Methodology and Study Design in Empirical Software Engineering: Two Case Studies

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik
http://www.inf.fu-berlin.de/inst/ag-se/

- The scientific method
  - Controlled experiments
- Case 1: Pair Programming (PP)
  - state of knowledge
  - research questions
- Case 2: Technological platforms
  - state of knowledge
  - research questions
- Research approaches
  - Pair Programming studies
  - Plat_Forms
- Nature of results
  - Pair programming studies
  - Plat_Forms

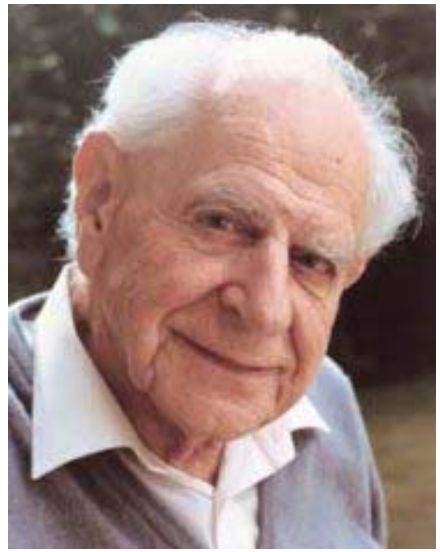
The scientific method

- Since Galilei, physics and other sciences work according to this model (applied iteratively):
  - Formulate a theory T describing how (some aspect of) the world behaves
  - Design and conduct experiments E for testing this theory
- Is accepted in all subjects where experimentation is possible
  - Natural sciences: Physics, chemistry, biology, medicine etc.
  - Engineering
  - Parts of some social sciences such as economics, sociology, etc.
- Is problematic where experiments cannot be performed
  - because of technical or ethical problems

The scientific method (2)

- Note the following:
  - T is called a scientific theory only if it predicts something specifically and hence can be tested
  - Even if T is wrong, it may happen that the results of E are as expected
  - But if E contradicts predictions of T, then T must be false

- This view of science was suggested by Karl Popper (1904–1994)
  - It is the prevalent scientific paradigm today
  - In this view, theories cannot be directly confirmed, only refuted
  - If a theory cannot be refuted for a long time, it will gradually be accepted as confirmed
    - example: special theory of relativity



Experiments and control

- When we empirically investigate something
  - we characterize the situation by a set of ***input variables***
    - usually quantitative or categorial
    - e.g. "team size = 4" or "design method used = A"
  - and the observations by a set of ***output variables***
  - If we choose the value of at least one input variable, the study is called an ***experiment***

- The act of consciously manipulating the values of input variables is called ***control***

- Every empirical study assumes that there is some systematic relationship between inputs and outputs
  - If we have a certain expectation about this relationship, this is called a ***hypothesis***
  - Any additional factors influencing the outputs are called ***extraneous variables***

Controlled experiments

Controlled experiment:

- Vary inputs variables systematically
  - Often only one input is varied and only two levels are used
- Keep all extraneous variables fixed
- Observe what changes in the output variables
  - Specifically: Look if the predictions of your theory hold up

Works well in physics and chemistry.

Is difficult whenever human beings are involved in the setup

- Because they bring in a shipload of extraneous variables (such as intelligence, knowledge, skills, *Tagesform*, etc.)
- Solved by
  - employing groups of human subjects (repeated measurement),
  - averaging the results, and
  - hoping all differences level out

Case 1: Pair Programming (PP)

- An old programming practice
  - recently became well-known along with eXtreme Programming
- Two programmers work side-by-side, using only one keyboard and monitor
  - Sometimes called *Driver* and *Observer*
  - They switch roles frequently (every few minutes)

- PP is hoped to be benefical for
  - productivity
  - design and code quality
  - spread of knowledge (domain, designing, design, code, technology, methodology)

PP: State of knowledge

- A number of controlled experiments have been performed comparing PP to single-programmer settings
  - and also some anecdotal evidence is available

Findings:

- Pairs are usually faster than single programmers
  - usually somewhere in the range from 10% to 90%
- Pairs are often subjectively happier than single programmers
  - and more confident in the quality of their results
- Their code is often of a better quality
  - shorter, more readable, fewer defects, better standards conformance

- Only superficial and purely speculative explanations are offered why this is so (mechanisms)
  - or how to optimize the benefits → **There is no theory of PP**

PP: Research questions

Freie Universität Berlin

- What do pairs do during PP?
  - Activities, interactions, differences to solo programming
- Which recurring behavior patterns lead to success?
  - fast progress, high quality
- Which recurring behavior patterns lead to problems?
  - lack of progress, lapses, frustration
- Which "best practices" can be recommended for PP?

Case 2: Technological platforms

- Web-based information systems are probably the most common type of custom SW project today.
- Several different technology packages are available for building such systems
  - e.g. ASP.NET, Java EE, Perl, PHP, Python, Ruby on Rails

They involve

- one or more programming languages and infrastructure
  - compiler/interpreter, web application server, build systems, etc.
- broad libraries of reusable components
- application frameworks
  - sometimes several alternative ones
- perhaps a certain design and work style or even a "culture"
  - e.g. pythonic style, TIMTOWTDI, DRY

Platforms: State of knowledge

- There are loads of anecdotal evidence regarding the characteristics that emerge when using these platforms
  - in particular strengths and weaknesses
  - e.g. "PHP is insecure", "Java EE consumes a lot of memory" etc.
- and plenty of advocacy and zealotry
- but essentially no sound empirical results

→ **There is no theory of platform differences**

Platforms: Research questions

Freie Universität Berlin

Considering the characteristics emerging when using a platform:
Are there typical differences between the platforms regarding

- development processes and work styles?
- productivity?
- quality of the results?
  - usability,
  - security,
  - efficiency/scalability,
  - flexibility,
  - extensibility/maintainability etc.

Research approaches: Differences

Freie Universität Berlin

- As research topics, PP and platforms have similarities and differences

Similarities:

- In both cases, no theory exists that explains what is going on and/or makes useful predictions to be tested

Differences:

1. PP: The research questions are on the level of mechanisms
   - suggesting primarily qualitative approaches and results

   Platforms: The questions concern outcomes
   - suggesting largely quantitative approaches and results
2. PP can be analyzed by itself.

   Platforms should be analyzed by comparison

Research approaches: No controlled experiments

Freie Universität Berlin

In both cases, controlled experiments (CEs) are not a very useful empirical method:

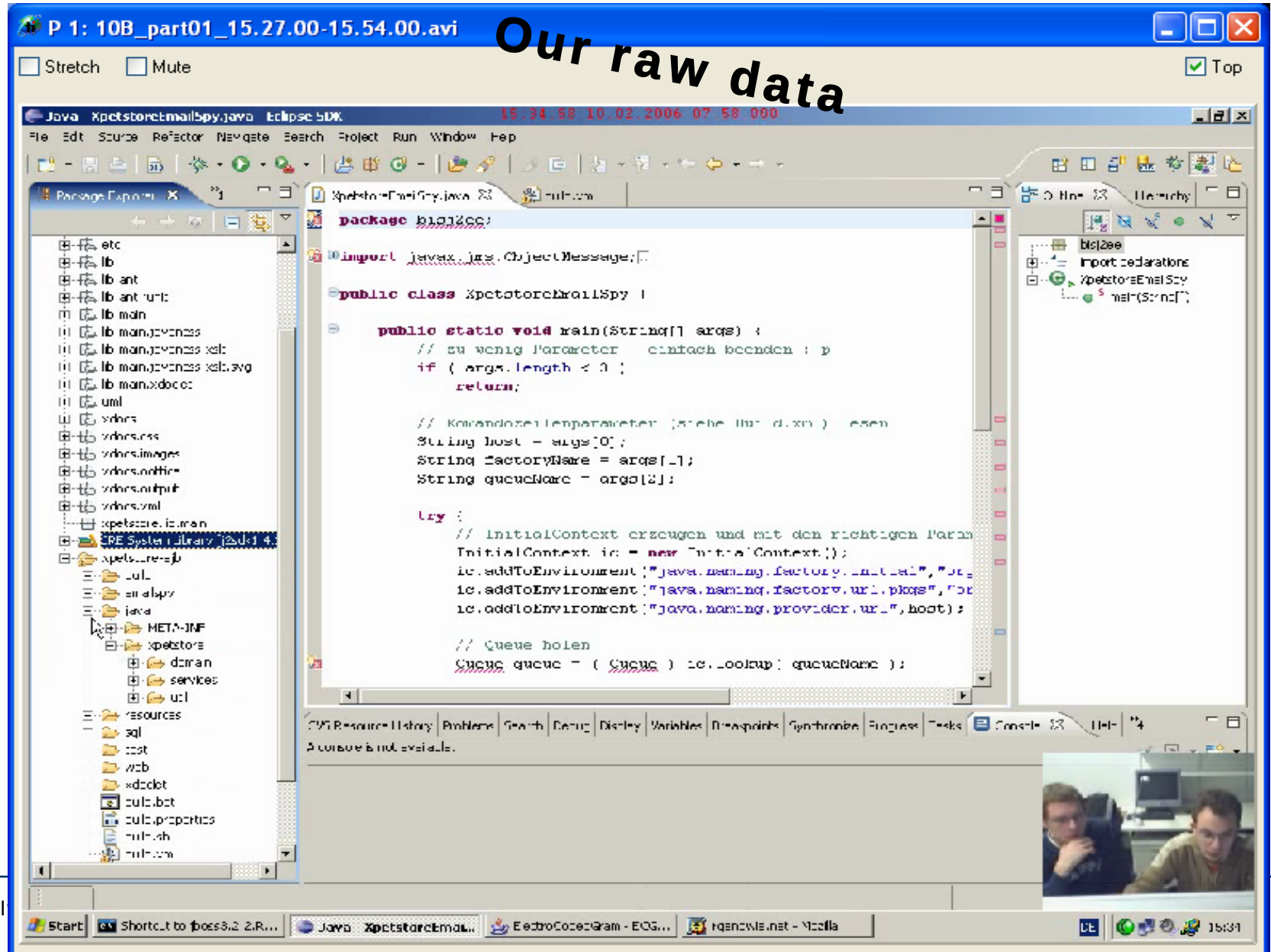
- CEs test hypotheses, but we do not possess interesting hypotheses
  - because we lack theories.
  - That is why most of the existing PP work is so unsatisfactory.
- PP: CEs involve comparison, but our research questions are not interested in comparison
- Platforms: CEs involve randomized assignment to groups, but there are no subjects who can master six different platforms

Approach: Pair Programming studies

- We use the Grounded Theory method to derive a conceptualization (an abstract view) of various PP sessions
  - We record sessions: Video of desktop, video of pair, audio of pair
- The conceptual description is built in a strictly observation-driven manner ("grounded in data")
  - Its structure conforms to a given meta-model
- The first step is developing the set of concepts to be used: A coding scheme
- The expectation then is to find recurring patterns of behavior and to be able to link these to PP success or lack thereof
  - using aggregation, filtering, visualization etc.

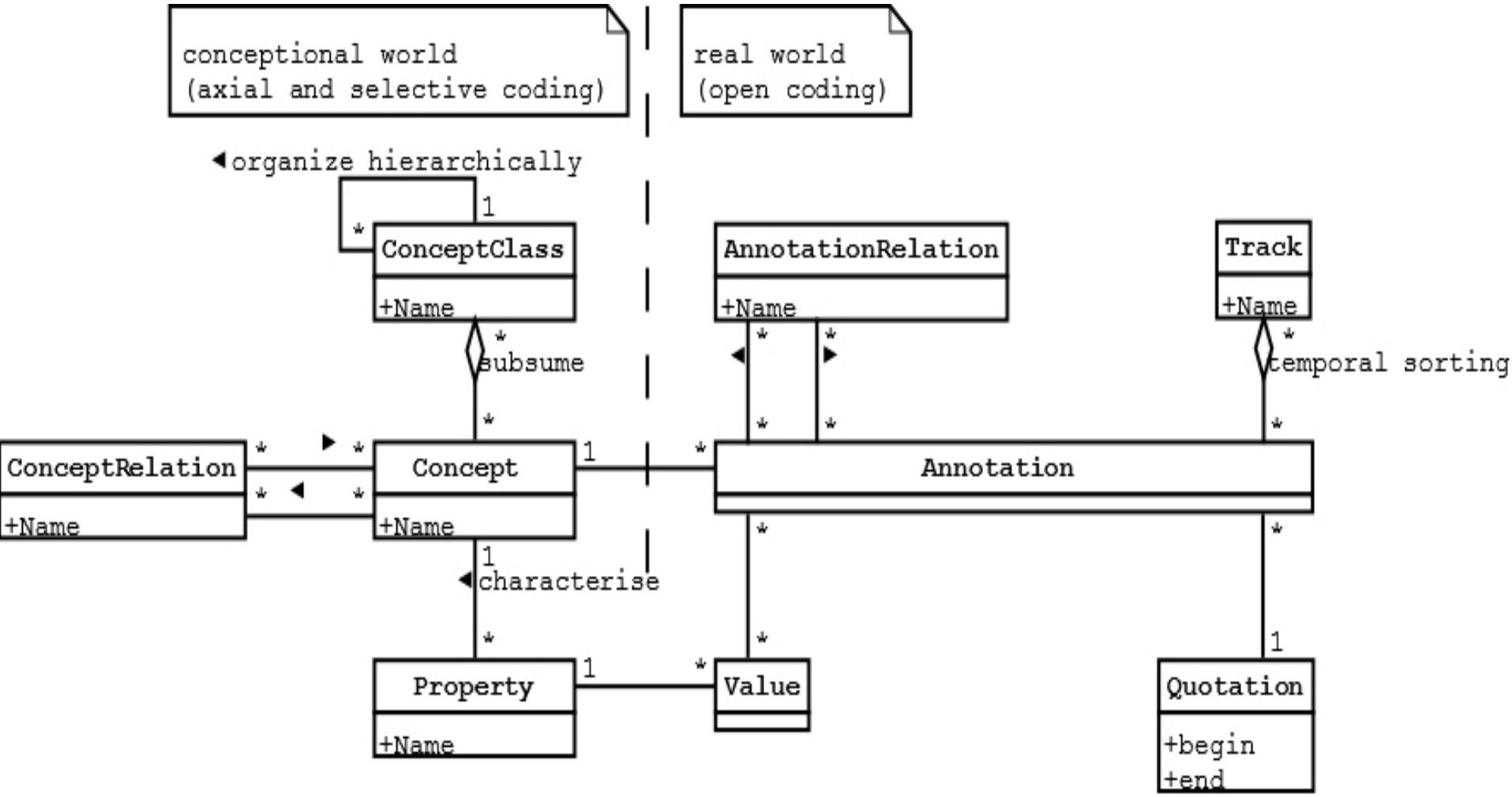
Approach: Pair Programming studies

Freie Universität Berlin



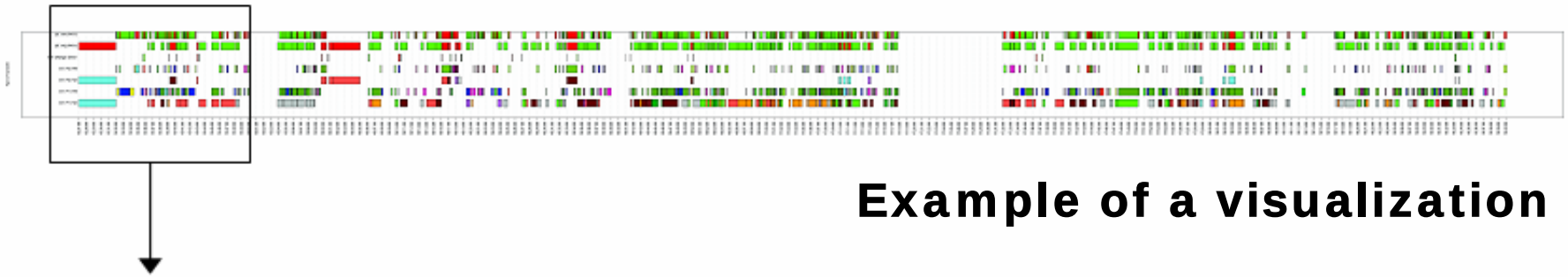
Approach: Pair Programming studies

- Our Grounded Theory meta-model

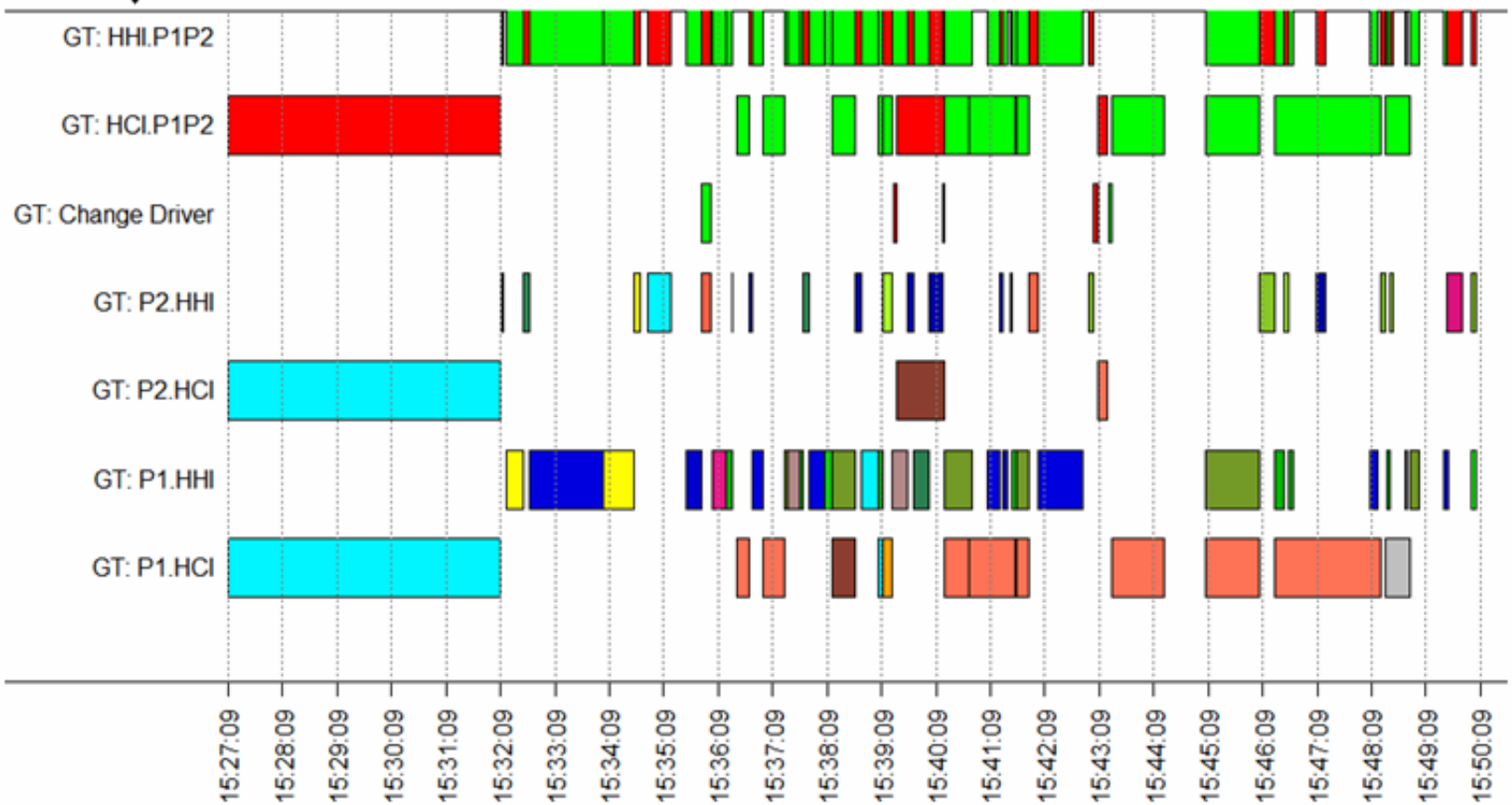


Approach: Pair Programming studies

Freie Universität Berlin



Example of a visualization



Approach: Plat_Forms

- Publicly announce a contest (called "Plat_Forms") for professional teams of 3 developers
  - Held in Nürnberg, January 2007
  - Teams apply for participation, the best ones are selected
- Each team has 30 hours to develop a solution for the same set of 150 fine-grain requirements
  - Task is a simple community portal
- There are 3 teams for each of the platforms
  - Java EE, Perl, PHP (not enough interest from the others)

- Teams submit solutions (source code, version archive)
- Experimenters analyze them thoroughly
  - 4 people, 5 months

Approach: Plat_Forms

Freie Universität Berlin

Plat_Forms The web development platform comparison

- View of the contest site

Approach: Plat_Forms

Plat_Forms uses a quasi-experiment approach:

- There is no randomization in assigning teams to platforms
  - A controlled experiment requires such randomization in order to control for possible selection effects
- Scientifically speaking, quasi-experiments are weaker
- Practically speaking, randomization would make no sense whatsoever
  - Whatever people-specific characteristics were uncontrolled will be just the same when using that platform in practice

Results: Pair programming studies

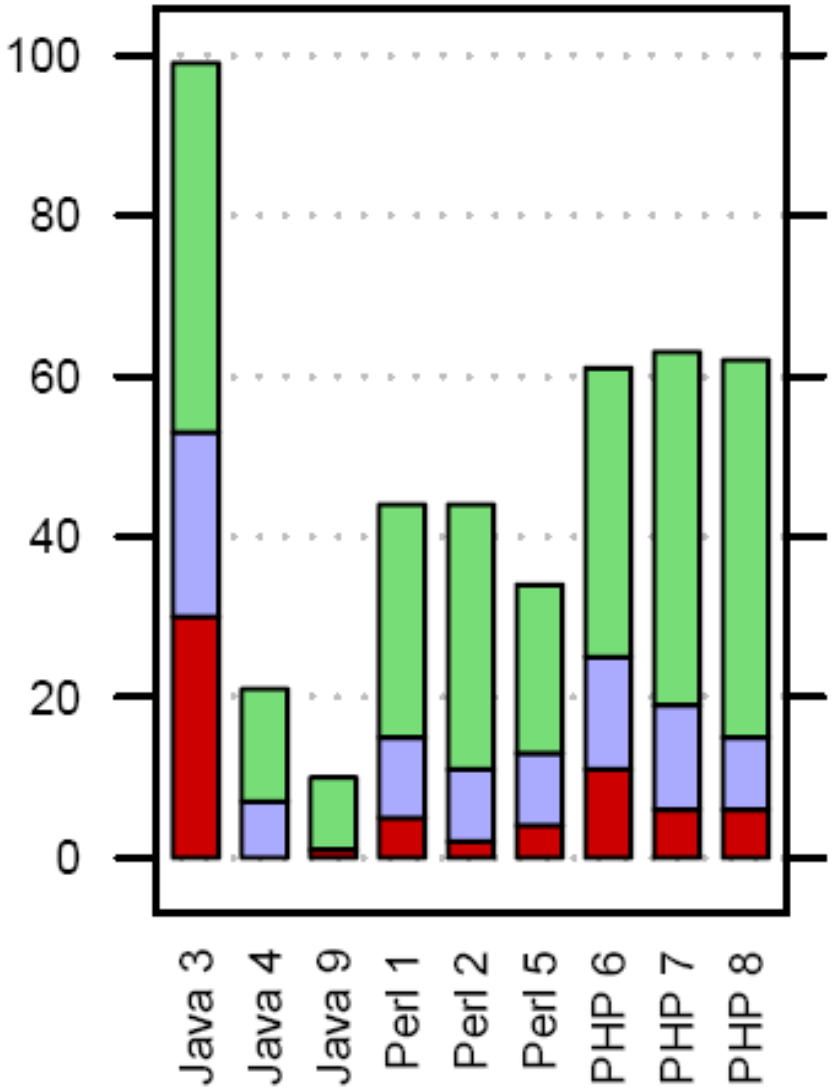
Freie Universität Berlin

- Grounded Theory work using audio/video data is *extremely* laborious
  - initially one minute of raw data often requires two hours of work for the conceptualization
  - even after the coding scheme has been formed it is usually about 10 to 30 minutes

- We have almost finished a coding scheme
- Without even looking, we found that a common PP assumption is wrong:
  - Driver and Observer are not usually on different levels of abstraction

- We have not searched for behavioral patterns yet

Results Plat_Forms (1)
Completeness of solutions

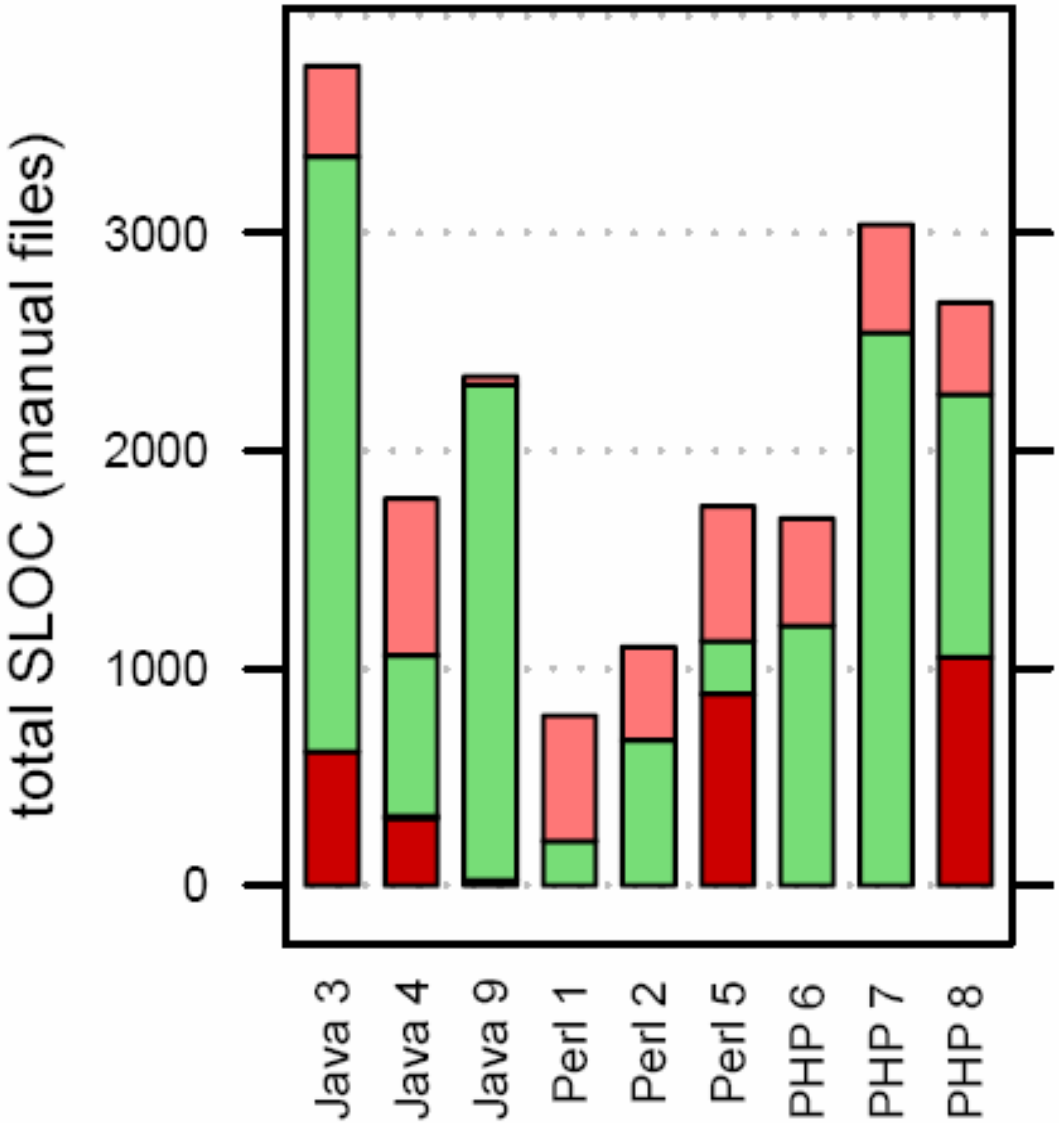


GUI requirements

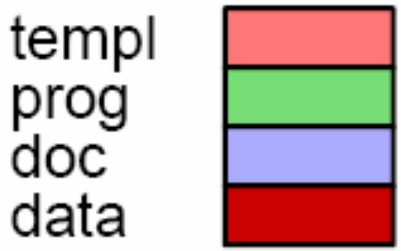
MUST
SHOULD
MAY

Note:
- Team Java 4 was hampered by a huge VMware setup problem for almost a full day
- Team Java 9 used a framework still in alpha development (RAP)

Results Plat_Forms (2)
Size of solution



source lines-of-code



Note:
Further manually written source code resides in modified reused files.

etc.

Results: Plat_Forms

- There are a large number of individual results
  - many of them are Null (i.e., no platform differences found)
  - some are as expected
  - some counter common expectations
    - e.g. PHP solutions were at least as secure as Java solutions
  - some are surprising and new
    - in particular: strong homogeneity among PHP solutions
  - some are even hard to interpret at all
- For details see http://www.plat-forms.org

- This was successful exploratory research:
  - We are no nearer a theory of platform differences than before
  - but we have made a number of sound observations
  - that lead to more specific research questions

Summary

- We have seen two different research areas:
  - understanding Pair Programming
  - finding differences between technology platforms
- that ask widely different research questions:
  - PP: What mechanisms are at work?
    What behavior is advantageous?
    What behavior is problematic?
  - Plat_Forms: Which characteristics emerge due to use of a particular platform?
    How are they different between platforms?
- and suggest rather different research methods:
  - PP: inductive qualitative analysis (Grounded Theory method)
  - Plat_Forms: quasi-experimental direct comparison
- although both streams of research are exploratory.

Thank you!