Why do we know so little about programming languages, and what would have happened if we had known more?

Stefan Hanenberg

University of Duisburg-Essen, Germany

Dynamic Language Symposium 2014
Portland, Orgegon, US

a) Why do we know so little about programming languages?

b) What would have happened if we had known more?

Stefan Hanenberg

University of Duisburg-Essen, Germany

Dynamic Language Symposium 2014
Portland, Orgegon, US

Why do we know so little about programming languages?

What is the goal of programming language (PL) research?

a) PLs let us tell the machine to **<u>do something</u>**

b) PLs let a machine do a **<u>fast</u>** computation

Goals achieved?

a) PLs let us tell the machine to **do something**

- Models of computation (turing-machines, etc.)

b) PLs let a machine do a **fast** computation

- Well, programs run already „quite fast“

Why is there **still** a need for PL research / PL development?

a) PLs let us tell the machine to **do something**

- Let's give people **better means** to tell the machine what to do

b) PLs let a machine do a **fast** computation

- Let's make it **even faster**

The two stories of PL research (simplified)

b) Let's do it faster

- What do people do?
  - Formal models, benchmarking, etc.

a) Let's provide better means for developers

- What do people do?
  - They report on their experience (aka anecdotal evidence)

The two stories of PL research (simplified)

There is nothing wrong with it...

b) Let's do it faster

- What do people do?
  - Formal models, benchmarking, etc.

a) Better means for developers

- What do people do?
  - They report on their experience (aka anecdotal evidence)

The two stories of PL research (simplified)

There is nothing wrong with it...

b) Let's do it faster

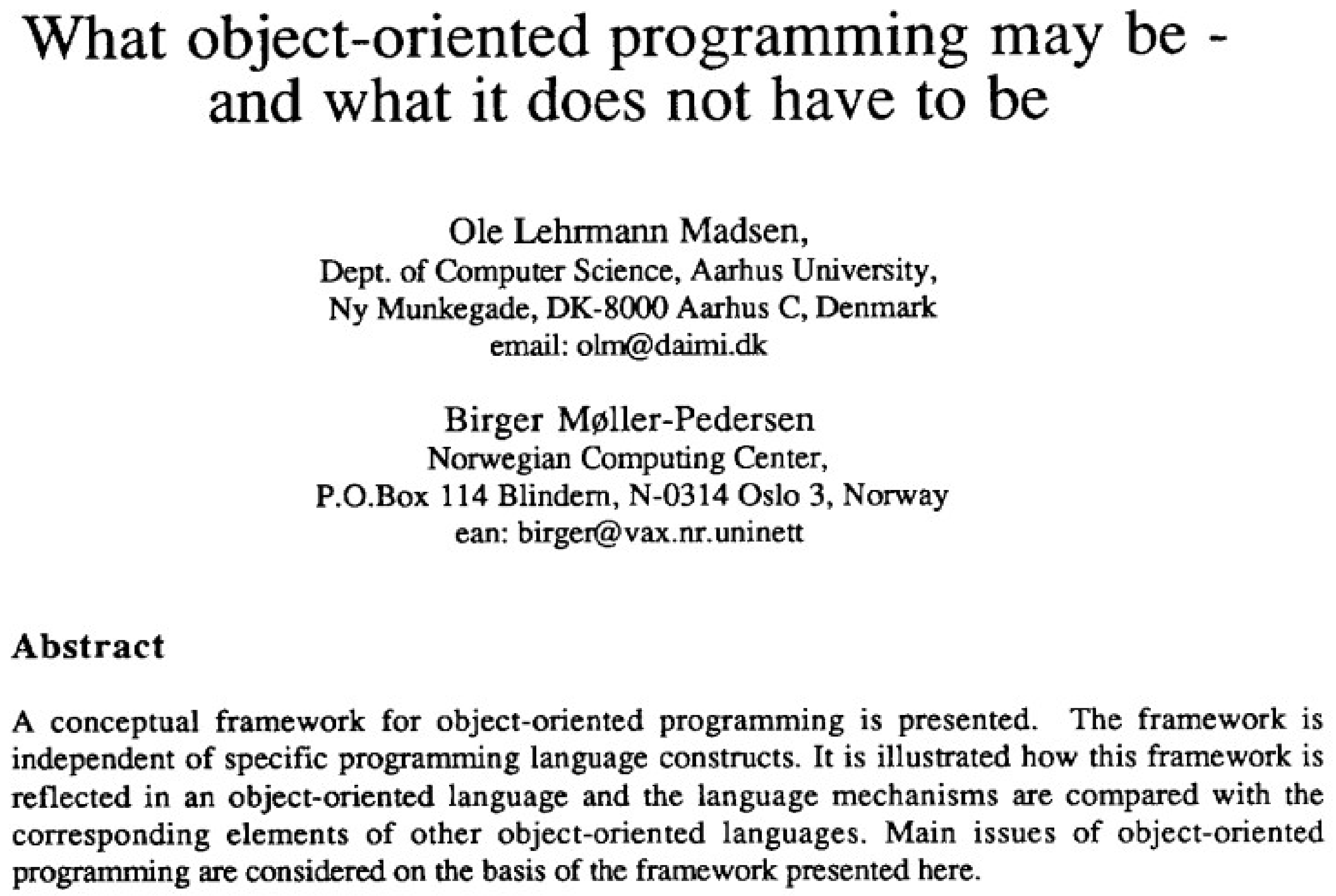
- What do people do?
  - Formal models, benchmarking, etc.

a) Let's provide better means for developers

- What do people do?
  - They report on their experience (aka anecdotal evidence)

Statements from literature

Statements from literature (1)



What object-oriented programming may be - and what it does not have to be

Ole Lehrmann Madsen,
Dept. of Computer Science, Aarhus University,
Ny Munkegade, DK-8000 Aarhus C, Denmark
email: olm@daimi.dk

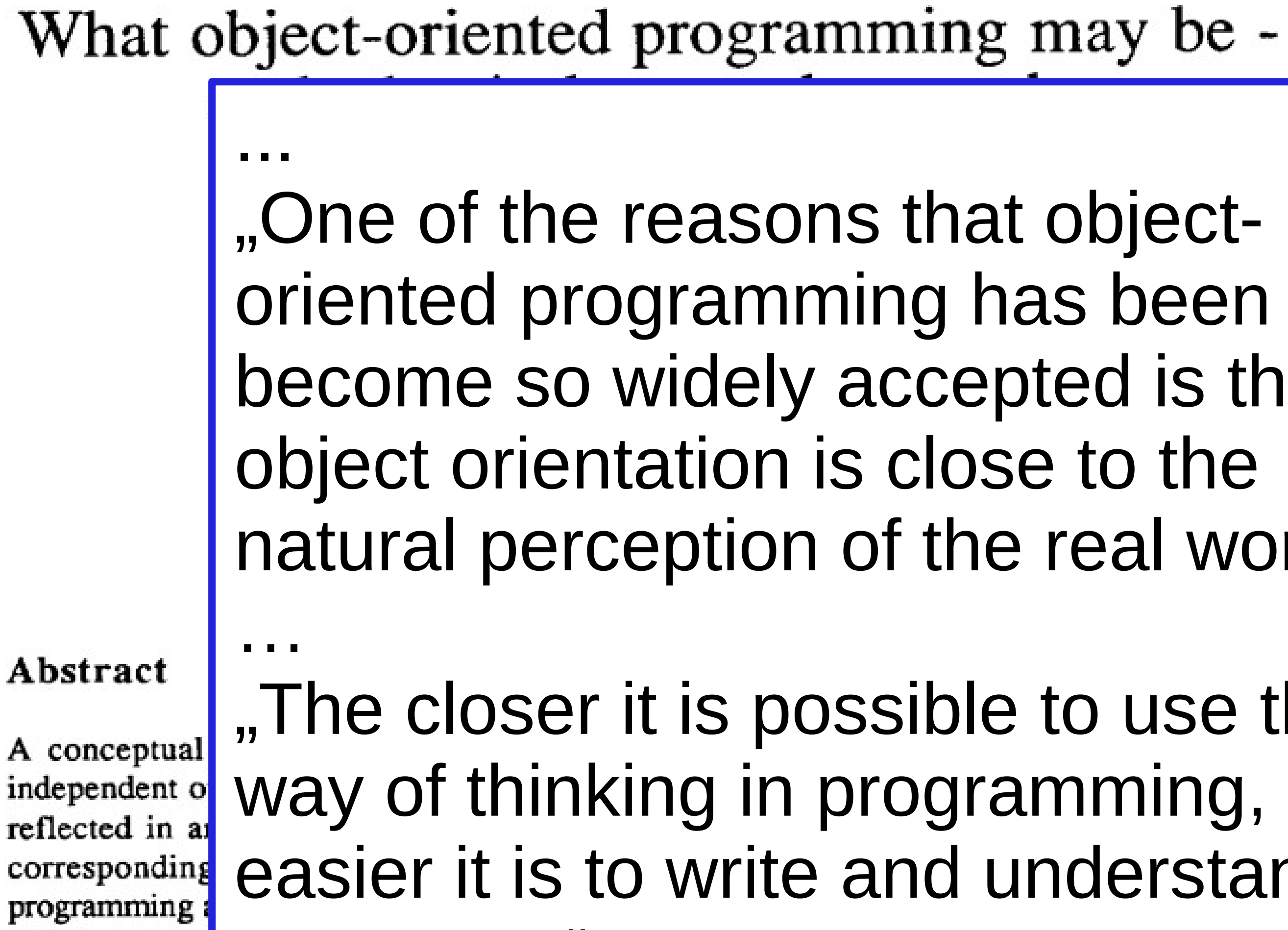
Birger Møller-Pedersen
Norwegian Computing Center,
P.O.Box 114 Blindern, N-0314 Oslo 3, Norway
ean: birger@vax.nr.uninett

Abstract

A conceptual framework for object-oriented programming is presented. The framework is independent of specific programming language constructs. It is illustrated how this framework is reflected in an object-oriented language and the language mechanisms are compared with the corresponding elements of other object-oriented languages. Main issues of object-oriented programming are considered on the basis of the framework presented here.

[ECOOP 1990]

Statements from literature (1)



What object-oriented programming may be -

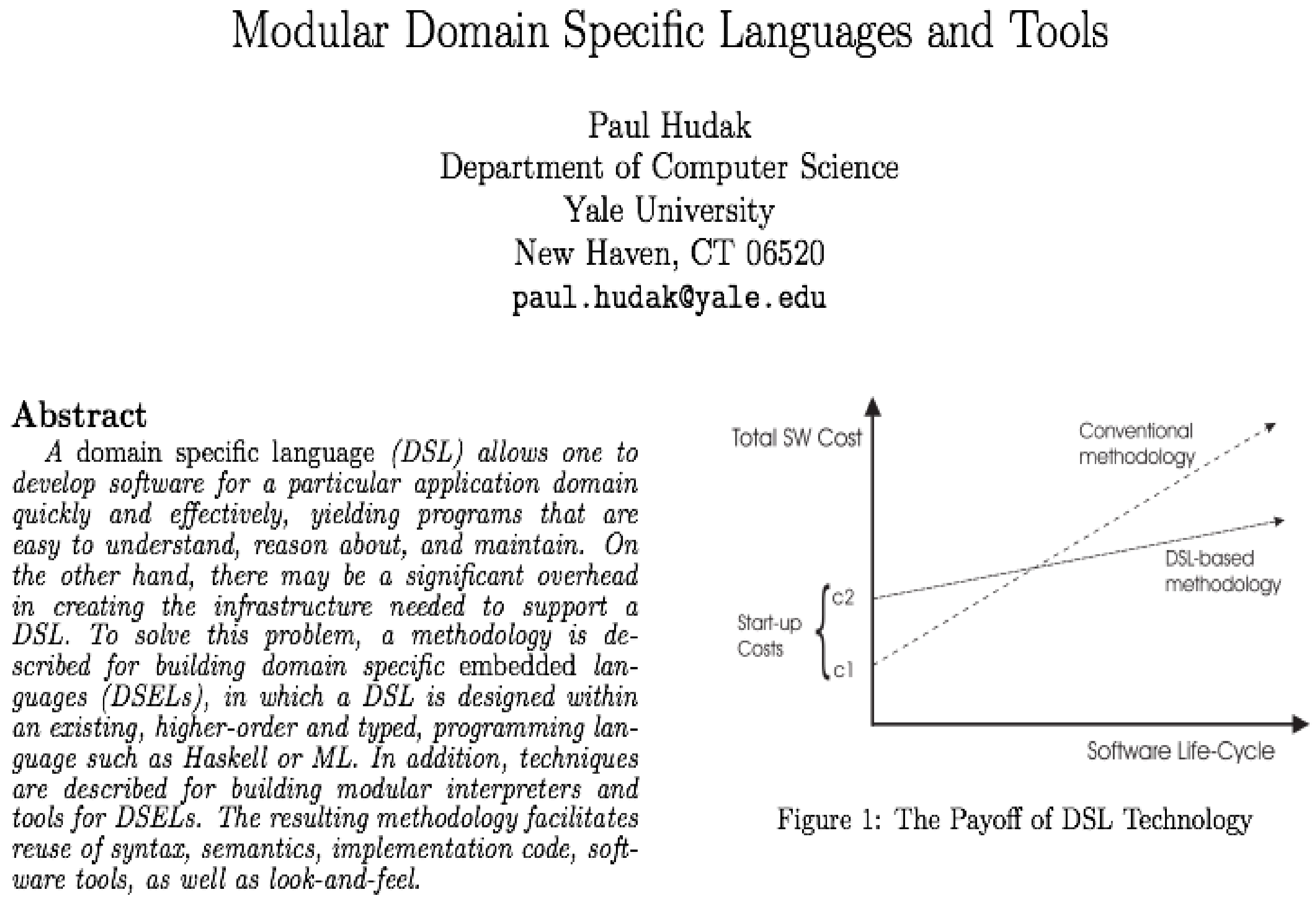
Abstract

A conceptual
independent o
reflected in a
correspondin
programming

...
„One of the reasons that object-oriented programming has been become so widely accepted is that object orientation is close to the natural perception of the real world.“
…
„The closer it is possible to use this way of thinking in programming, the easier it is to write and understand programs.“

90]

Statements from literature (2)



Modular Domain Specific Languages and Tools

Paul Hudak
Department of Computer Science
Yale University
New Haven, CT 06520
`paul.hudak@yale.edu`

Abstract

A domain specific language (DSL) allows one to develop software for a particular application domain quickly and effectively, yielding programs that are easy to understand, reason about, and maintain. On the other hand, there may be a significant overhead in creating the infrastructure needed to support a DSL. To solve this problem, a methodology is described for building domain specific embedded *languages (DSELs), in which a DSL is designed within an existing, higher-order and typed, programming language such as Haskell or ML. In addition, techniques are described for building modular interpreters and tools for DSELs. The resulting methodology facilitates reuse of syntax, semantics, implementation code, software tools, as well as look-and-feel.*

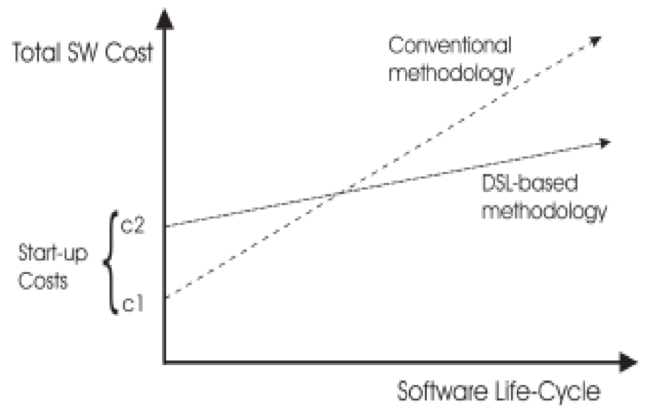
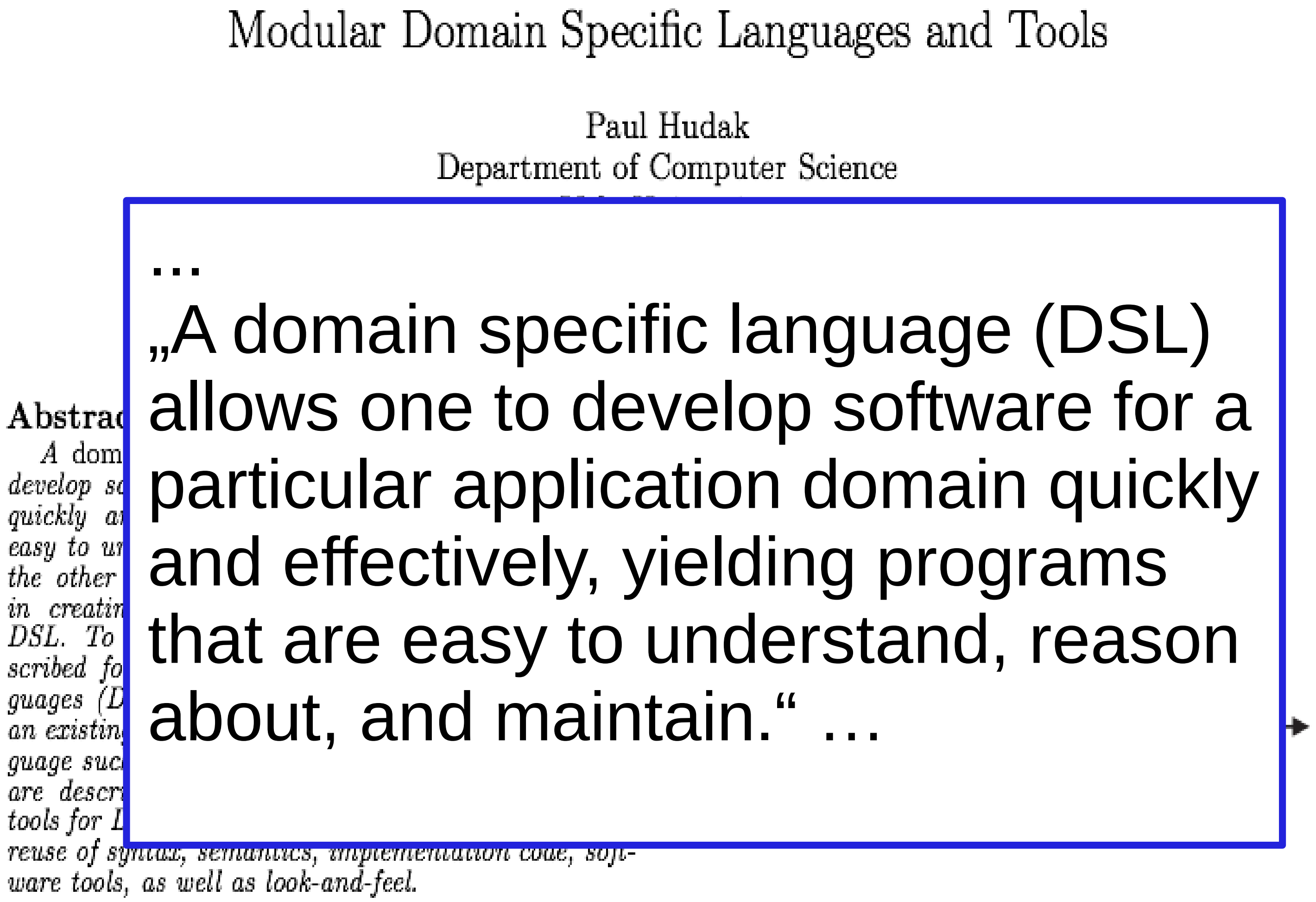


Figure 1: The Payoff of DSL Technology

[ICSR '98]

Statements from literature (2)



Modular Domain Specific Languages and Tools

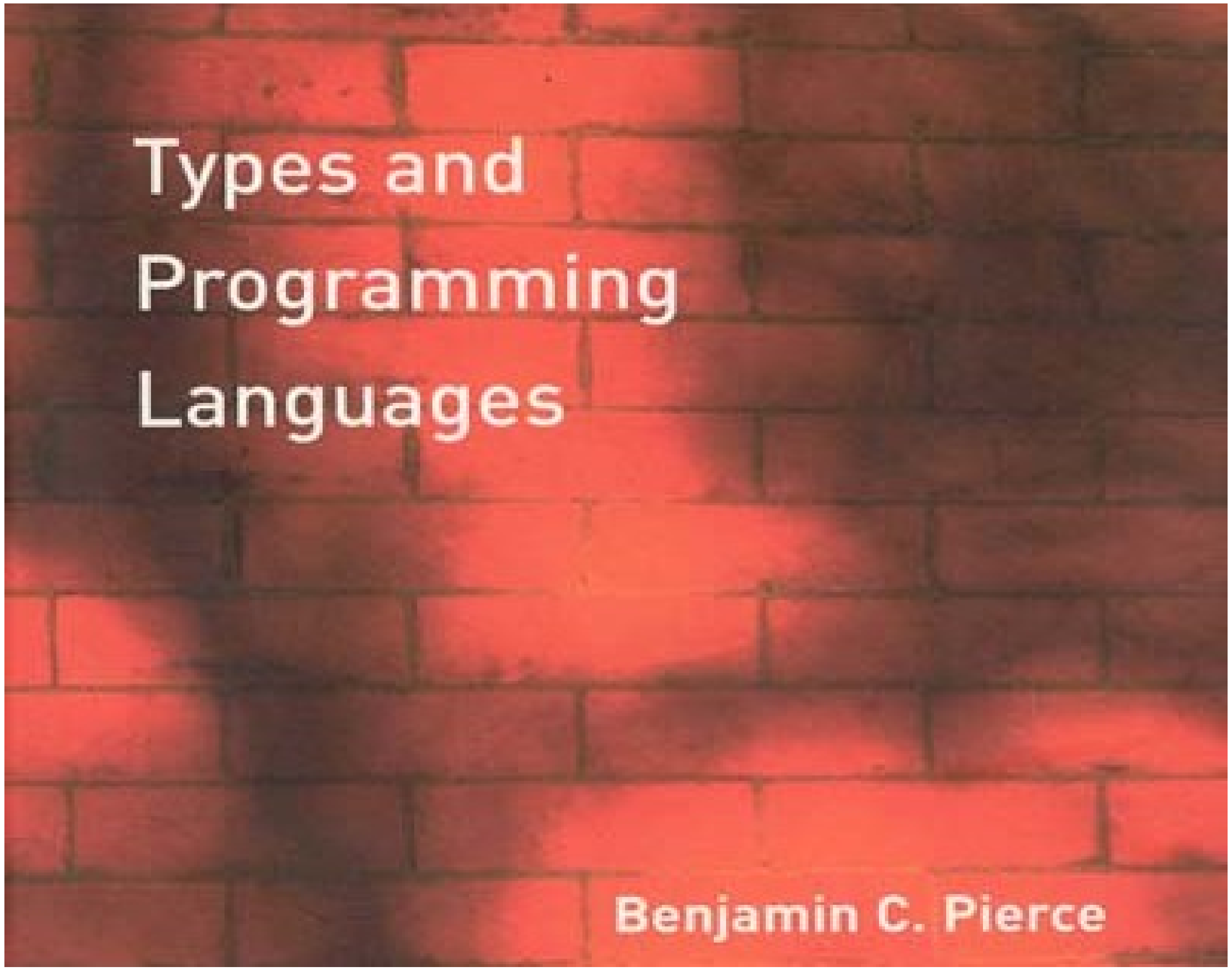
Paul Hudak
Department of Computer Science

Abstrac
A dom
develop s
quickly a
easy to u
the other
in creatin
DSL. To
scribed fo
guages (D
an existin
guage suc
are descr
tools for D
reuse of syntax, semantics, implementation code, soft-
ware tools, as well as look-and-feel.

...
„A domain specific language (DSL) allows one to develop software for a particular application domain quickly and effectively, yielding programs that are easy to understand, reason about, and maintain.“ …

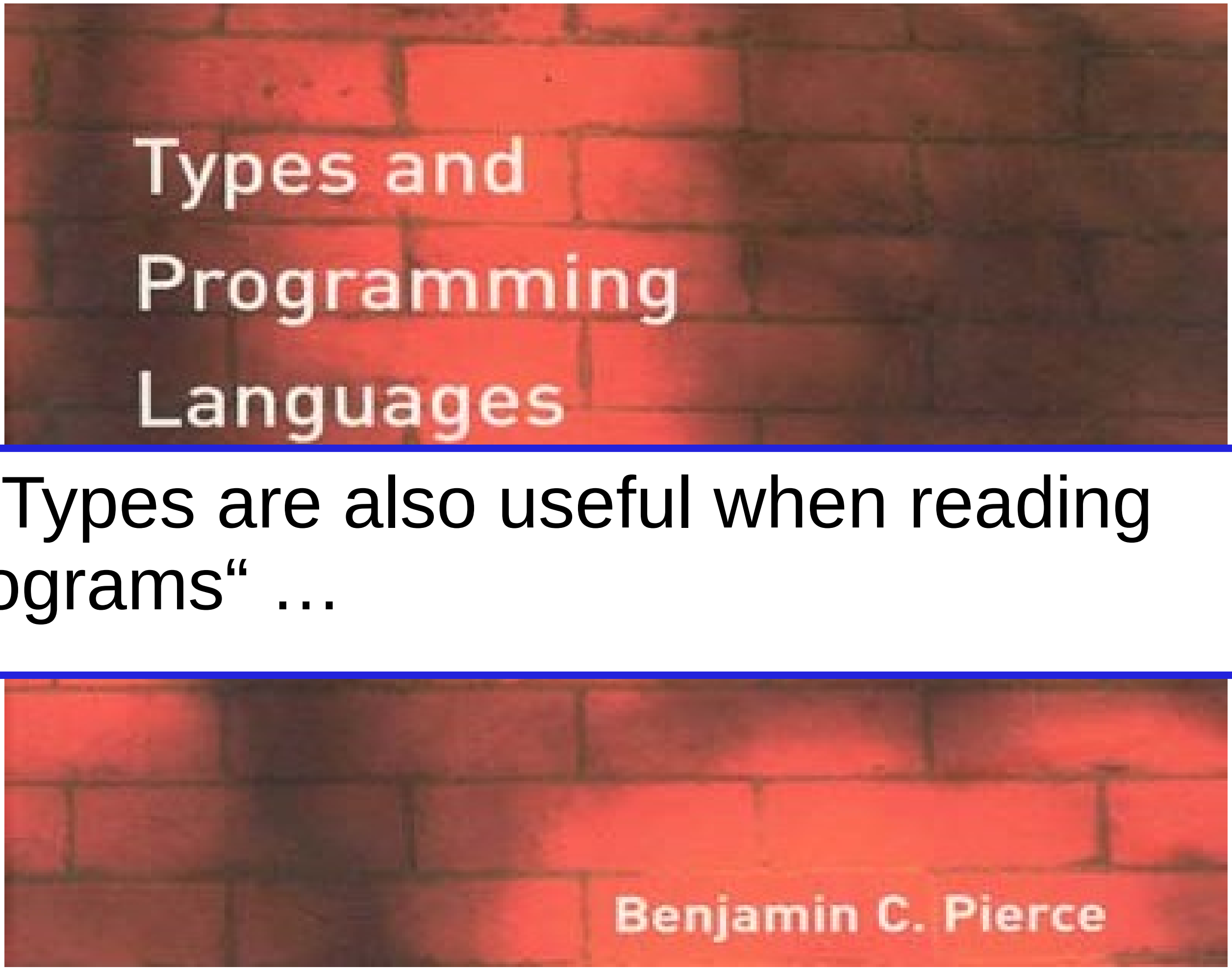
[ICSR '98]

Statements from literature (3)



MIT Press 2002

Statements from literature (3)



...„Types are also useful when reading programs“ ...

MIT Press 2002

What's common in previous statements?

- They refer to human behavior
- Human behavior is essential for the argumentation

- Human behavior is being made pausible
- Human behavior is not tested

- **...and...**

What's common in previous statements?

- They refer to human behavior
- Human behavior is essential for the argumentation

- Human behavior is being made pausible
- Human behavior is not tested

- **The works do not refer to any other work that gives evidence for validity of statements**

It is **not** a singular phenomena

„*A total of 1.1% of papers both had evidence in WWC categories 1 or 2 and were about language design [for PPIG]*“

...14.3% for PLATEAU, 16.7% for ESP...

[Stefik et al ICPC'14]

(average WWC scores between .2 and .7)

What's the problem?

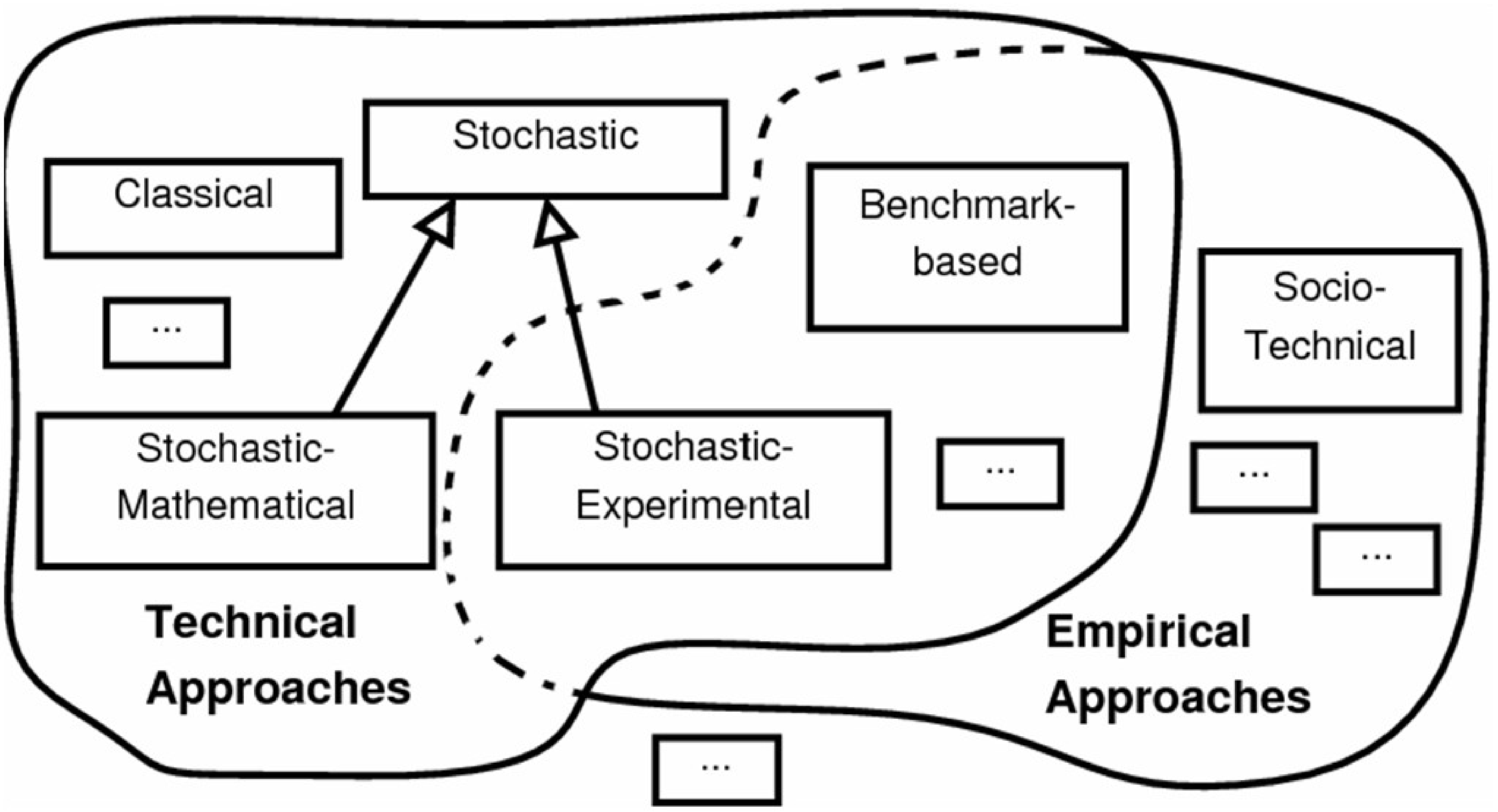
- Neither assumptions nor conclusions are tested
- Risk that ...
  - … some(?) PL tools never ever showed the expected influence on developers
  - … some(?) statements in SE literature are wrong
  - … some(?) of our tools are useless. Which ones?
  - … some(?) of our tools are harmful. Which ones?

Conclusion

- Human characteristics and behavior often used to argue for or against some techniques
- No **known techniques** for testing human behavior are applied

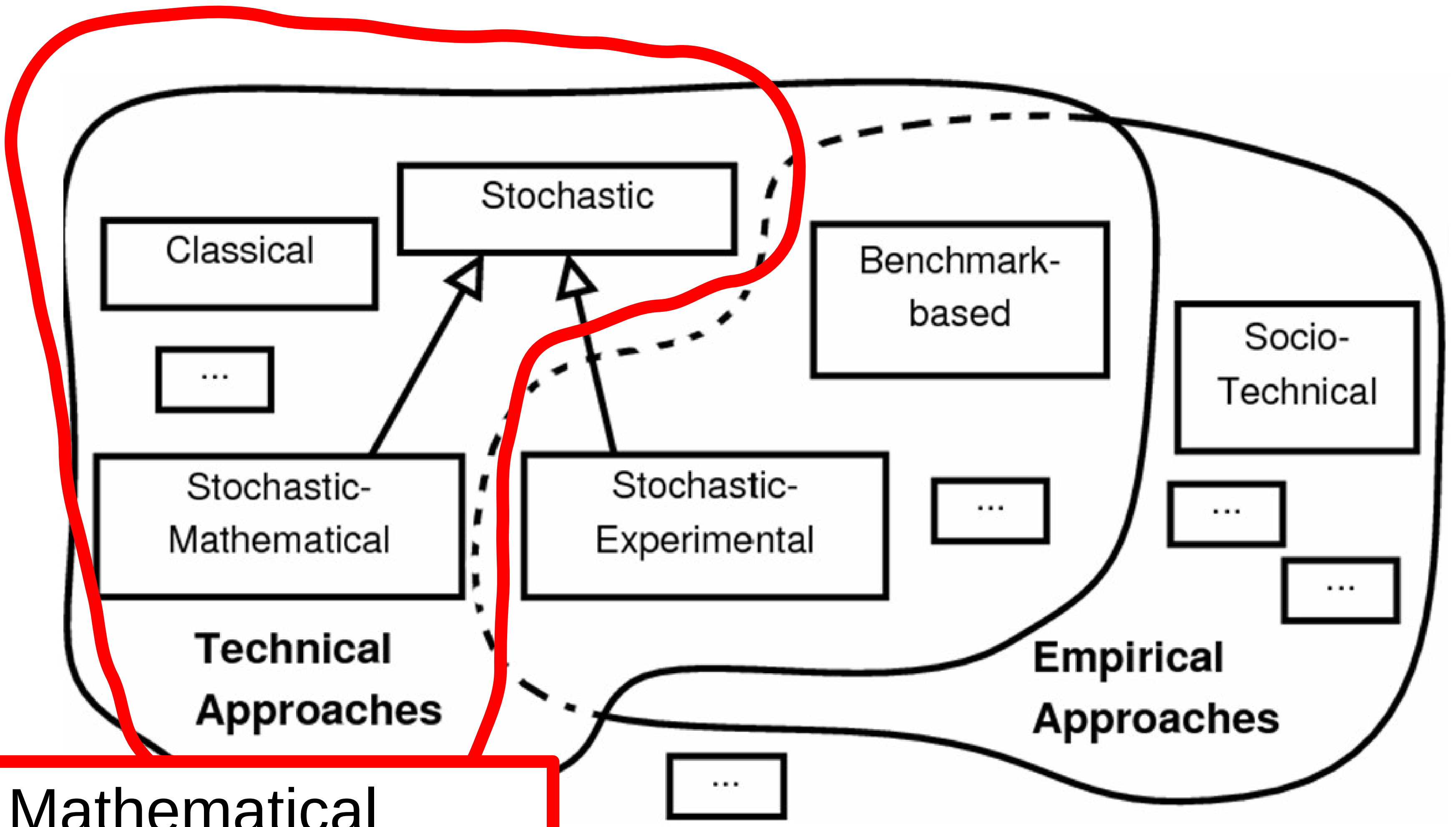
Why?

SE Research Methods & Education



[Hanenberg, Faith, Hope, Love, Onward'10]

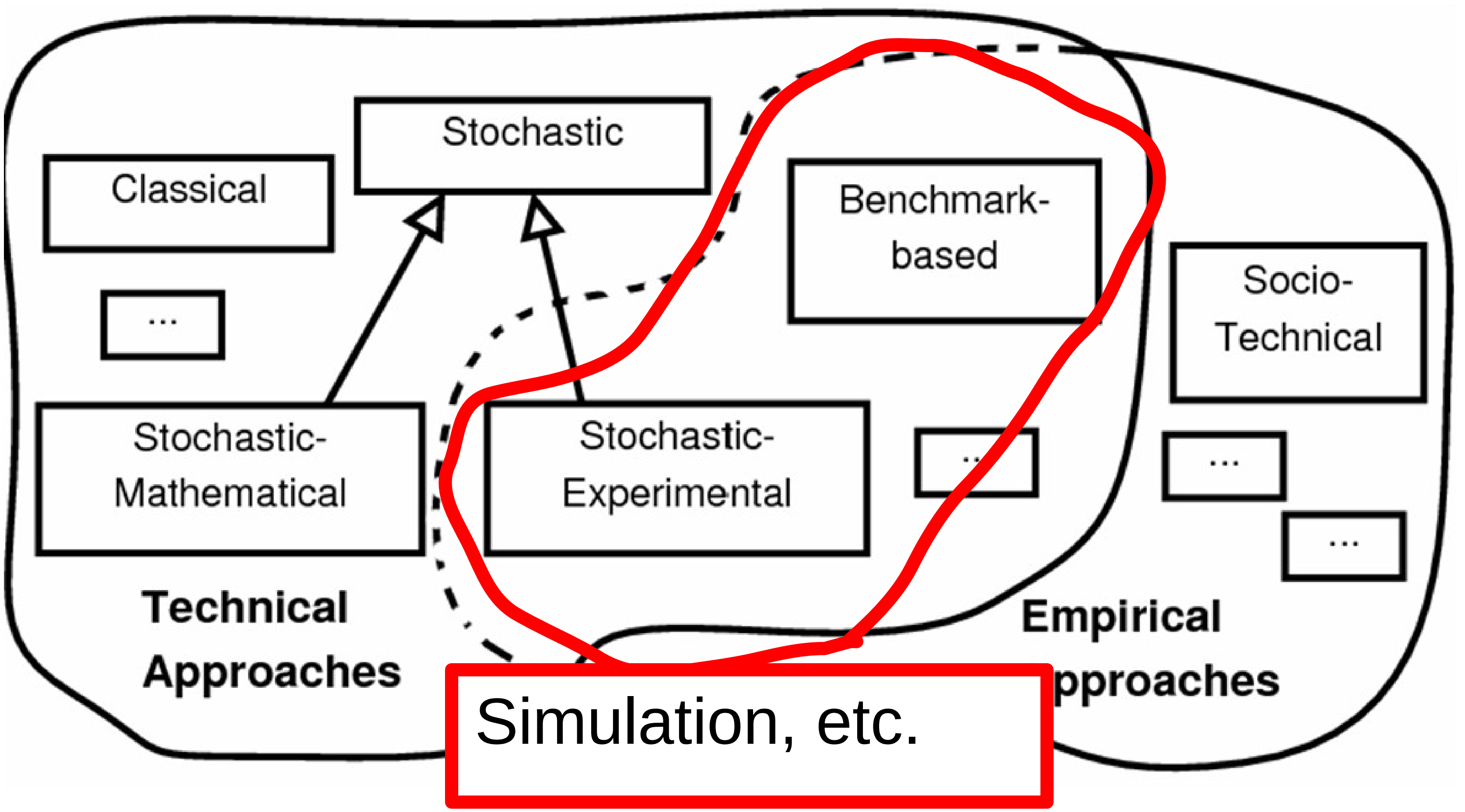
SE Research Methods & Education



Mathematical education

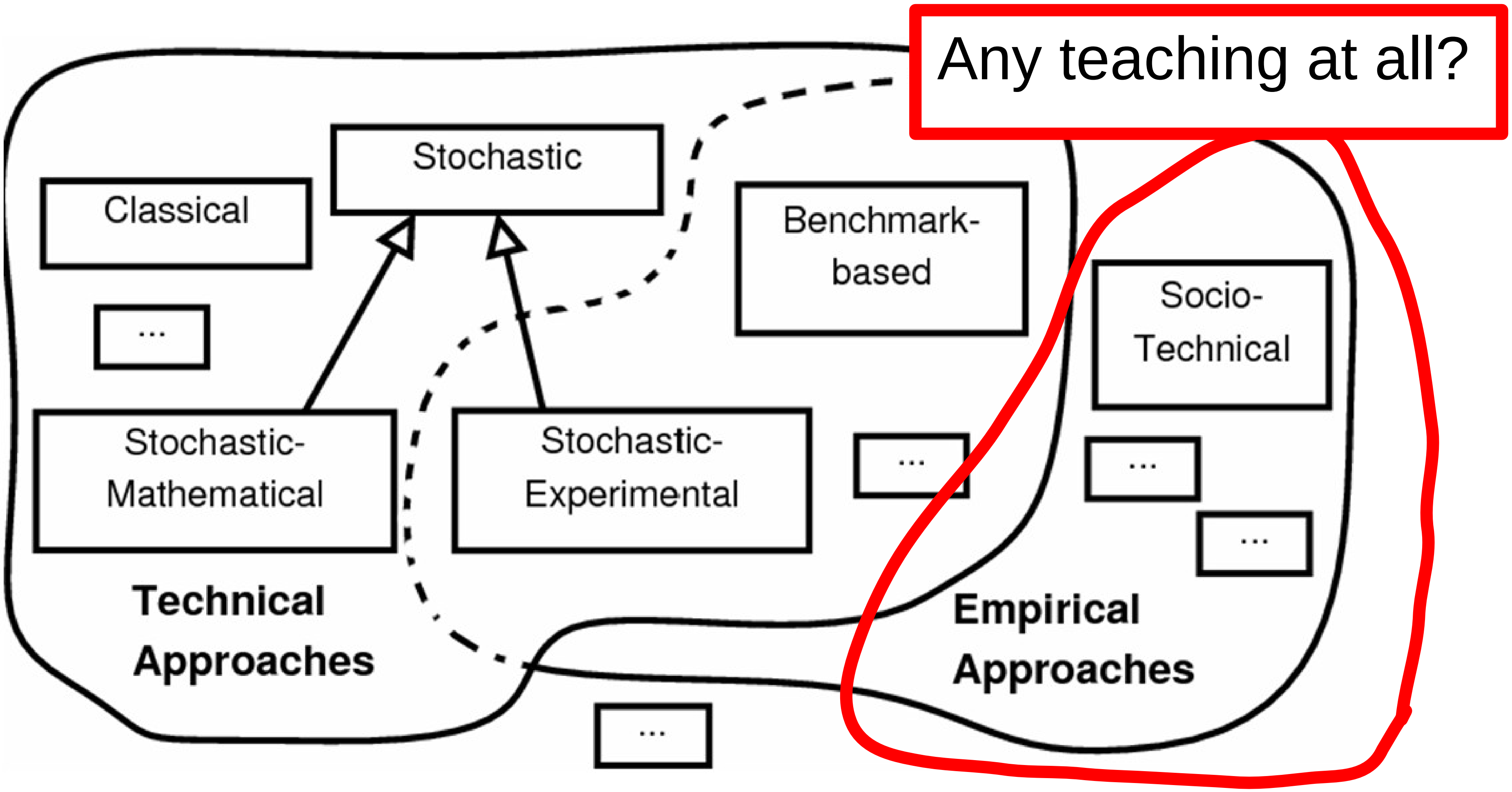
[Hanenberg, Faith, Hope, Love, Onward'10]

SE Research Methods & Education



[Hanenberg, Faith, Hope, Love, Onward'10]

SE Research Methods & Education



[Hanenberg, Faith, Hope, Love, Onward'10]

Conclusion

Human methods not taught

=> No human methods applied

=> No data available on human behavior in PL usage

=> Statements about human behavior are speculative

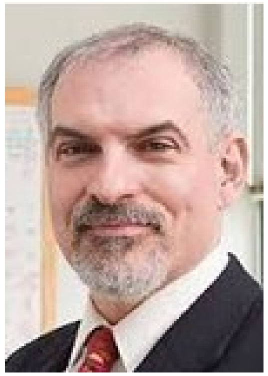
=> Missing knowledge in PL usage and usability

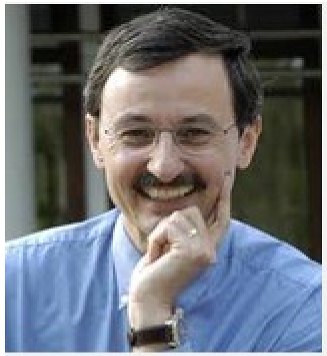
This is not completely true....

- there are people that apply human-centered methods ...

This is not completely true....

- there are people that apply human-centered methods ...





- … but they are still relatively few

This is not completely true....

- there are books that introduce into human-centered methods ...

Lutz Prechelt
Kontrollierte Experimente in der Softwaretechnik
EXPERIMENTATION IN SOFTWARE ENGINEERING An Introduction
Claes Wohlin
Per Runeson
Martin Höst
Basics of Software Engineering Experimentation
Natalia Juristo and Ana M. Moreno
Foreword by Shari Lawrence Pfleeger
Kluwer Academic Publishers
SOFTWARE PSYCHOLOGY
HUMAN FACTORS IN COMPUTER AND INFORMATION SYSTEMS
BEN SHNEIDERMAN
Little, Brown Computer Systems Series
Barry Boehm · Hans Dieter Rombach
Marvin V. Zelkowitz (Eds.)
Foundations of Empirical Software Engineering
The Legacy of Victor R. Basili
Springer
The Fraunhofer IESE Series on Software Engineering
A Handbook of Software and Systems Engineering
Empirical Observations, Laws and Theories
Albert Endres
Dieter Rombach

- … but they are still relatively few

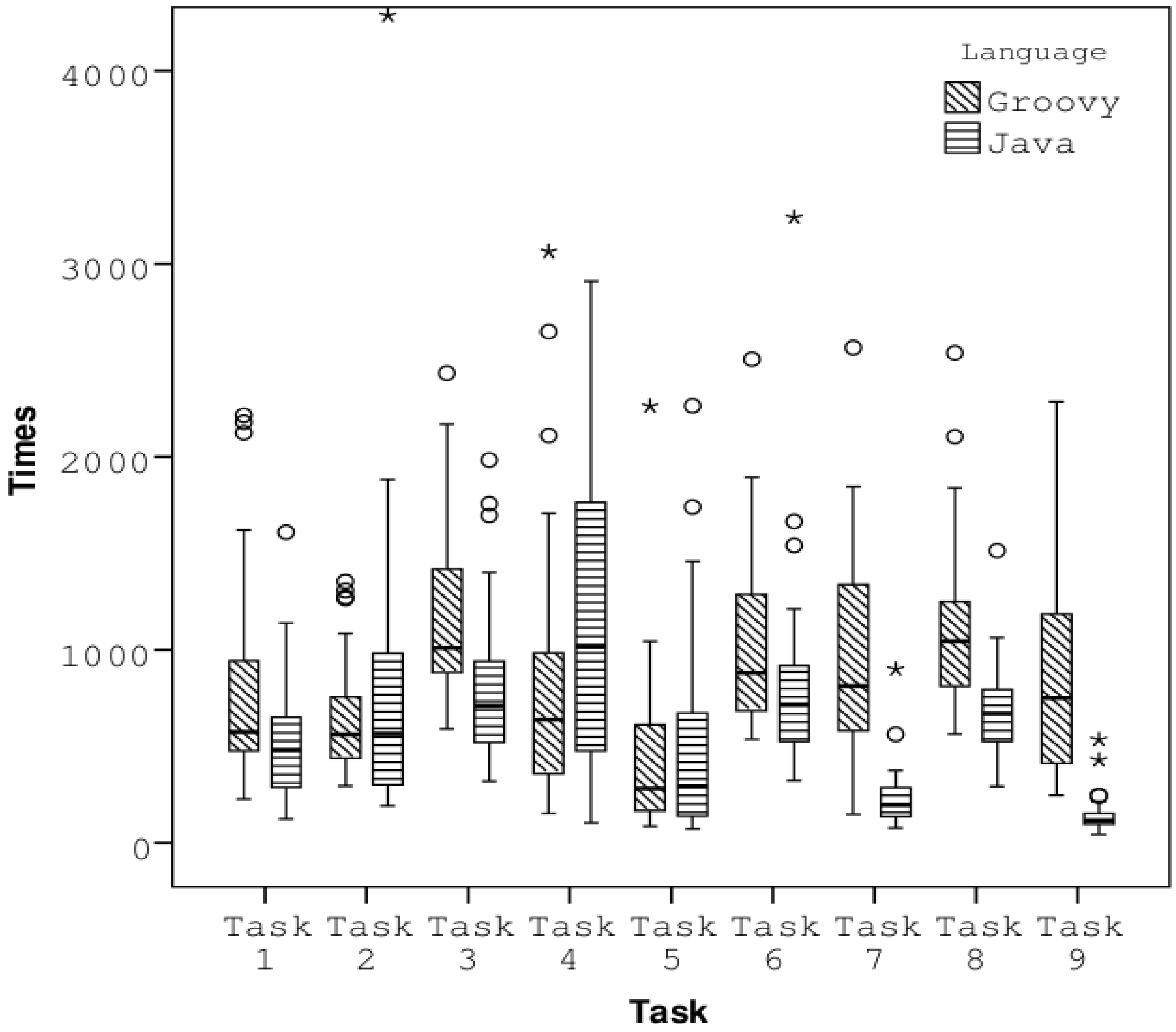
Example of applied human-centered methods: Experiment series on type systems

Experiment 5: Types & APIs [ICPC'12]

- Idea: Static type systems help when using an undocumented API
- Experiment
  - Java / Groovy as Pls, Development time as measurement
  - 9 programming tasks
    - 2 tasks: fix semantic error / 2 tasks: fix type error / 5 tasks: use API classes
  - 33 subjects (mainly students)
  - Within-subject design (2 groups)
- Result
  - Positive effect for 6/9 tasks
    - No effect on fixing semantic error
    - Positive effect on fixing type error
    - Mostly (4/5) positive effect on using API classes

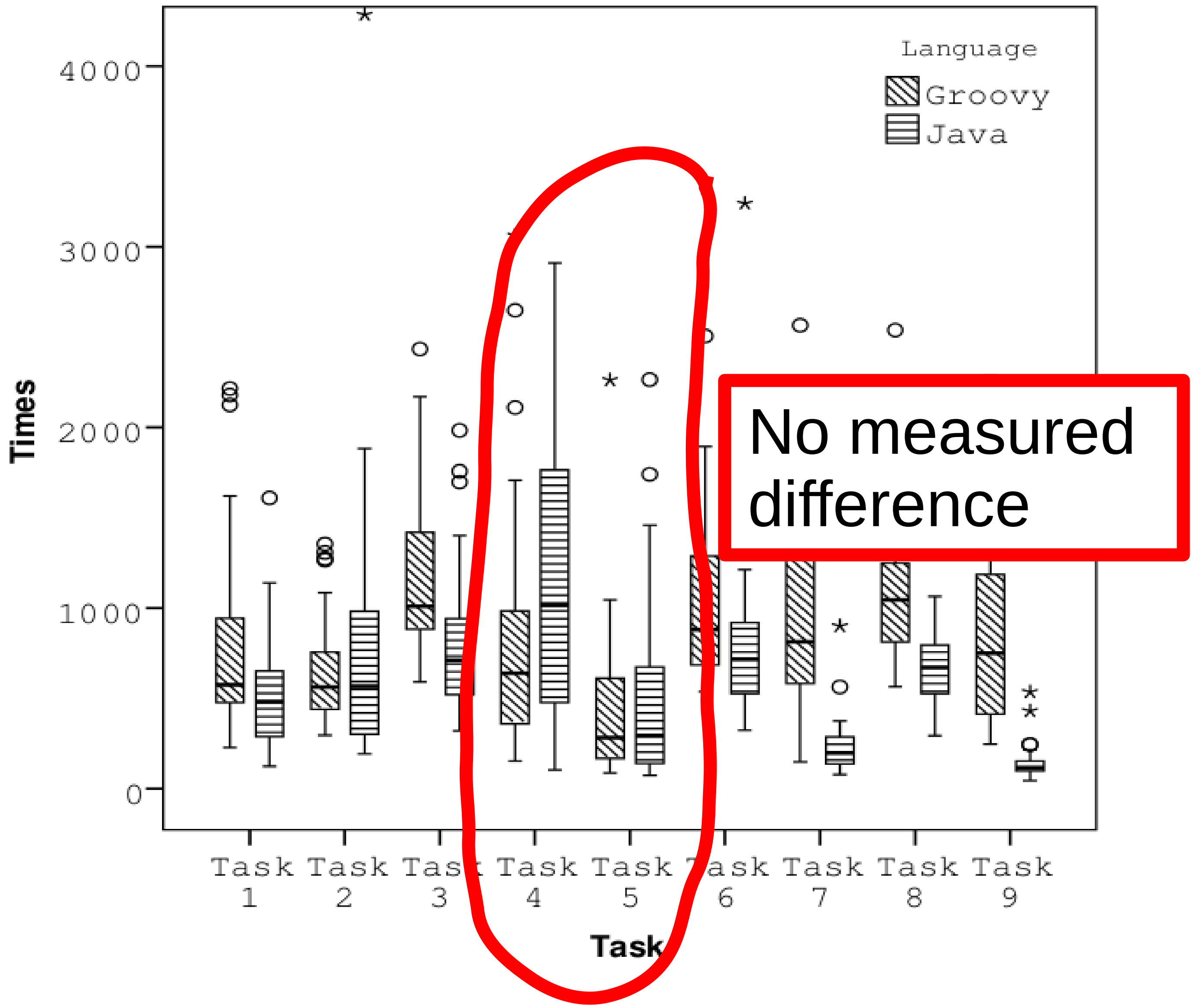
Experiment 5: Types & APIs [ICPC'12]

- Task 4,5: Semantic errors
- 1,2,3,6,8: New class usage
- 7, 10: Type errors



Experiment 5: Types & APIs [ICPC'12]

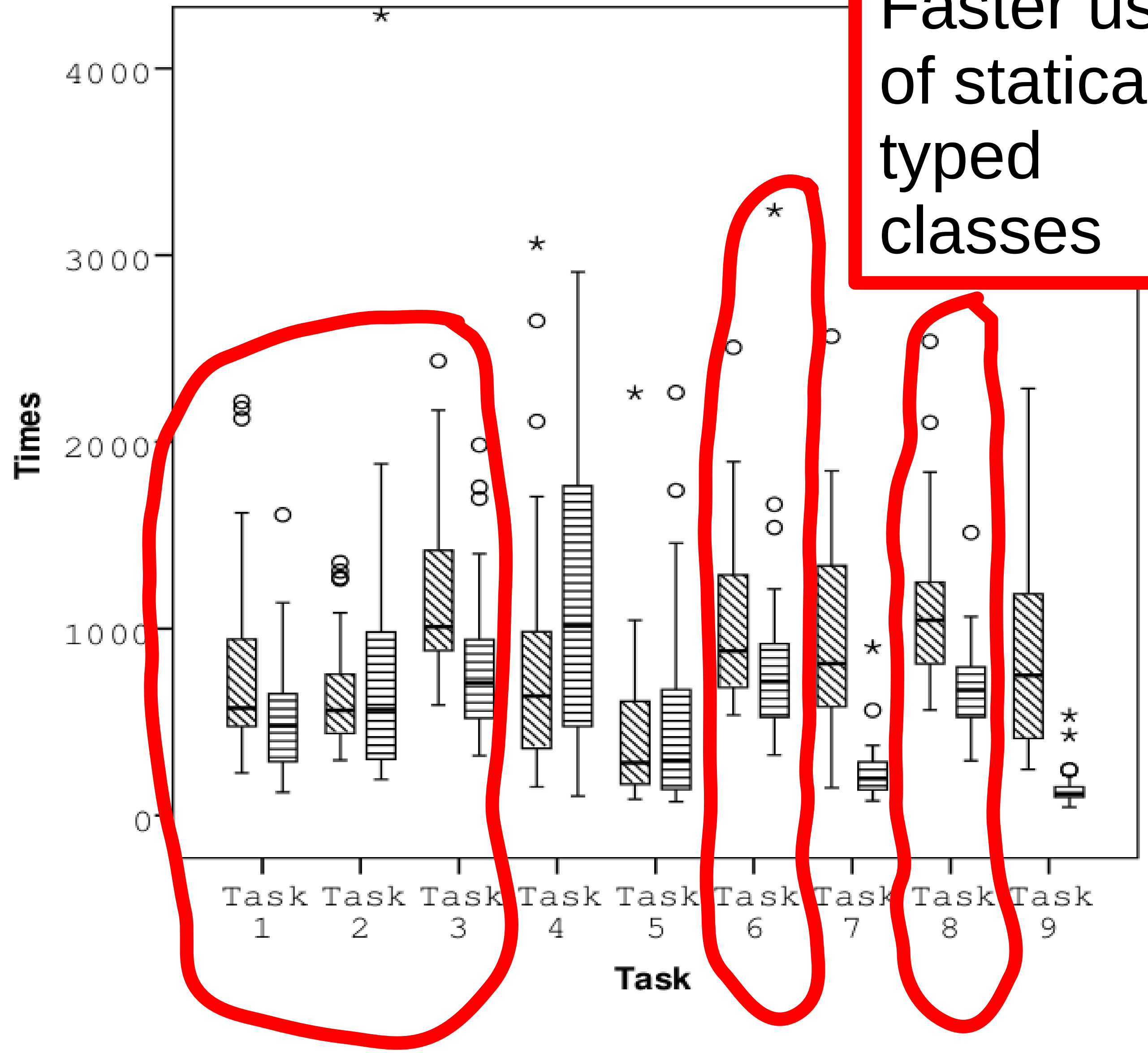
- Task 4,5: Semantic errors
- 1,2,3,6,8: New class usage
- 7, 10: Type errors



No measured difference

Experiment 5: Types & APIs [ICPC'12]

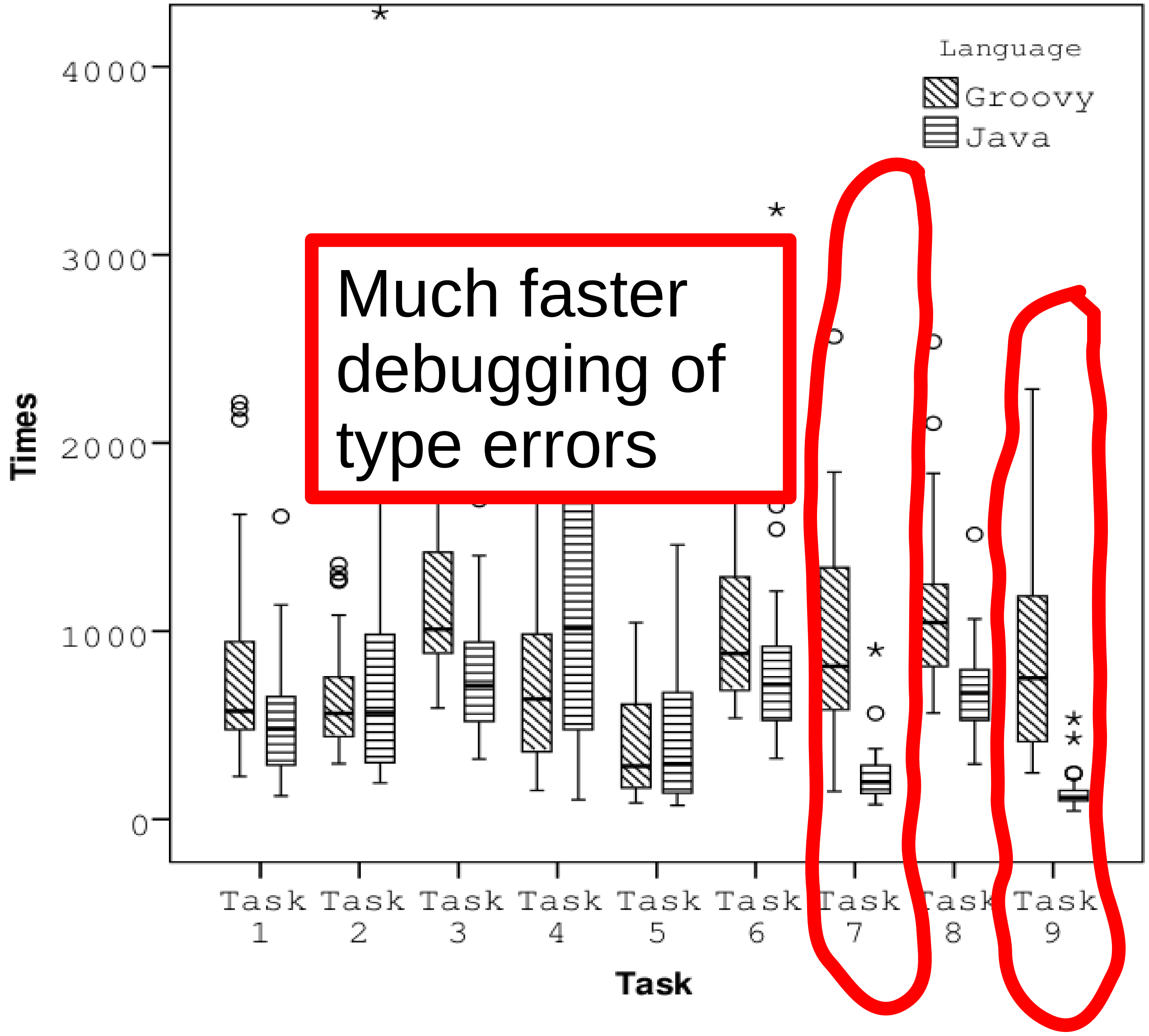
- Task 4,5: Semantic errors
- 1,2,3,6,8: New class usage
- 7, 10: Type errors



Faster use of statically typed classes

Experiment 5: Types & APIs [ICPC'12]

- Task 4,5: Semantic errors
- 1,2,3,6,8: New class usage
- 7, 9: Type errors



Much faster debugging of type errors

Experiment 5: Types & APIs [ICPC'12]

- Potential problems
  - Artificially constructed API
    - parameter names do not reflect on type names (but names were chosen from the domain)
    - Is it repesentative?
  - Artificially constructed environment
  - Artificial programming tasks
  - Java type system
- <u>Maybe we measured something else</u>
  - „*Existence of type annotations in the code help....no matter whether they are statically type checked or not*“
- <u>Maybe „in the wild“ positive effect of static type system „vanishs“</u>
  - There is no generalizability

Results so far....

It looks like (Java-like) static type system (in Java-like languages) really help in development!

Tested Statements and Results (1)

Naive Experiment: [OOPSLA'10]
Dynamic Type System are great....almost...

Do type casts matter? [DLS'11]
Not really.

Are dynamic TS as quick for fixing type errors as static TS?
No, not even close! But no difference for semantic errors.
[unpublished'11, ICPC'12]

Tested Statements and Results (2)

Are statically typed APIs faster to use? [OOPSLA'12, ICPC'12]
Yes

Is the previous finding only a matter of syntax? [AOSD'13]
Yes, but in case there is an error in the (unchecked) type it is worse than having no type declaration at all!

Does documentation compensate the positive effect of static types?[ICSE'14] No.

Tested Statements and Results (3)

Do generics really help? [OOPSLA'13]
Yes, if they occur in API interface. No, if application has additional constraints because of generics.

Do current IDEs (for dynamic TSs) compensate the previous measured positive effect of static types? [ICPC'14]
No

Can code completion in dynamically typed languages compensate the benefit of statically types PLs? [..just finished..]
No

Results so far....

It looks like (Java-like) static type system (in Java-like languages) really help in development!

...but we still find exceptions where it is the opposite...
*[interaction effects task*TS in almost all experiments]...*

Ok, good starting point

- ...so far approximately 80% of all existing controlled trials on type system

- BUT ...

Ok, good starting point

- ...so far approximately 80% of all existing controlled trials on type systems

- **BUT we still have to learn / to do a lot**
  - Still very few experiments (not so many replications)
  - How do different tasks differ?
  - How do subjects differ?
  - How to programming styles differ?
  - Standard procedures for experimental design / analysis?
  - Community agreements on even most trivial things such as alpha level?

Believe of Empirical Researchers

- The more experiments are available, the more knowledge we have
- The more knowledge we have, the better are our decisions
- The better our decisions are, the better is our resulting software (or at least less expensive)

Why do we know so little about programming languages and

what if we had known more?

Thought Experiments

- Starting point
  - Empirical researchers believe that empirical knowledge would change the way how

    a) people adopt a language
    b) how people create languages

Thought Experiment 1

- Assume the knowledge on type systems would have been known in 1984...

  ...would this have changed Smalltalk, Lisp, etc?

Thought Experiment 1

- Assume the knowledge on type systems would have been known in 1984...

  ...would this have changed Smalltalk, Lisp, Basic, etc?

  - **….of course not!!!!**
    (just 10 experiments so far, relationship between type system and reflection unclear, etc.)

  - ...but maybe StrongTalk would have appeared earlier

Thought Experiment 2

- Assume the knowledge on PLs from 2044 would be available today
  - ….and let's assume that empirical research is massively applied in the next 30 years...

- How do we currently adopt new languages?

PL Adoption

[Meyerovich, Rabkin, OOPSLA'13]

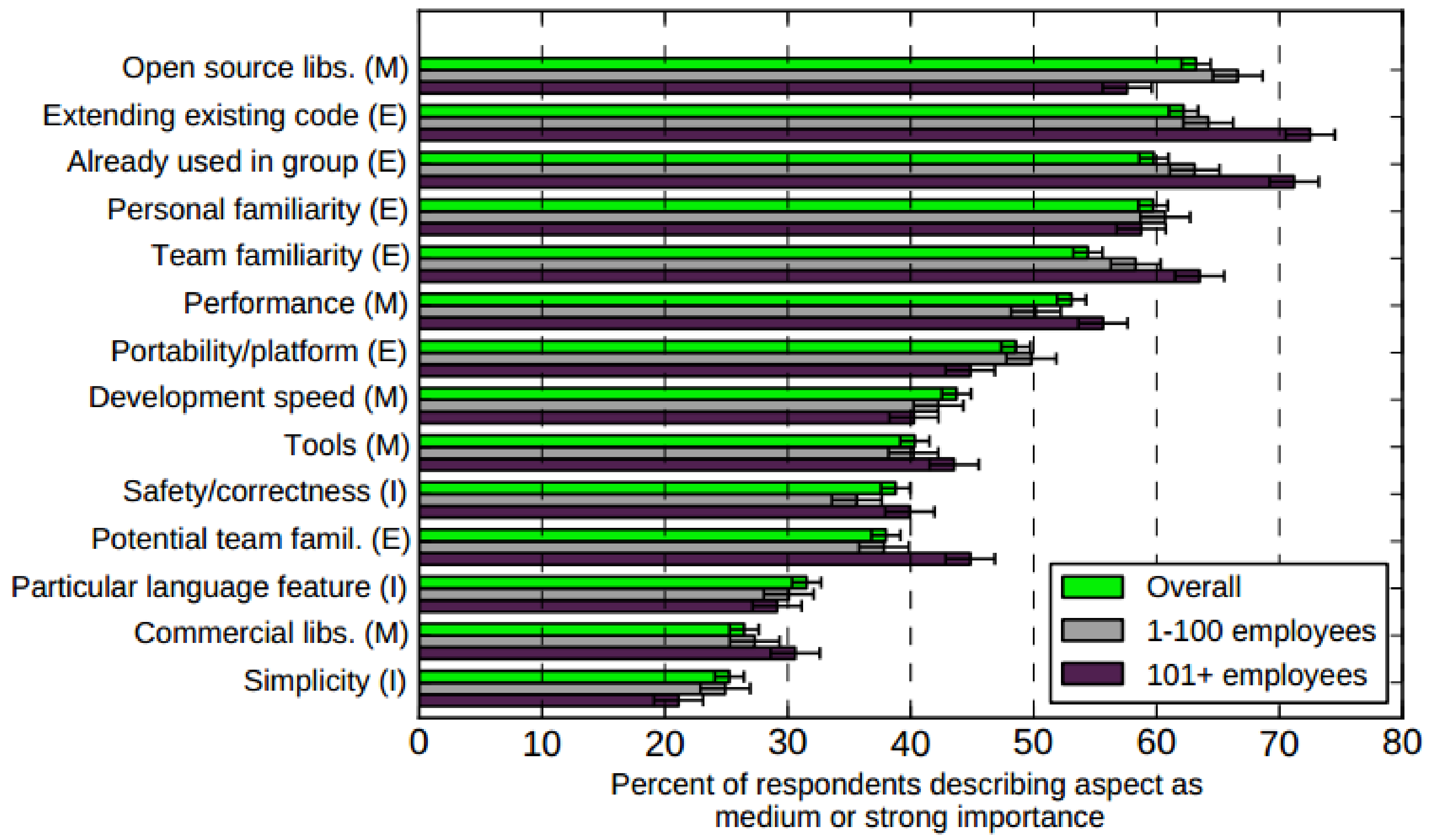


Figure 5: **Importance of different factors when picking a language**. Self-reported for every respondent's last project. Bars show standard error. E = Extrinsic factor, I = Intrinsic, M = Mixed. Shows results broken down by company size for respondents describing a work project and who indicated company size. (Slashdot, n = 1679)

PL Adoption

[Meyerovich, Rabkin, OOPSLA'13]

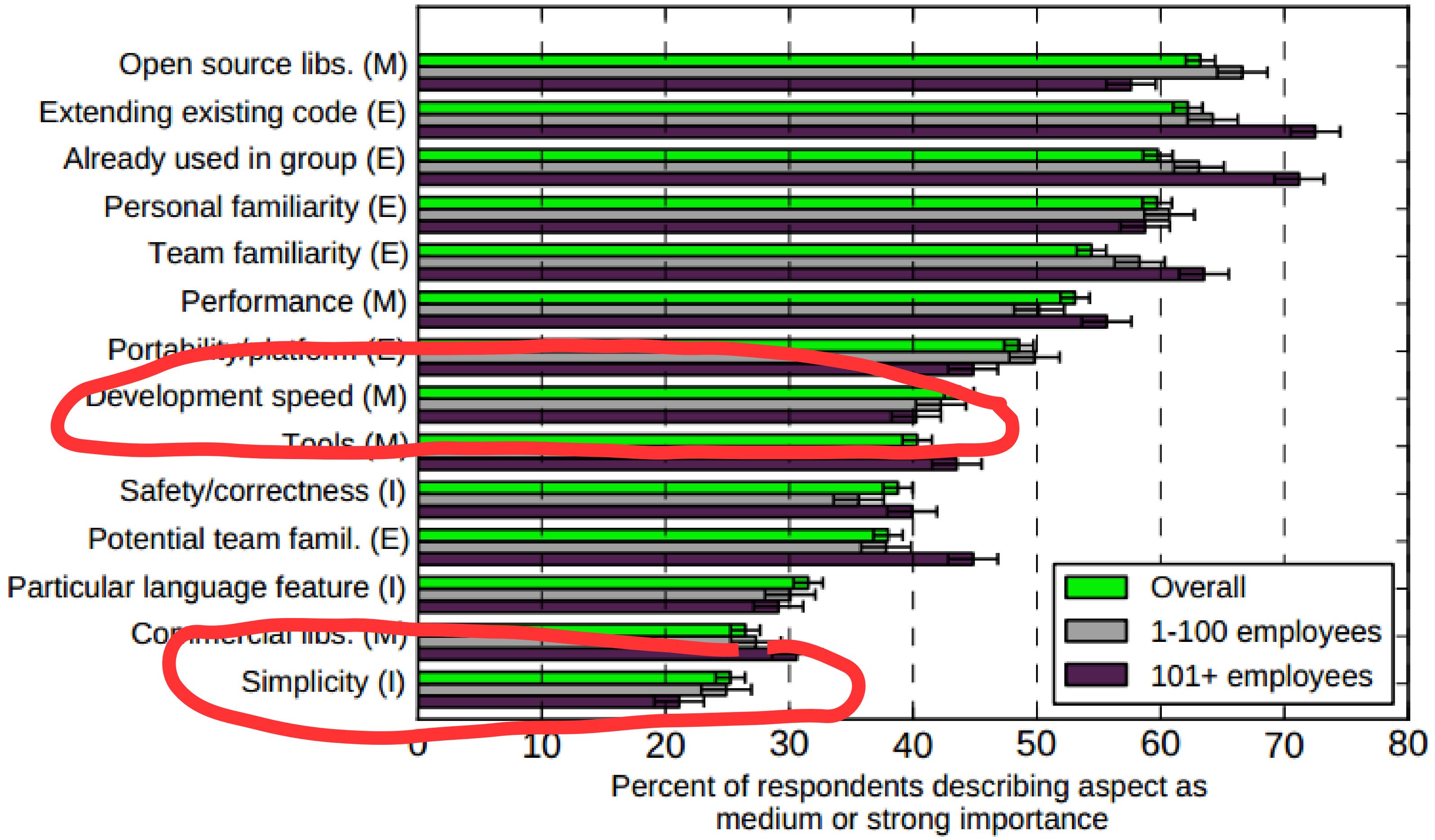


Figure 5: **Importance of different factors when picking a language**. Self-reported for every respondent's last project. Bars show standard error. E = Extrinsic factor, I = Intrinsic, M = Mixed. Shows results broken down by company size for respondents describing a work project and who indicated company size. (Slashdot, n = 1679)

PL Adoption

[Meyerovich, Rabkin, OOPSLA'13]

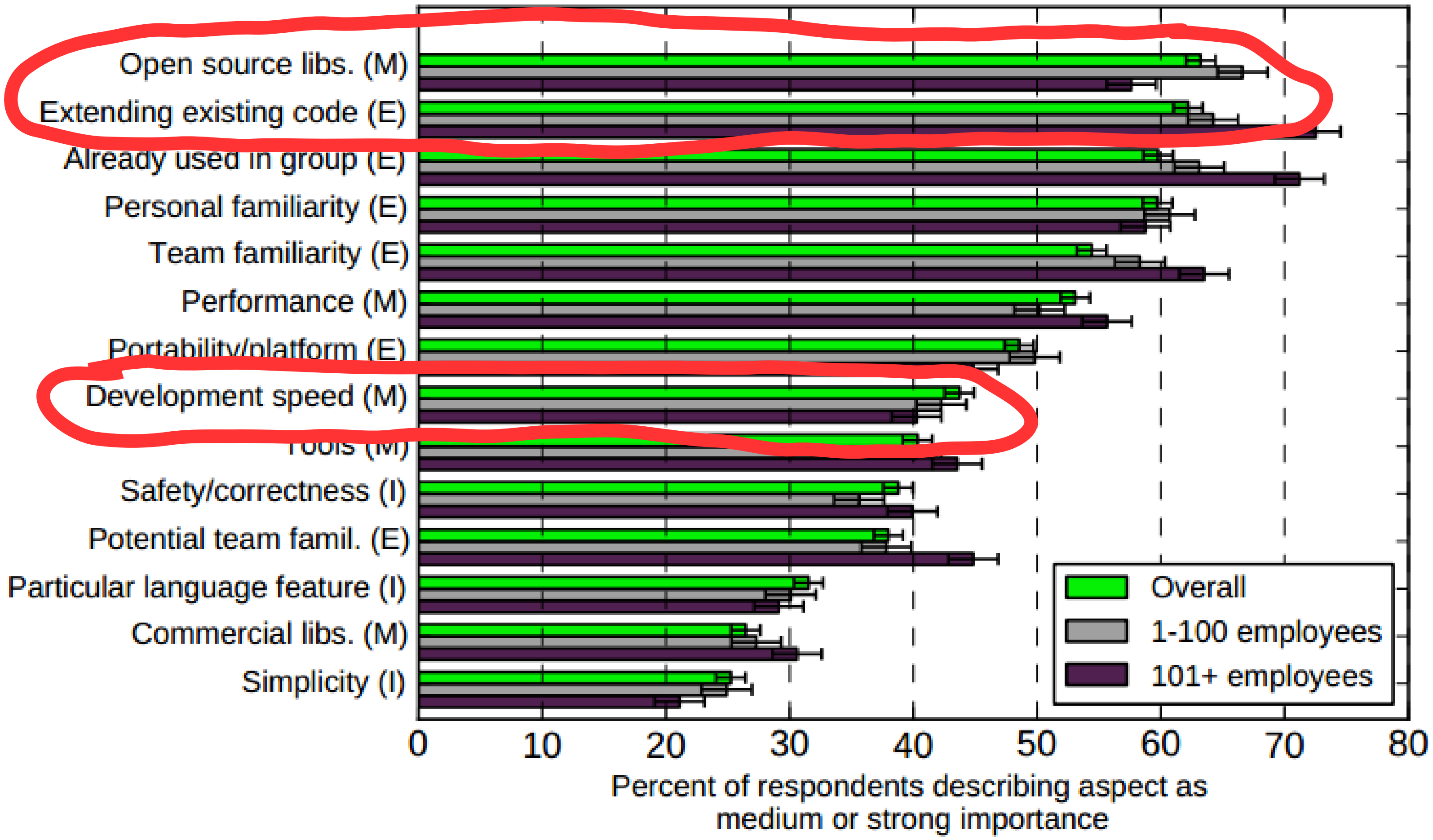


Figure 5: **Importance of different factors when picking a language**. Self-reported for every respondent's last project. Bars show standard error. E = Extrinsic factor, I = Intrinsic, M = Mixed. Shows results broken down by company size for respondents describing a work project and who indicated company size. (Slashdot, n = 1679)

Thought Experiment 2 – Question 1

- If there is a language X that requires 10x more development time, would that stop us from adopting the language?

Thought Experiment 2 – Question 1

- If there is a language X that requires 10x more development time, would that stop us from adopting the language?

  NO!

  (development speed does not matter that much for our decisions)

Thought Experiment 2 – Question 2

- If there is a language X with development speed of factor 10, would this stop us using the other languages?

Thought Experiment 2 – Question 2

- If there is a language X with development speed of factor 10, would this stop us using the other languages?

  NO!

  (development speed does not matter that much for our decisions)

Thought Experiment 2 – Questions

- If there is a language X that requires 10x more development time, would that stop us from adopting the language?

  No

- If there is a language X with development speed of factor 10, would this stop us using the other languages?

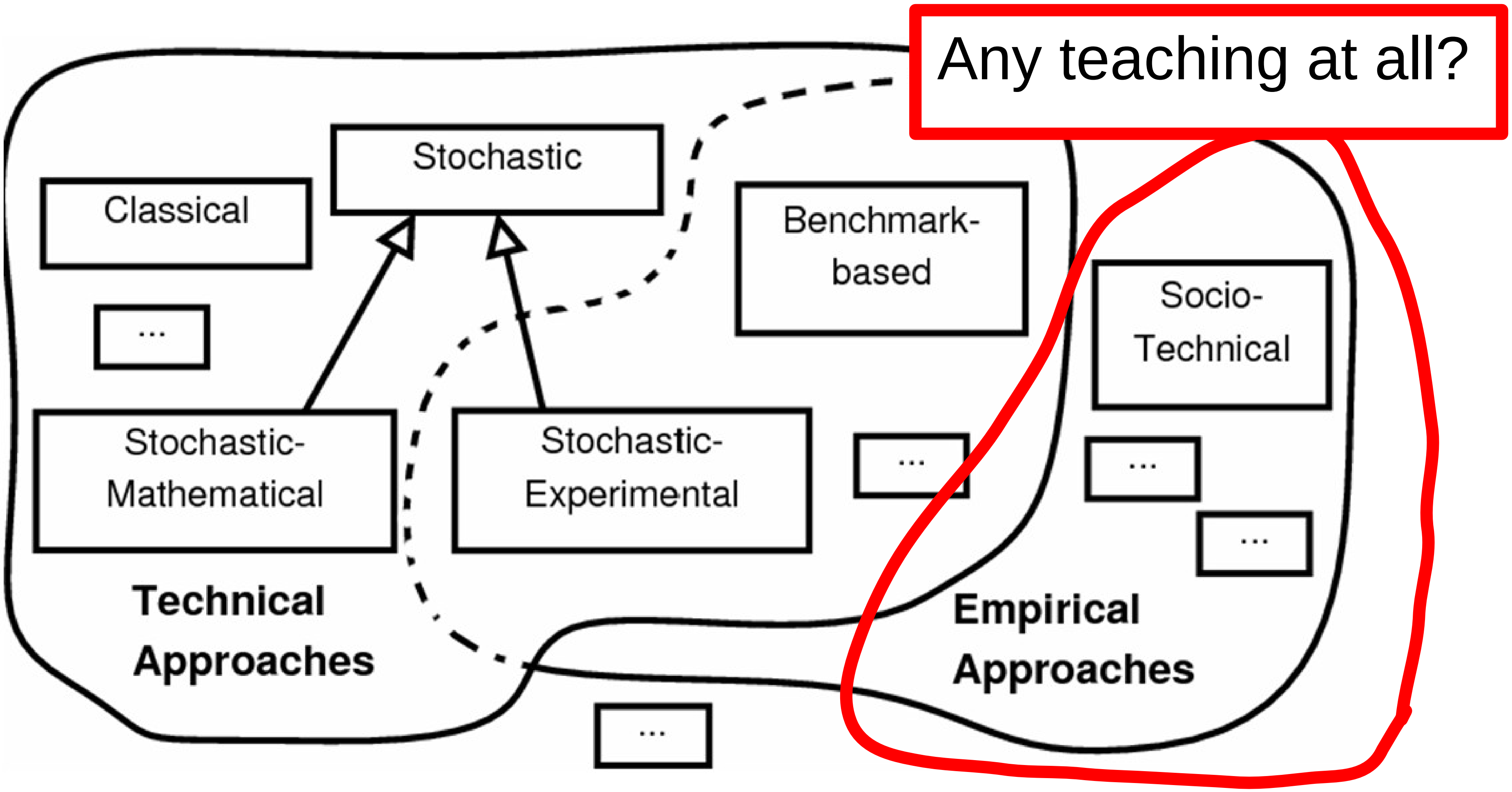
  No

Thought Experiment 2 – Questions

What's wrong with us?

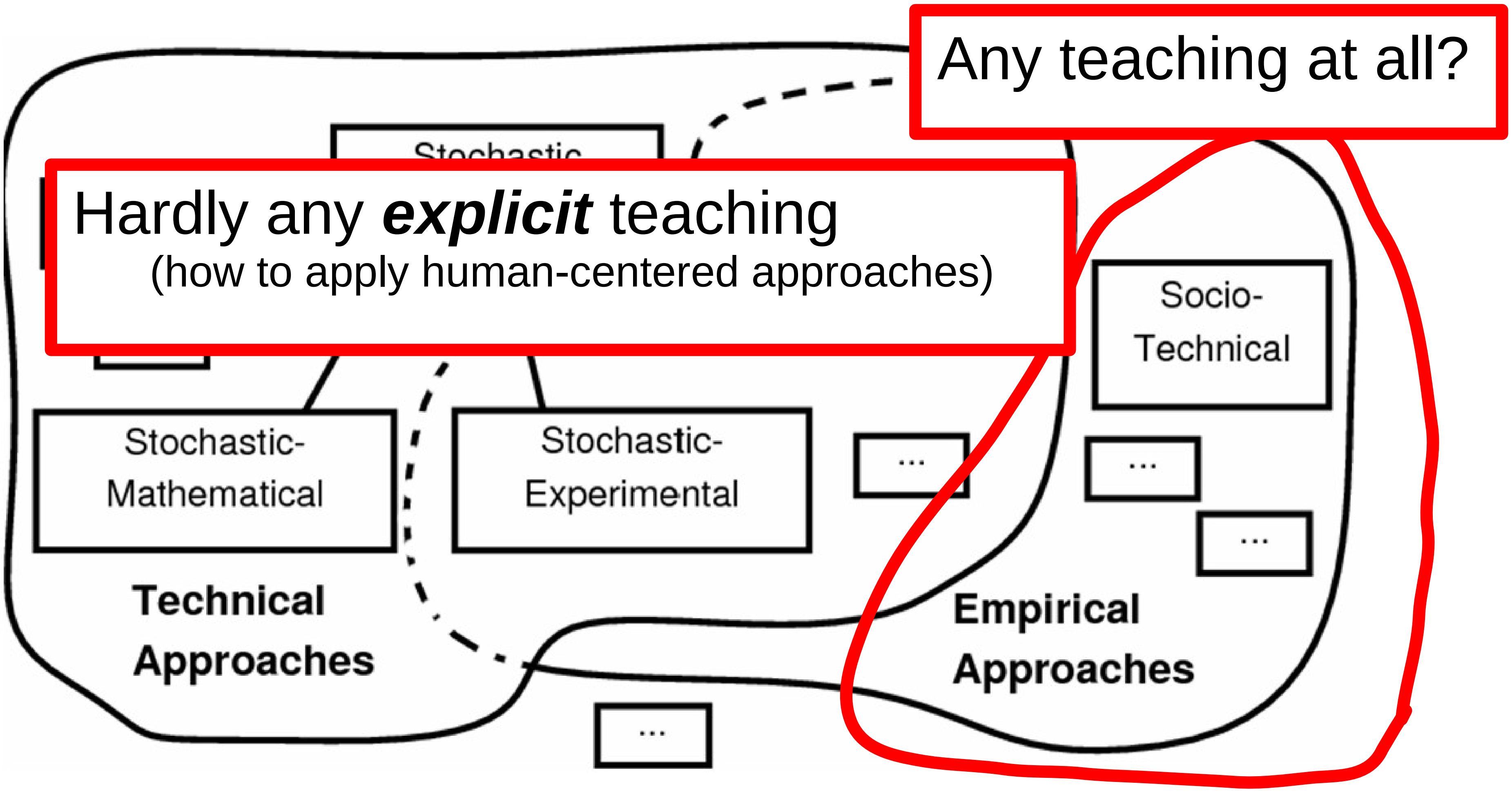
No

SE Research Methods & Education



[Hanenberg, Faith, Hope, Love, Onward'10]

SE Research Methods & Education

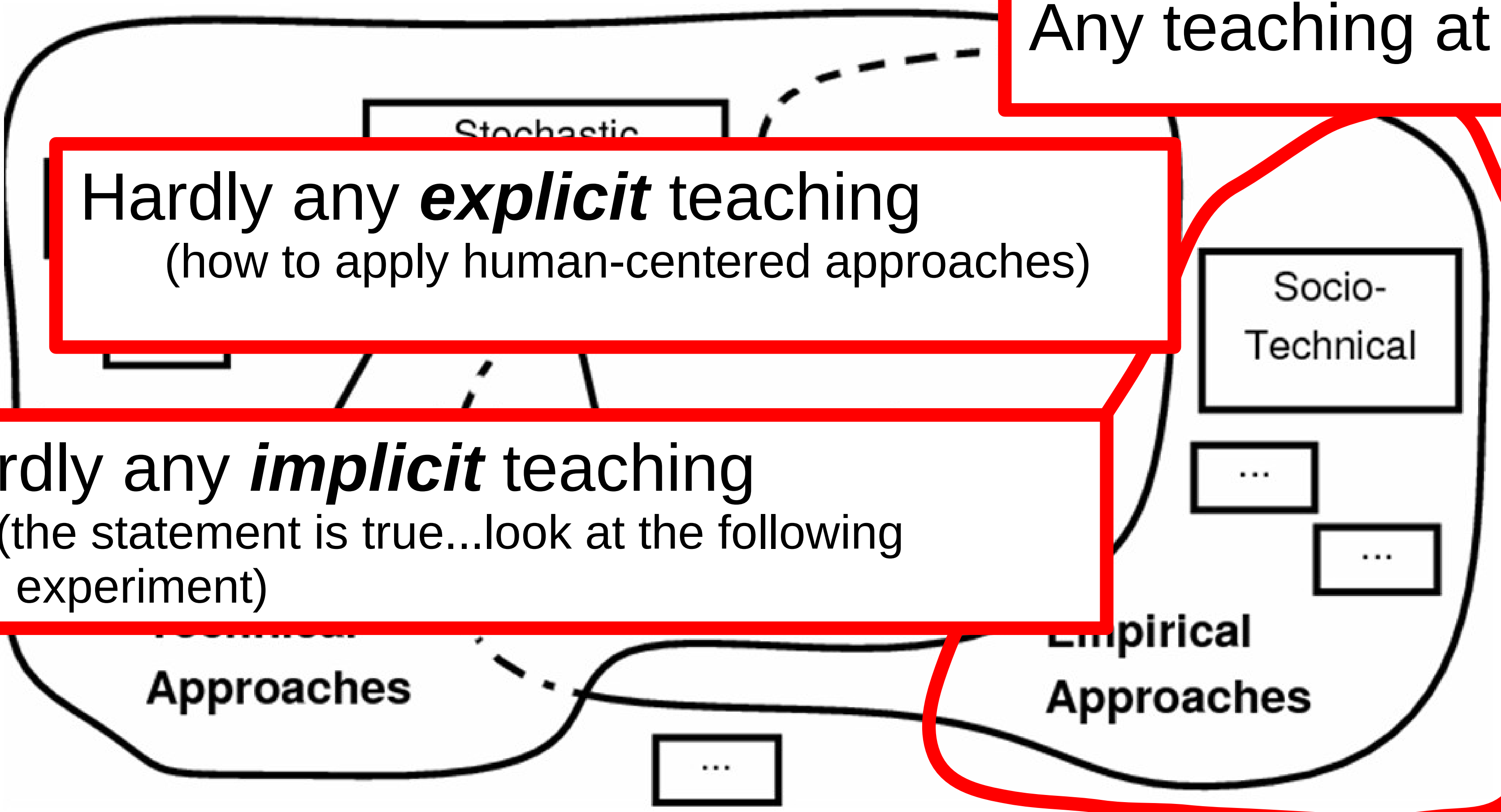


Any teaching at all?

Hardly any ***explicit*** teaching
(how to apply human-centered approaches)

[Hanenberg, Faith, Hope, Love, Onward'10]

SE Research Methods & Education



Any teaching at all?

Hardly any ***explicit*** teaching
(how to apply human-centered approaches)

Hardly any ***implicit*** teaching
(the statement is true...look at the following experiment)

[Hanenberg, Faith, Hope, Love, Onward'10]

Implications of missing teaching

- Empirical knowledge is not well communicated => „numbers are unknown“
  (software engineering courses without mentioning any experiment, same with PL courses)
- We are not trained in „using numbers“ as part of our argumentation
- We do not (or hardly) draw any conclusions from empirical knowledge

What's needed?

- Cultural shift in the PL community
- We need to develop common lines of reasoning
- We need to develop common „standards of knowledge“
- Finally, we need to draw conclusions from the collected knowledge
  - ...don't use languages that make life much harder...
  - ...even if they come from big companies...
  - ...even if they are massivly adverticed....

Conclusion

- Empirical methods need to be taught and applied

  more on this:

  Hanenberg, Stefik, Designing Programming Languages for People: Data-Driven Methods - Tutorial 9: Designing PL for People at Salon G, Friday 10.30

- But it also requires a cultural shift in the PL community to draw conclusions

  more on this:

  Stefik, Hanenberg, The Programming Language Wars, Onward! Essays - Session 3 at Salon A, Friday 1.30 pm

Why do we know so little about programming languages, and what would have happened if we had known more?

Stefan Hanenberg

University of Duisburg-Essen, Germany

Dynamic Language Symposium 2014
Portland, Orgegon, US