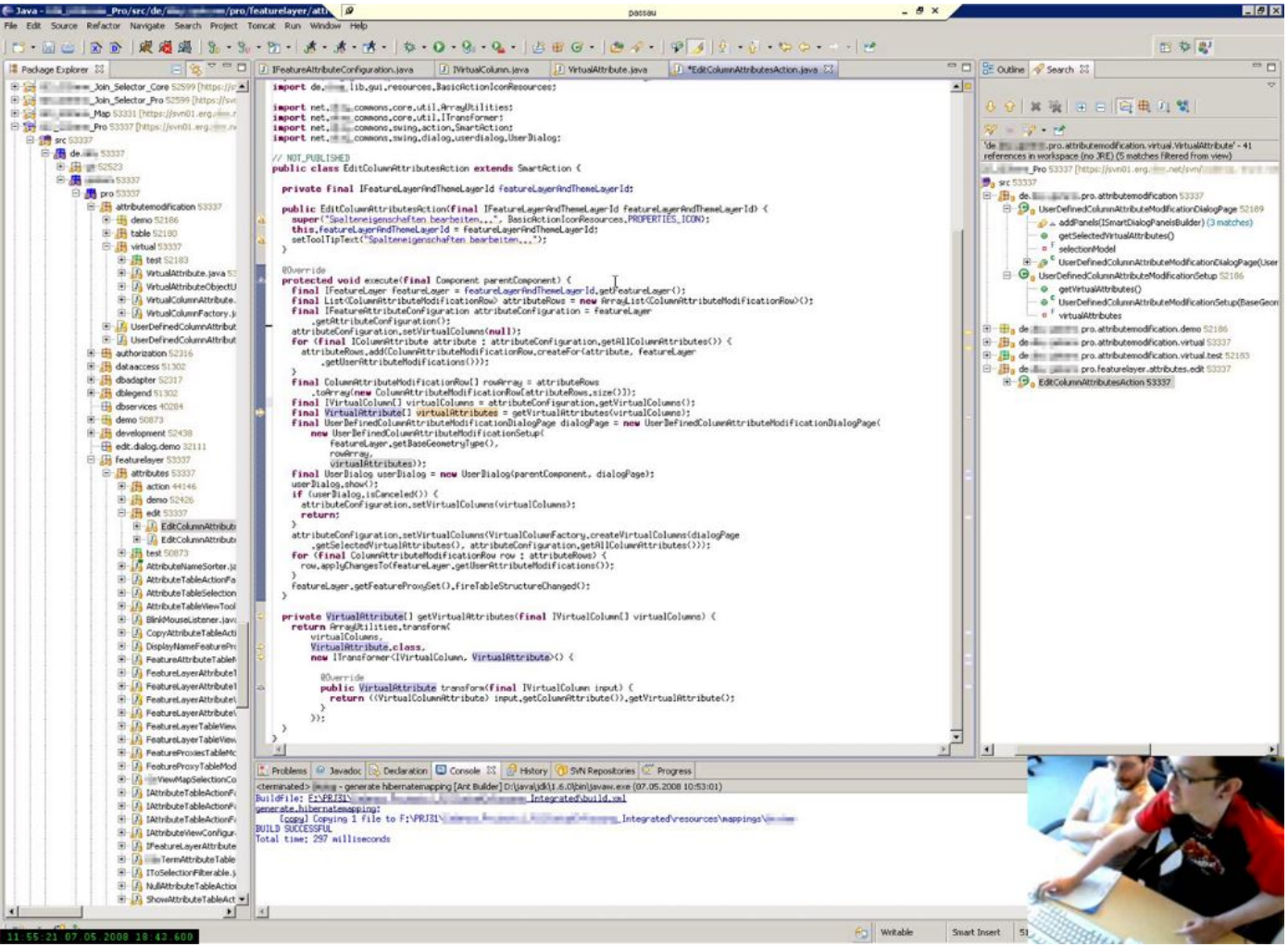
Research Goal

Characterize **how** effective and efficient **knowledge transfer works** in pair programming.

- Geared towards practitioners:
  - Formulate **patterns** of beneficial and problematic behavior



Data collection



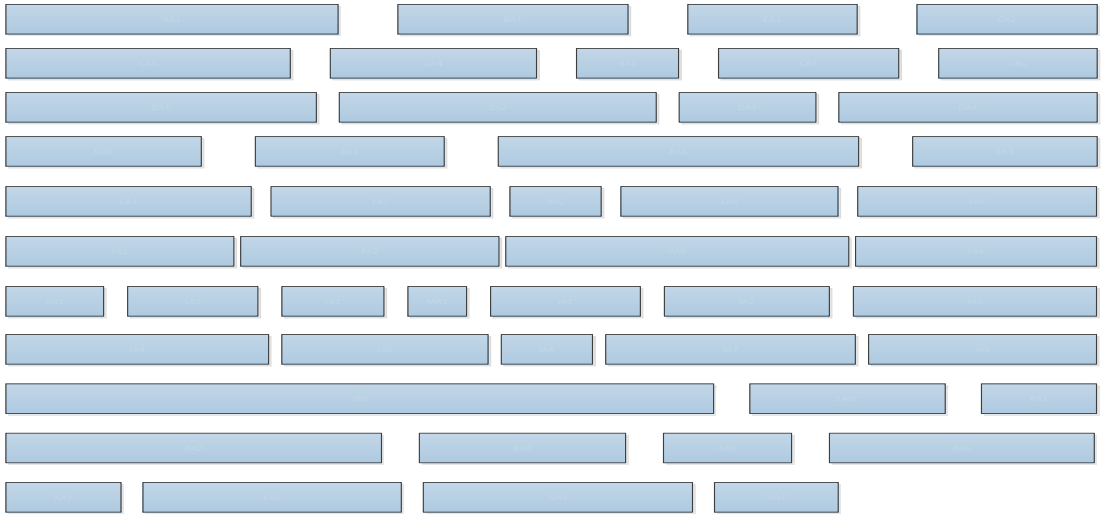
Authentic

- In-vivo recordings of professionals: no artificial pairs, tasks, or settings

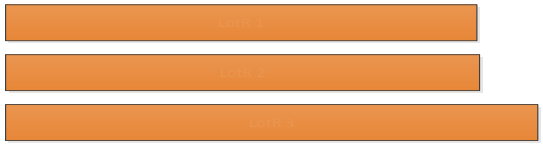
Diverse

- different domains & pair types

Data analysis

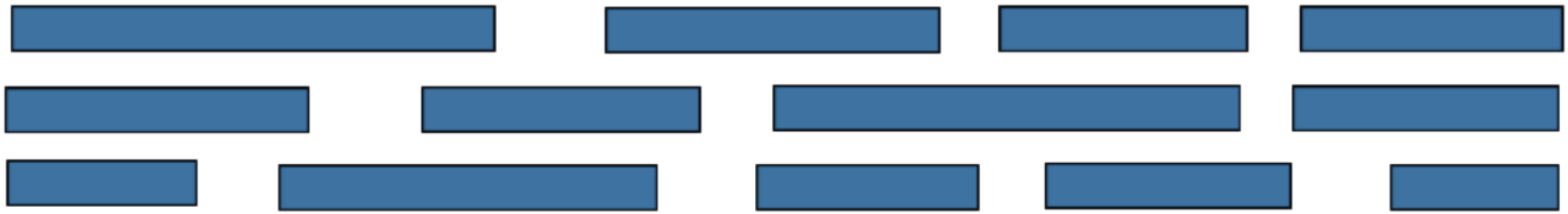


49+ Sessions from **11** companies



Lord of the Rings I-III

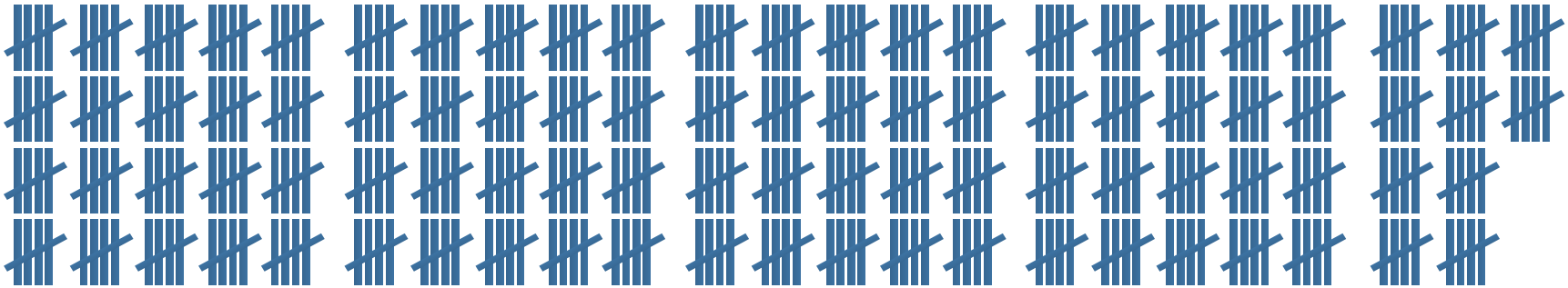
Data analysis



13 Sessions selected

Grounded Theory
(Strauss & Corbin)

450 Knowledge Transfer Episodes analyzed



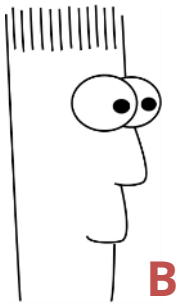
Three Examples

Example 1:
Efficient Pull Episode

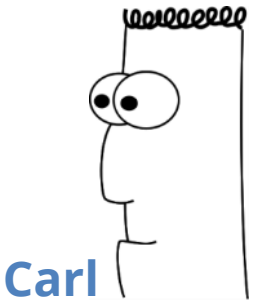
Context: Java, with complex self-written framework
Task: Get business object (“MiniObject”) from proxy object (“objectHandle”)

```
return objectHandle.fetchM;
```

- fetchMiniObject() CMMiniObject
- fetchMiniObject(boolean allowMicroObject)



Bob



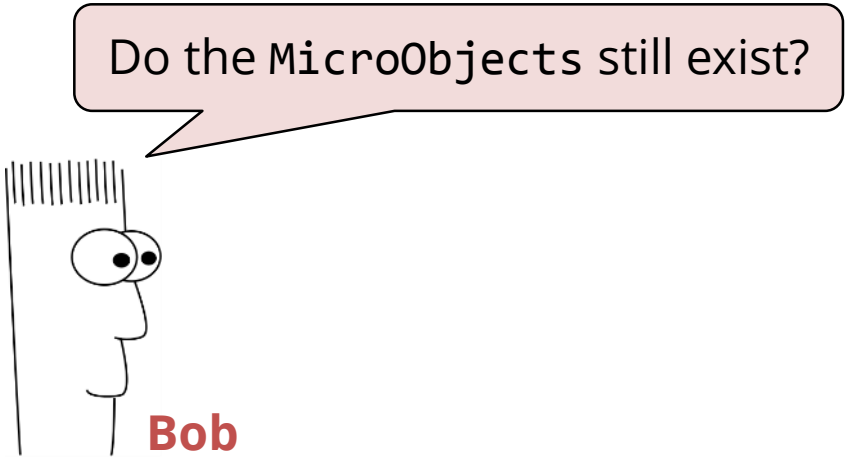
Carl

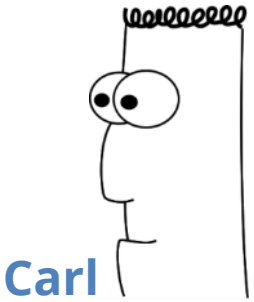
Example 1:
Efficient Pull Episode

Context: Java, with complex self-written framework
Task: Get business object (“MiniObject”) from proxy object (“objectHandle”)

```
return objectHandle.fetchM;
```

- fetchMiniObject() CMMiniObject
- fetchMiniObject(boolean allowMicroObject)



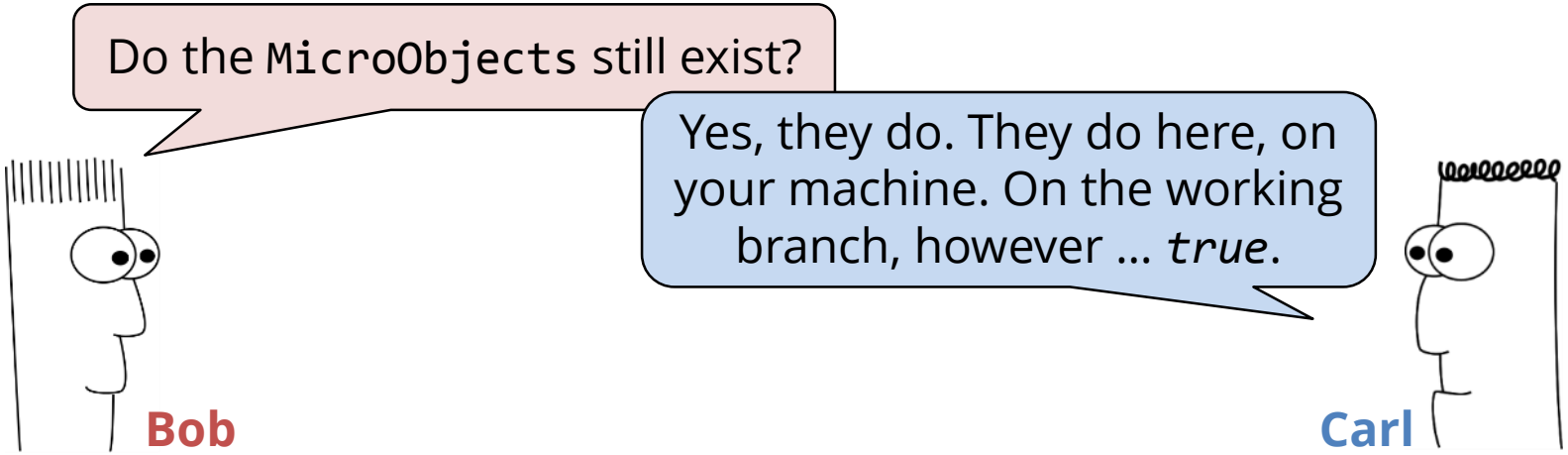


Example 1:
Efficient Pull Episode

Context: Java, with complex self-written framework
Task: Get business object (“MiniObject”) from proxy object (“objectHandle”)

```
return objectHandle.fetchM;
```

- fetchMiniObject() CMMiniObject
- fetchMiniObject(boolean allowMicroObject)

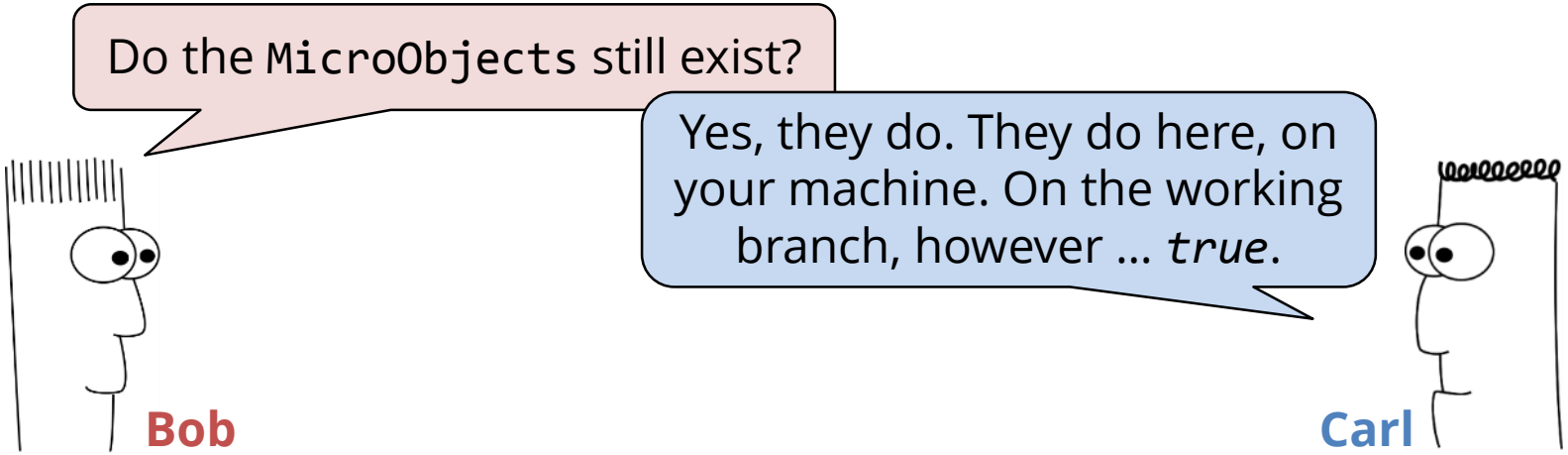


Example 1:
Efficient Pull Episode

Context: Java, with complex self-written framework
Task: Get business object (“MiniObject”) from proxy object (“objectHandle”)

boolean allowMicroObject

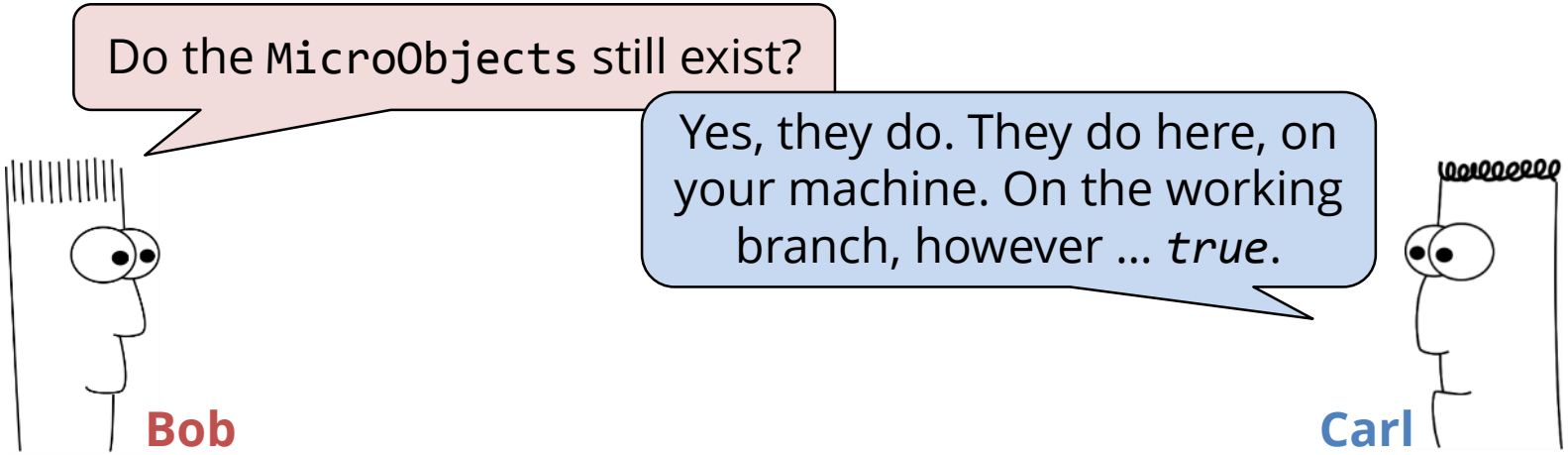
```
return objectHandle.fetchMiniObject(allowMicroObject);
```



Example 1:
Efficient Pull Episode

Context: Java, with complex self-written framework
Task: Get business object (“MiniObject”) from proxy object (“objectHandle”)

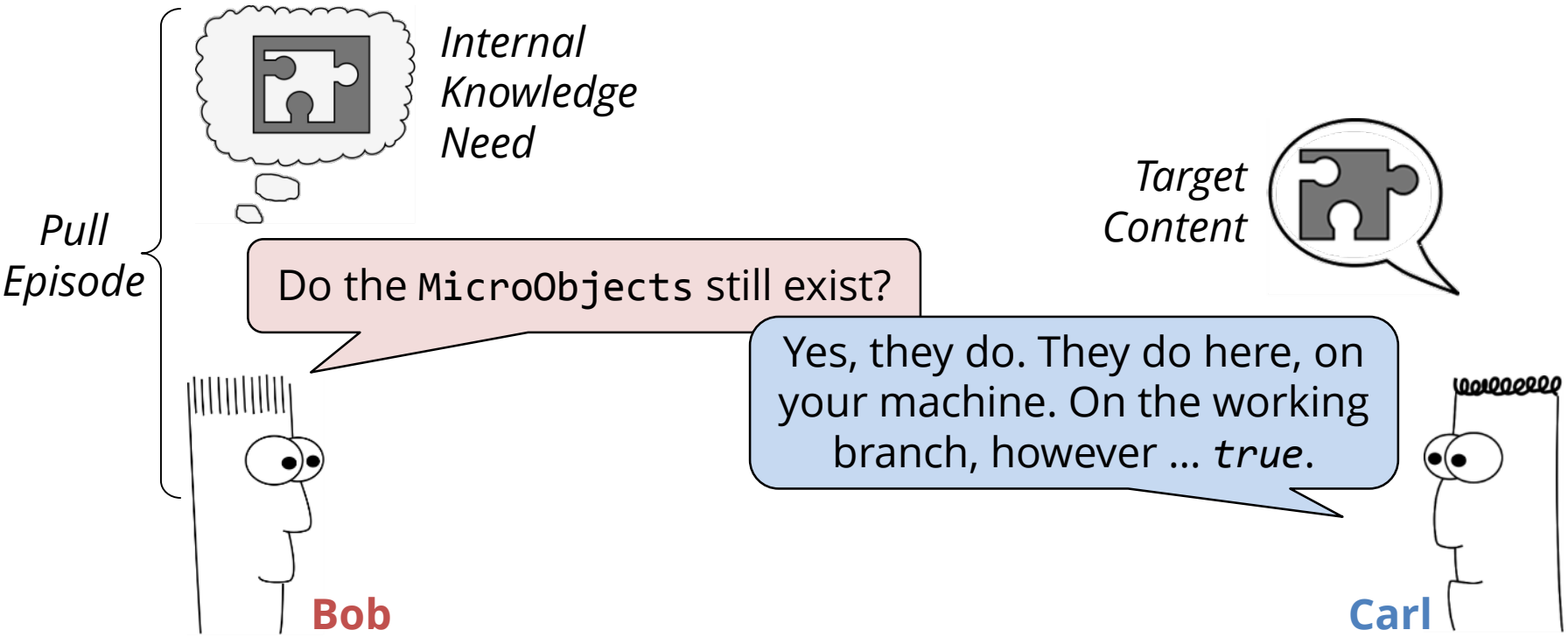
```
return objectHandle.fetchMiniObject(true);
```



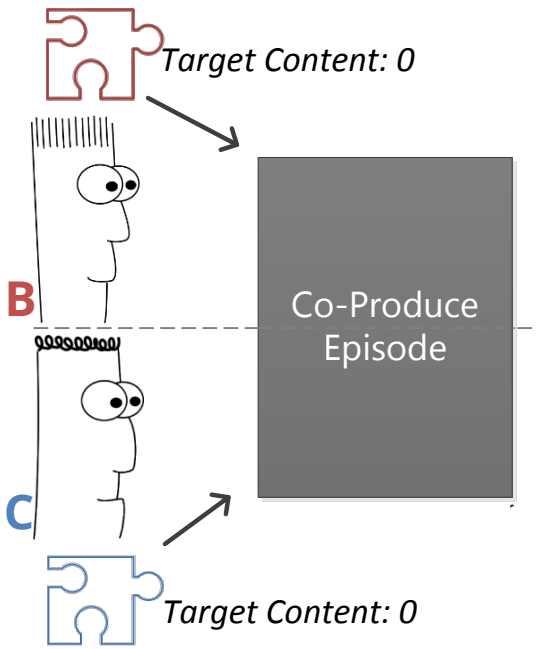
Example 1:
Efficient Pull Episode

Context: Java, with complex self-written framework
Task: Get business object (“MiniObject”) from proxy object (“objectHandle”)

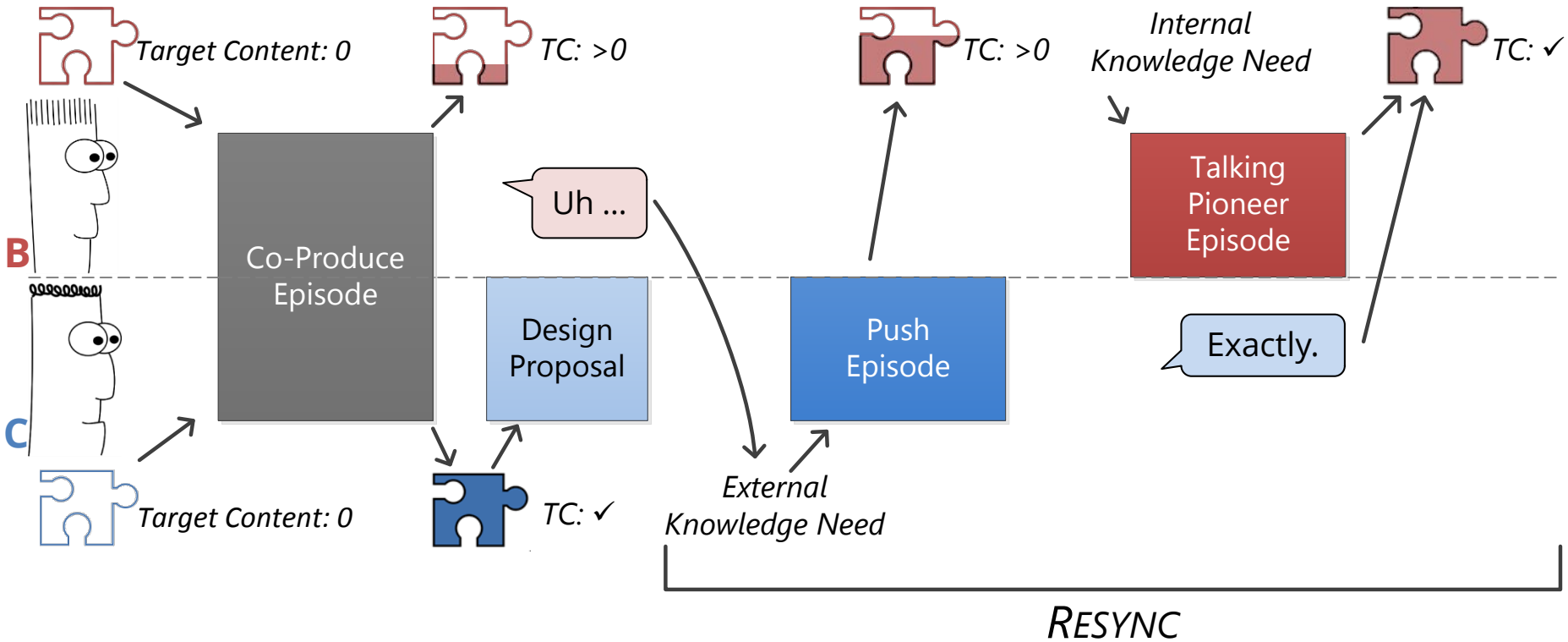
```
return objectHandle.fetchMiniObject(true);
```



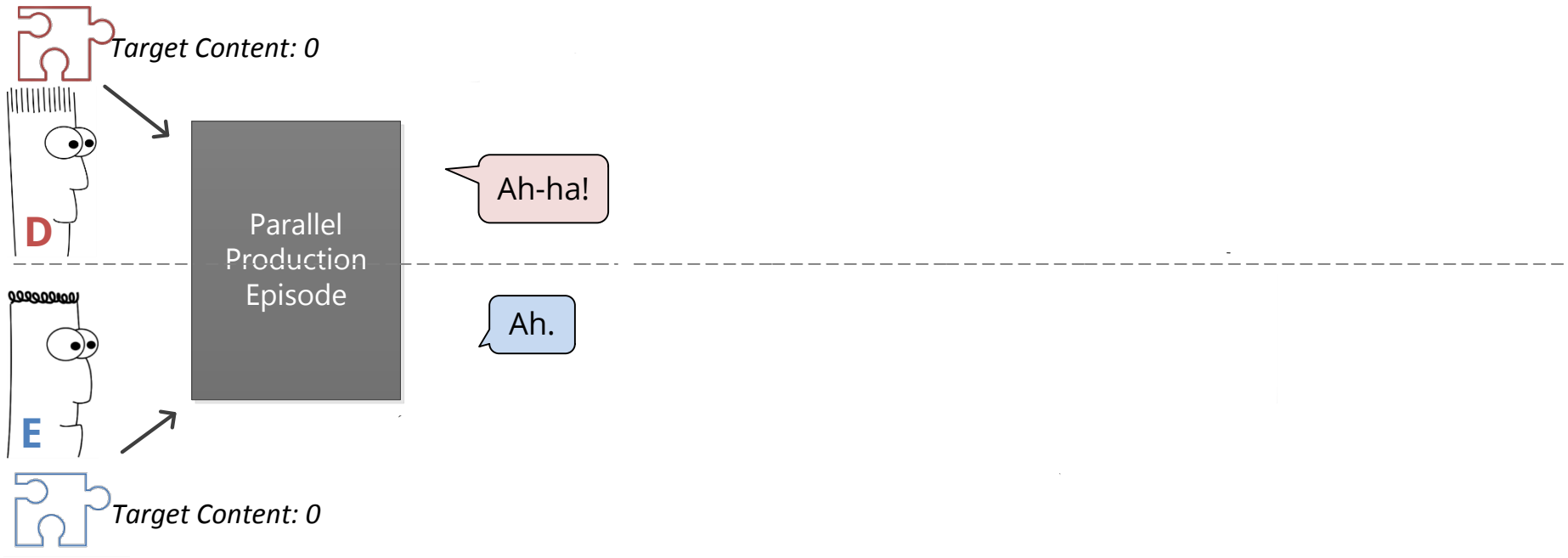
Example 2: (earlier, same session)
Effective Resynchronization



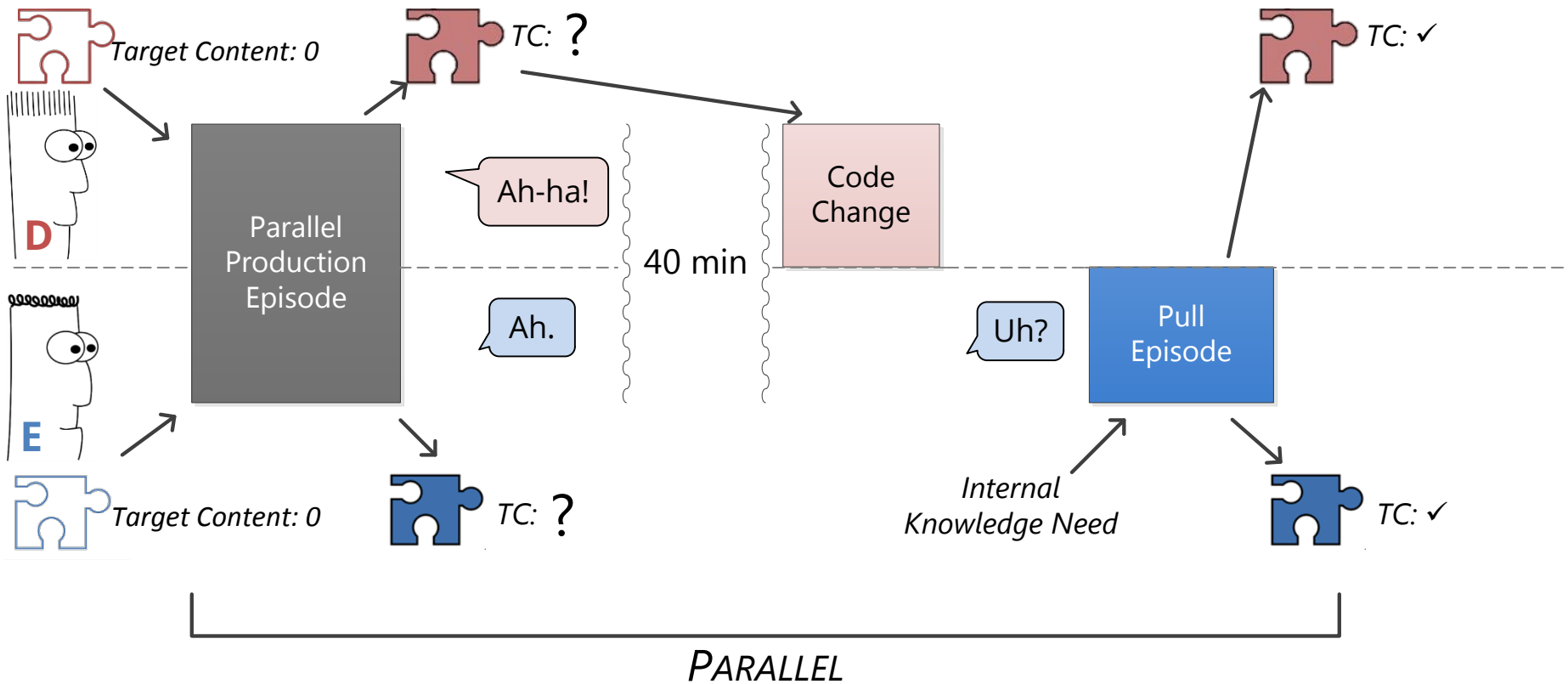
Example 2: (earlier, same session)
Effective Resynchronization



Problematic Example 3: (different pair) **Parallel Work**



Problematic Example 3: (different pair) **Parallel Work**



Researcher's perspective:
E's confusion makes the effects of the Parallel Production Episode visible.

Summary of the examples

- Knowledge transfer episodes **can be very efficient** (with very few utterances)
  - … if the pair **invests** in staying close together.
- If they don't, episodes can become much **longer** and may take **several attempts**
  - (not shown here)
- Even worse, the pair risks working on **parallel** tracks (as in the last example)

Results

On Pair Programming in general

- No pair member more knowledgeable in all regards
- Positive effects of knowledge-wise inferior member

(Anti-)Pattern candidates

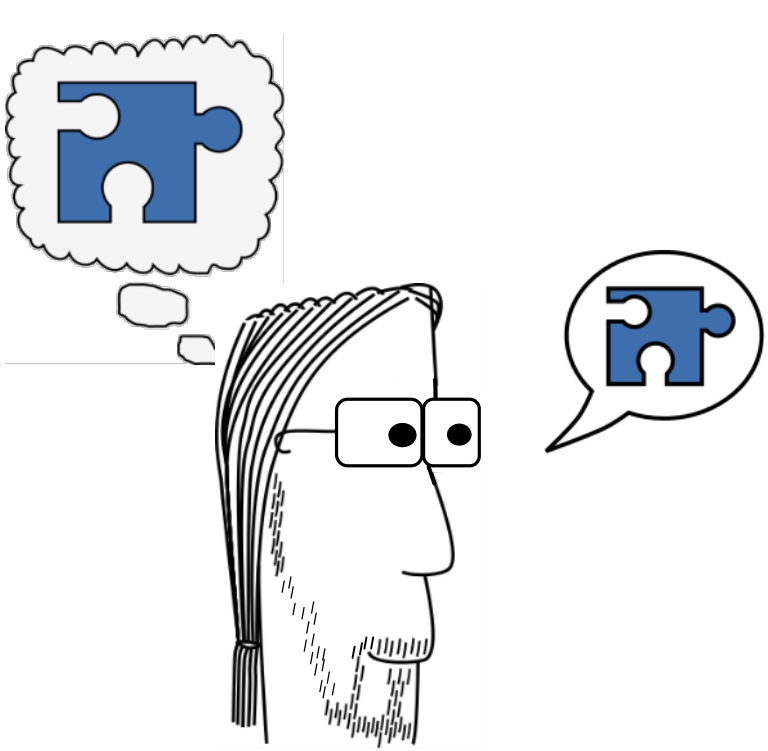
- Resync
- Parallel
- Talking Pioneering

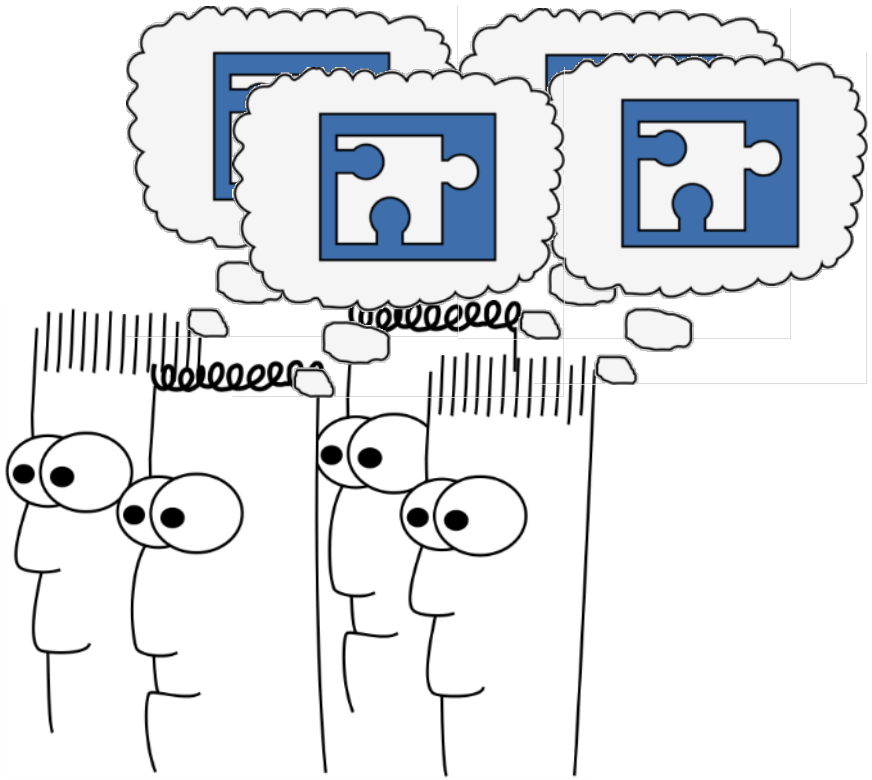
Open: How to teach this?

Mechanisms of knowledge transfer episodes

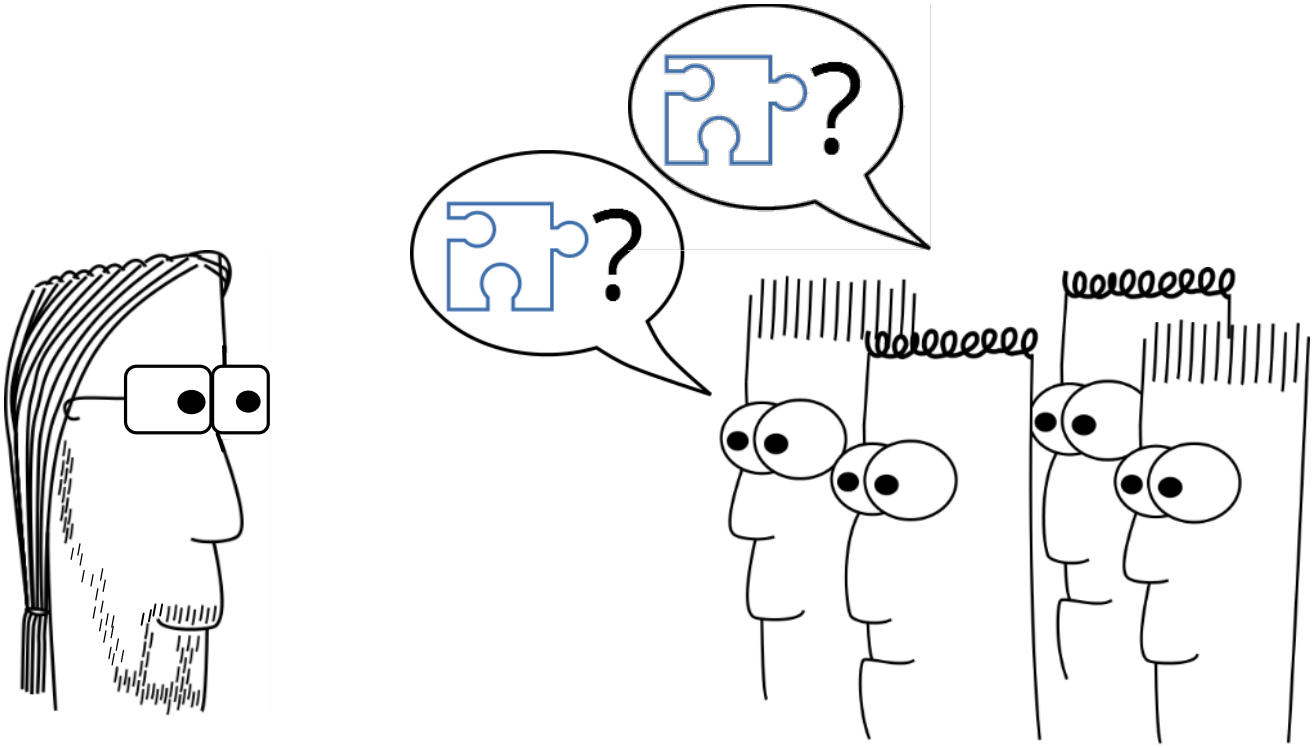
- Push & Pull
- Co-Produce & Pioneering Production

Enough with Push …





… time for Pull!



Thank you!

https://bitbucket.org/spooning

Used Images

https://web.archive.org/web/20080509191418/http://www.cenqua.com/pairon/

https://www.flickr.com/photos/a2gemma/552208117

http://www.iconarchive.com/show/oxygen-icons-by-oxygen-icons.org/Categories-system-help-icon.html

http://www.iconarchive.com/show/vista-people-icons-by-icons-land/Person-Male-Light-icon.html

http://www.iconarchive.com/show/vista-people-icons-by-icons-land/Groups-Meeting-Light-icon.html

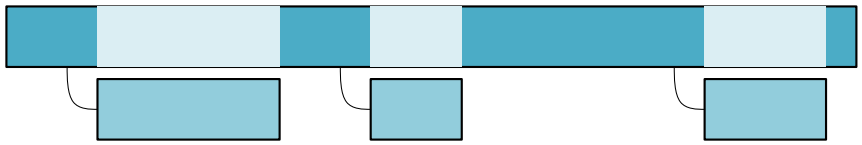
http://www.iconarchive.com/show/xedia-icons-by-photoshopedia/My-Computer-icon.html

http://www.iconarchive.com/show/nuoveXT-2-icons-by-saki/Devices-keyboard-icon.html

Episode Infrastructure

- The Episode **infrastructure** of a session

Main Topic, paused for Sub-Topics



New Topics (colors) preclude main Topic's conclusion