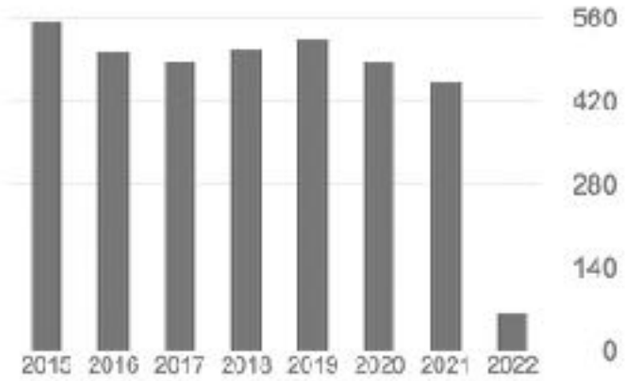
Research Methods in Computer Science

Prof. Serge Demeyer

GRASCOMP Seminar — September 2024

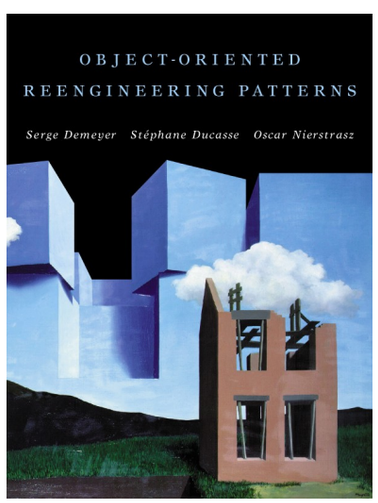
Cited by VIEW ALL

	All	Since 2017
Citations	9287	2521
h-index	43	23
i10-index	107	57



HI10 SECURITY INTEGRATIONS

garvis



Helicopter View

(Ph.D.)
Research

How to *perform* research?
(and get "empirical" results)

How to *write* research?
(and get papers accepted)

How many of you have
done / will do a case-study?

Computer Science

All science is either physics or stamp collecting (E. Rutherford)

We study artefacts produced by *humans*

Computer science is no more about computers than astronomy is about telescopes. (E. Dijkstra)

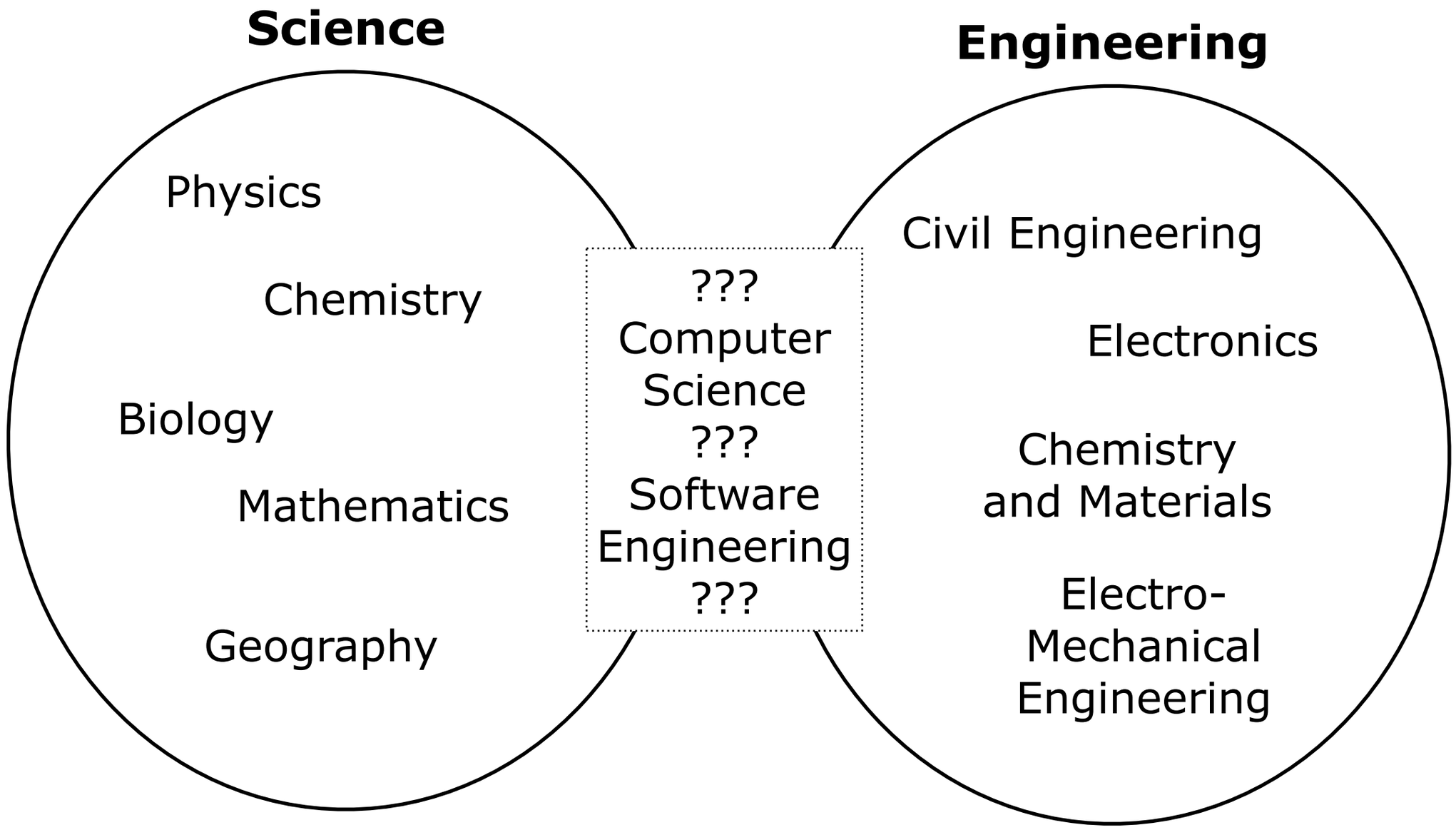
Computer science

Computer engineering

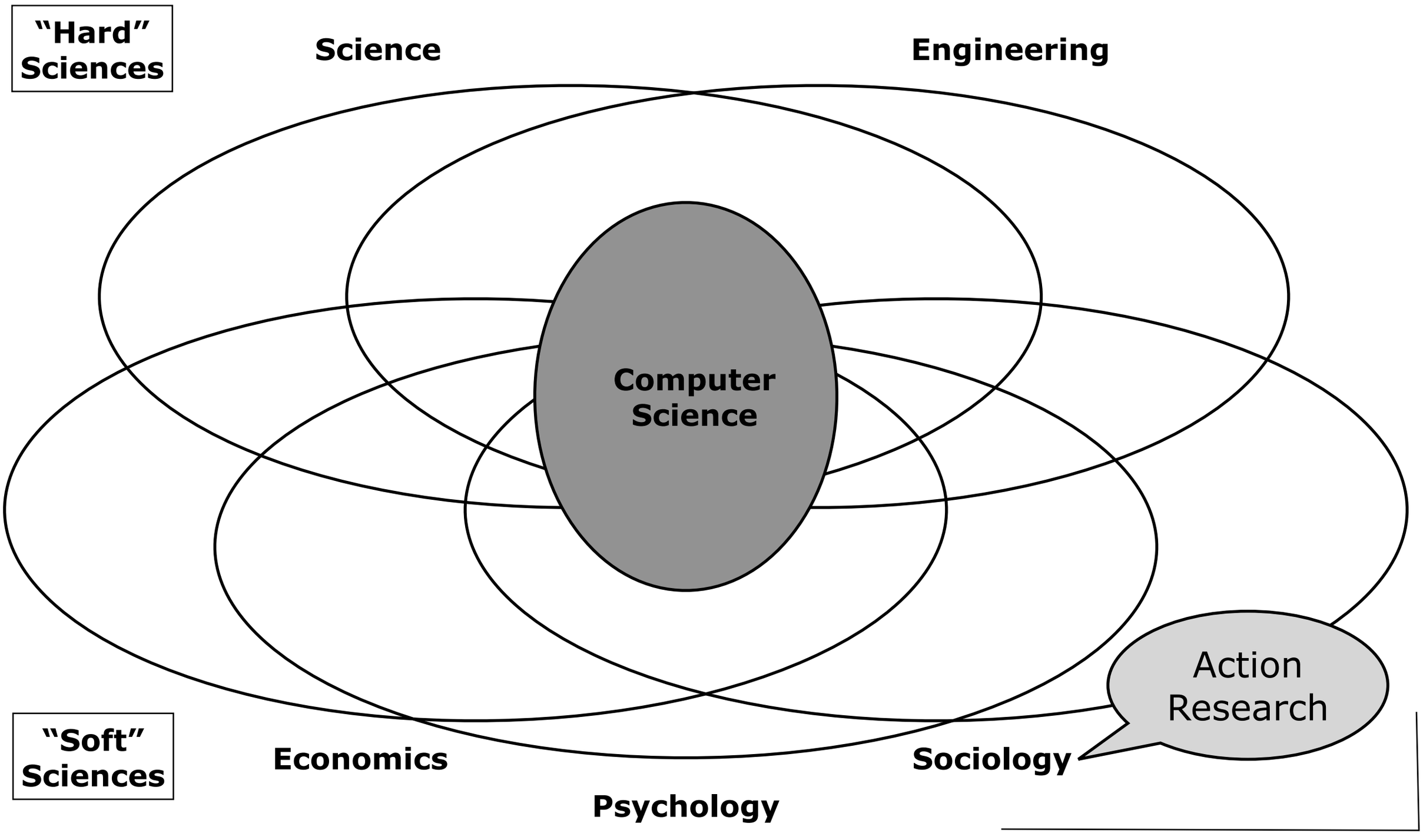
Informatics

Software Engineering

Science vs. Engineering

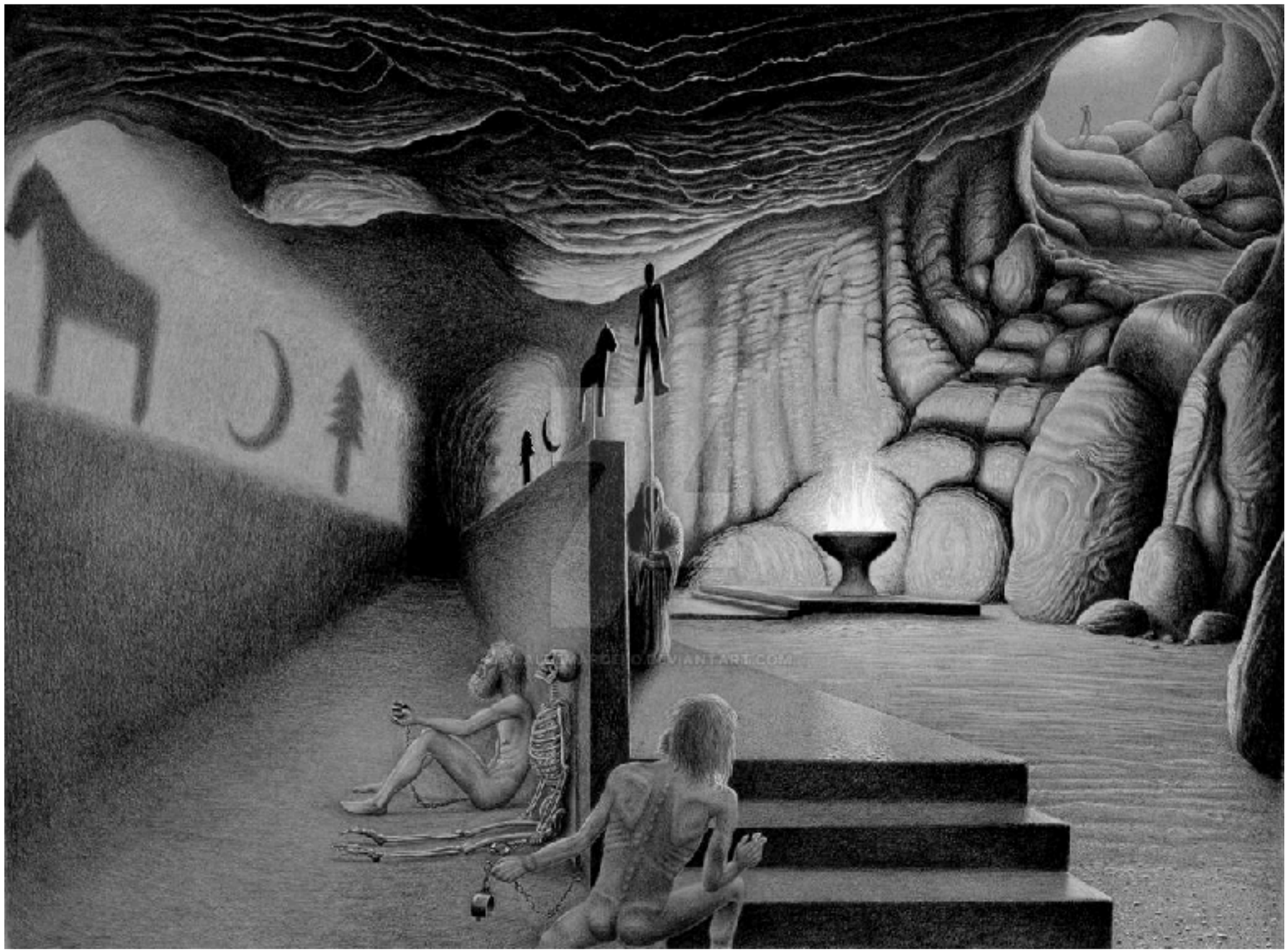


Interdisciplinary Nature



The Oak Forest
Robert Zünd - 1882

The Allegory of the Cave (a.k.a. Plato's Cave)



© calebmarcelo — https://www.deviantart.com/

Dominant view on Research Methods

Physics
("The" Scientific method)

- form hypothesis about a phenomenon
- design experiment
- collect data
- compare data to hypothesis
- accept or reject hypothesis
- … publish (in Nature)
- get someone else to repeat experiment (replication)

Medicine
(Double-blind treatment)

- form hypothesis about a treatment
- select experimental and control groups that are comparable except for the treatment
- collect data
- commit statistics on the data
- treatment ⇒ difference (statistically significant)

Cannot answer the "big" questions
… in timely fashion

- smoking is unhealthy
- climate change
- darwin theory vs. intelligent design
- …
- agile methods

Information and Software Technology
Volume 133, May 2021, 106514

Case Study Research in Software Engineering—It is a Case, and it is a Study, but is it a Case Study?

Claes Wohlin ✉

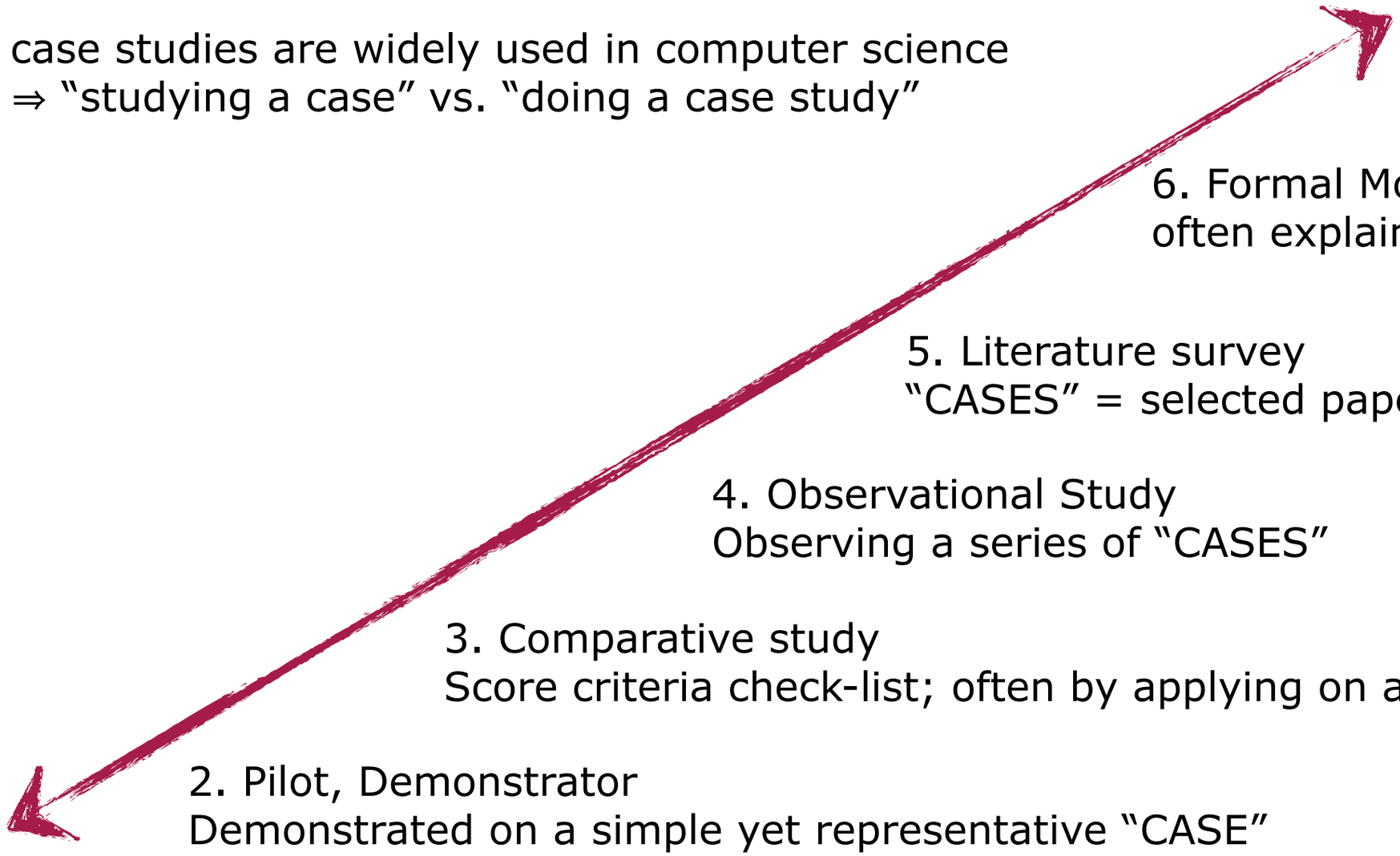
Show more

Share Cite

https://doi.org/10.1016/j.infsof.2021.106514

Case studies

case studies are widely used in computer science
⇒ “studying a case” vs. “doing a case study”



7. Simulation: test prognoses with real observations obtained via a “CASE”

6. Formal Model
often explained using a “CASE”

5. Literature survey
“CASES” = selected papers

4. Observational Study
Observing a series of “CASES”

3. Comparative study
Score criteria check-list; often by applying on a “CASE”

2. Pilot, Demonstrator
Demonstrated on a simple yet representative “CASE”

1. Feasibility study
Proof-of-Concept; often by applying on a “CASE”

Spectrum of cases

created for explanation
- foo, bar examples
- simple model; illustrates differences

Toy-example

accepted teaching vehicle
- "textbook example"
- simple but illustrates *relevant* issues

Exemplar

Martin S. Feather , Stephen Fickas , Anthony Finkelstein , Axel Van Lamsweerde, Requirements and Specification Exemplars, Automated Software Engineering, v.4 n.4, p.419-438, October 1997

Runeson, P. and Höst, M. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. 14, 2 (Apr. 2009), 131-164.

real-life example
- industrial system, open-source system
- context is difficult to grasp

Case

Mining Software Repositories Challenge. [Yearly workshop where research tools compete against one another on a common predefined case.]

competition (tool oriented)
- approved by community
- comparing

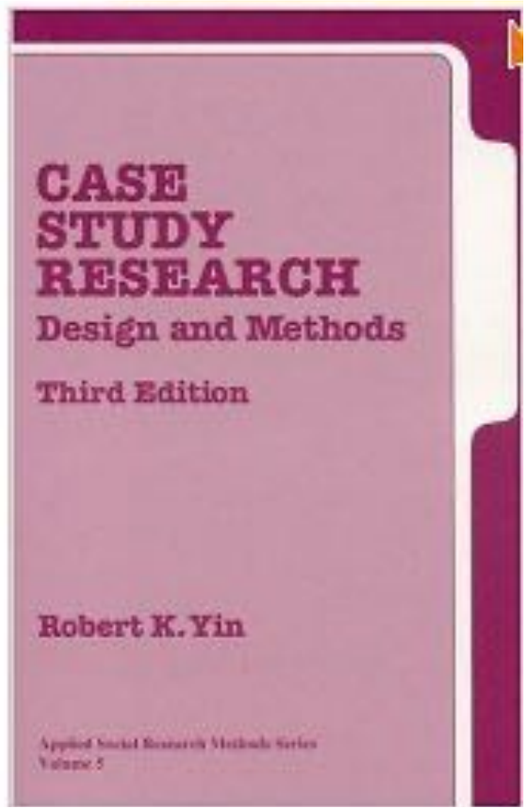
Community case

Susan Elliott Sim, Steve Easterbrook, and Richard C. Holt. Using Benchmarking to Advance Research: A Challenge to Software Engineering, Proceedings of the Twenty-fifth International Conference on Software Engineering, Portland, Oregon, pp. 74-83, 3-10 May, 2003.

Benchmark

benchmark
- approved by community
- known context
- "*planted*" issues

Case Study Research



Sources

- Robert K. Yin. Case Study Research: Design and Methods. 3rd Edition. SAGE Publications. California, 2009.
- Bent Flyvbjerg, "Five Misunderstandings About Case Study Research." Qualitative Inquiry, vol. 12, no. 2, April 2006, pp. 219-245.
- Runeson, P. and Höst, M. 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. 14, 2 (Apr. 2009), 131-164.

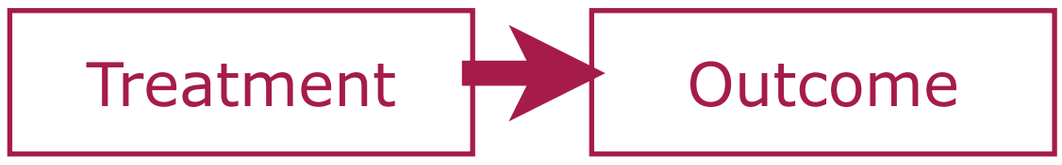
Studying a Case
vs. Performing a Case Study

- Proposition
- Unit of Analysis
- Threats to Validity

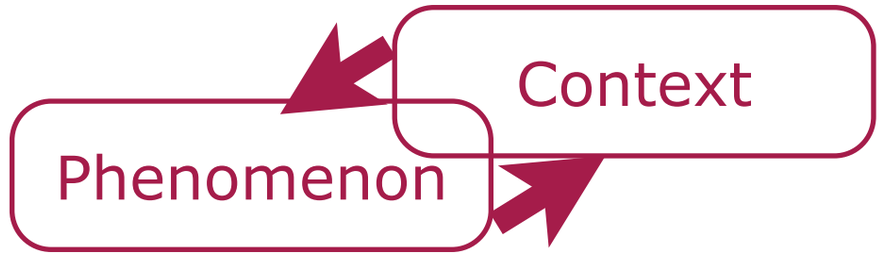
Case study — definition

A case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between the phenomenon and context are not clearly evident
[Robert K. Yin. Case Study Research: Design and Methods; p. 13]

- empirical inquiry: yes, it is empirical research
- contemporary: (close to) real-time observations
  + incl. interviews
- boundaries between the phenomenon and context not clear
  + as opposed to “experiment”

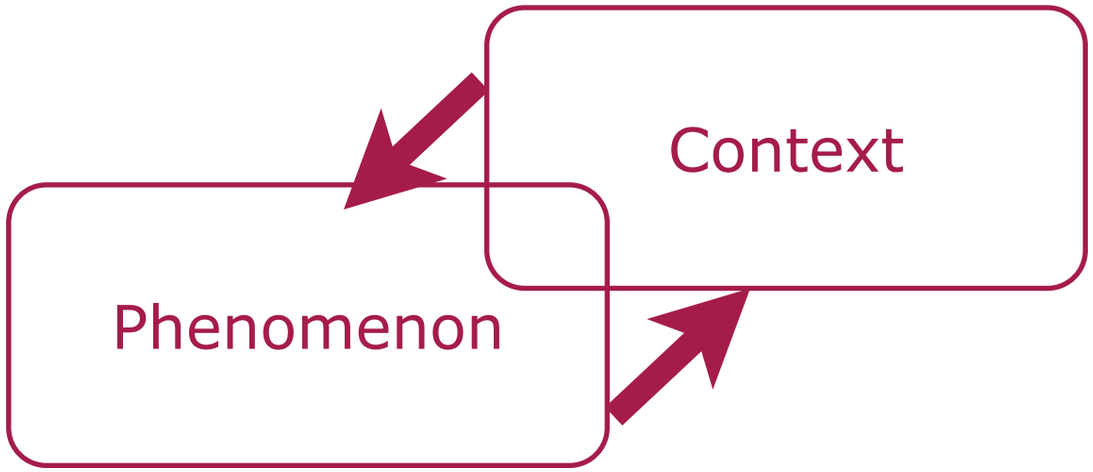


Experiment



Case Study

Case Study — Counter evidence



- many more variables than data points
- multiple sources of evidence; triangulation
- theoretical propositions guide data collection
  (try to confirm or refute propositions with well-selected cases)

Case studies also look for *counter evidence*

Misunderstanding 2: Generalization

One cannot generalize on the basis of an individual case; therefore the case study cannot contribute to scientific development.

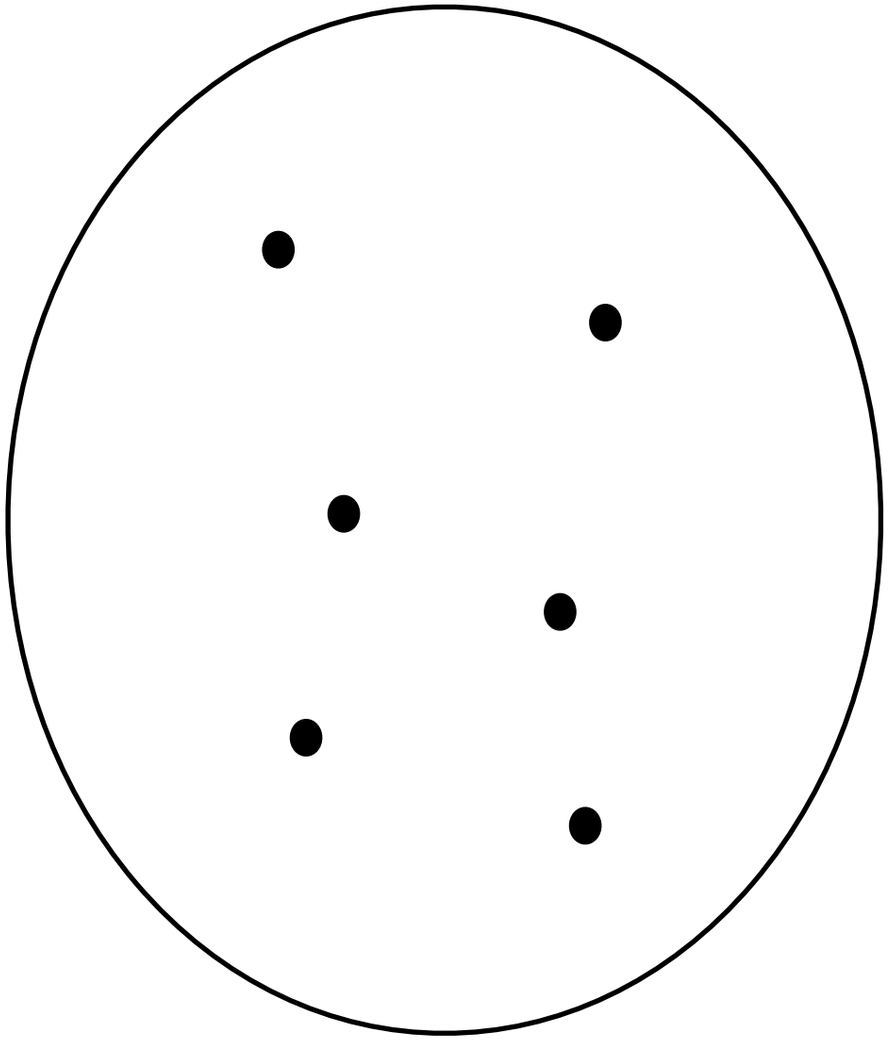
[Bent Flyvbjerg, "Five Misunderstandings About Case Study Research."]

- Understanding
  + The power of examples
  + Formal generalization is overvalued
    - dominant research views of physics and medicine

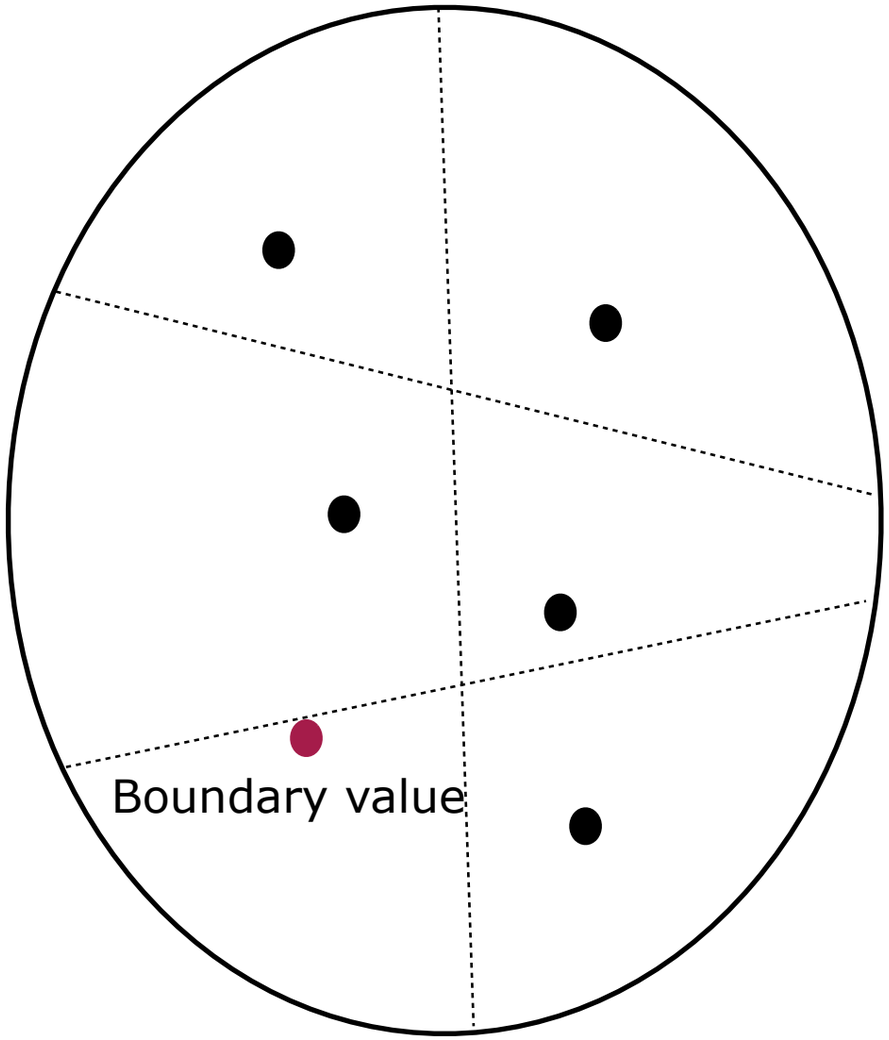
- Counterexamples
  + one black swan falsifies “all swans are white”
    - case studies generate deep understanding; what appears to be white often turns out to be black

- sampling logic vs. replication logic
  + sampling logic: operational enumeration of entire universe
    - use statistics: generalize from “randomly selected” observations
  + replication logic: careful selection of boundary values
    - use logic reasoning: presence of absence of property has effect

Sampling Logic vs. Replication Logic



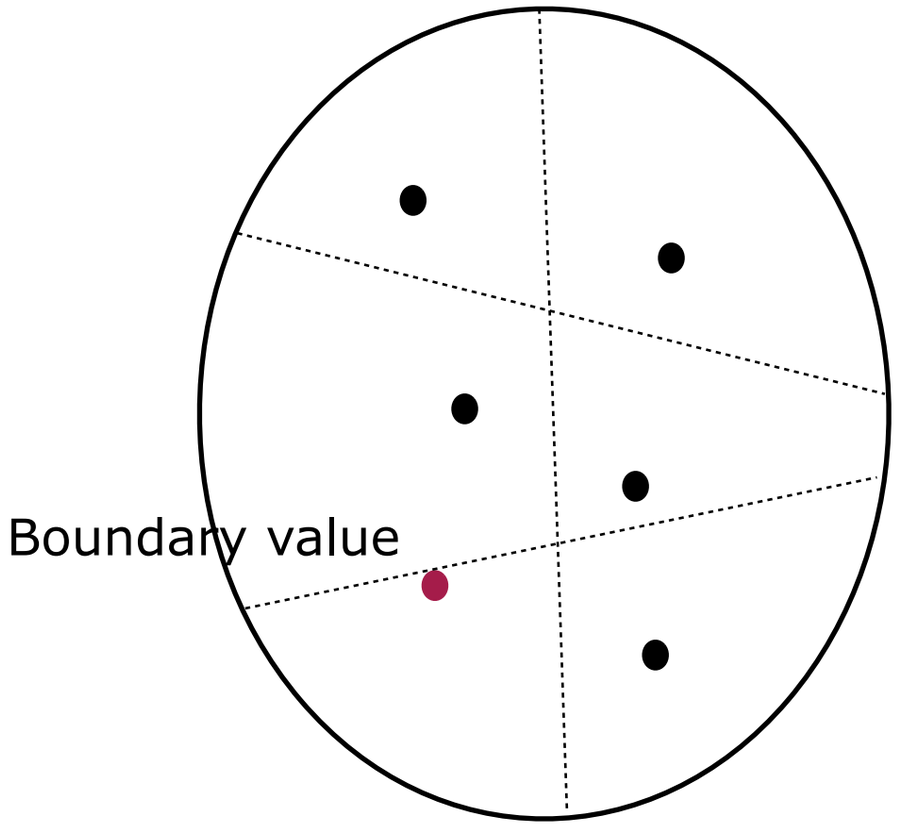
Random selection
⇒ generalize for entire population



Selection of (boundary) value
⇒ understand differences

- propositions
- units of analysis

Proposition (a.k.a. Purpose)



Where to expect boundaries?
⇒ Thorough preparation is necessary!

⇒ You need an explicit *theory*.

Exploratory	Confirmatory
Exploratory case studies are used as initial investigations of some phenomena to derive new hypotheses and build theories.(*)	*Confirmatory* case studies are used to test existing theories. The latter are especially important for refuting theories: a detailed case study of a real situation in which a theory fails may be more convincing than failed experiments in the lab.(*)

(*) Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. Selecting empirical methods for software engineering research. In Forrest Shull, Janice Singer, and Dag I. K. Sjoberg, editors, Guide to Advanced Empirical Software Engineering, pages 285—311. Springer London, 2008.

Units of Analysis

What phenomena to analyze
- depends on research questions
- affects data collection & interpretation
- affects generalizability

Possibilities
- individual developer
- a team
- a decision
- a process
- a programming language
- a tool

Example: Clone Detection, Bug Prediction
- the tool/algorithm
  + Does it work?
- the individual developer
  + How/why does he produce bugs/clones?
- about the culture/process in the team
  + How does the team prevent bugs/clones?
  + How successful is this prevention?
- about the programming language
  + How vulnerable is the programming language towards clones / bugs? (COBOL vs. AspectJ)

Design in advance
- avoid “easy” units of analysis
  + cases restricted to Java because parser
    - Is the language really an issue for your research question?
  + report size of the system (KLOC, # Classes, # Bug reports)
    - Is team composition not more important?

Threats to validity (Case Studies)

- Source: Runeson, P. and Höst, M. 2009. Guidelines for conducting and reporting case study research in software engineering.

1. Construct validity

- Do the operational measures reflect what the researcher had in mind?

2. Internal validity

- Are there any other factors that may affect the results?

> Critical when investigating causality!

3. External validity

- To what extent can the findings be generalized?

> Precise research question & units of analysis required

4. Reliability

- To what extent is the data and the analysis dependent on the researcher (the instruments, …)

Other categories have been proposed as well

- credibility, transferability, dependability, confirmability

Threats to validity = Risk Management

No experimental design can be "perfect"
… but you can limit the chance of deriving false conclusions

- manage the risk of false conclusions as much as possible
  + likelihood
  + impact

- state clearly what and how you alleviated/mitigated the risk
  + construct validity
    - precise metric definitions
    - GQM paradigm
  + internal & external validity
    - report the context consciously
  + Reliability
    - bugs in tools: testing, usage of well-known libraries, …
    - classification: develop guidelines & others repeat classification
    - search for evidence (mailing archives, bug reports, …):
      have an explicit search procedure

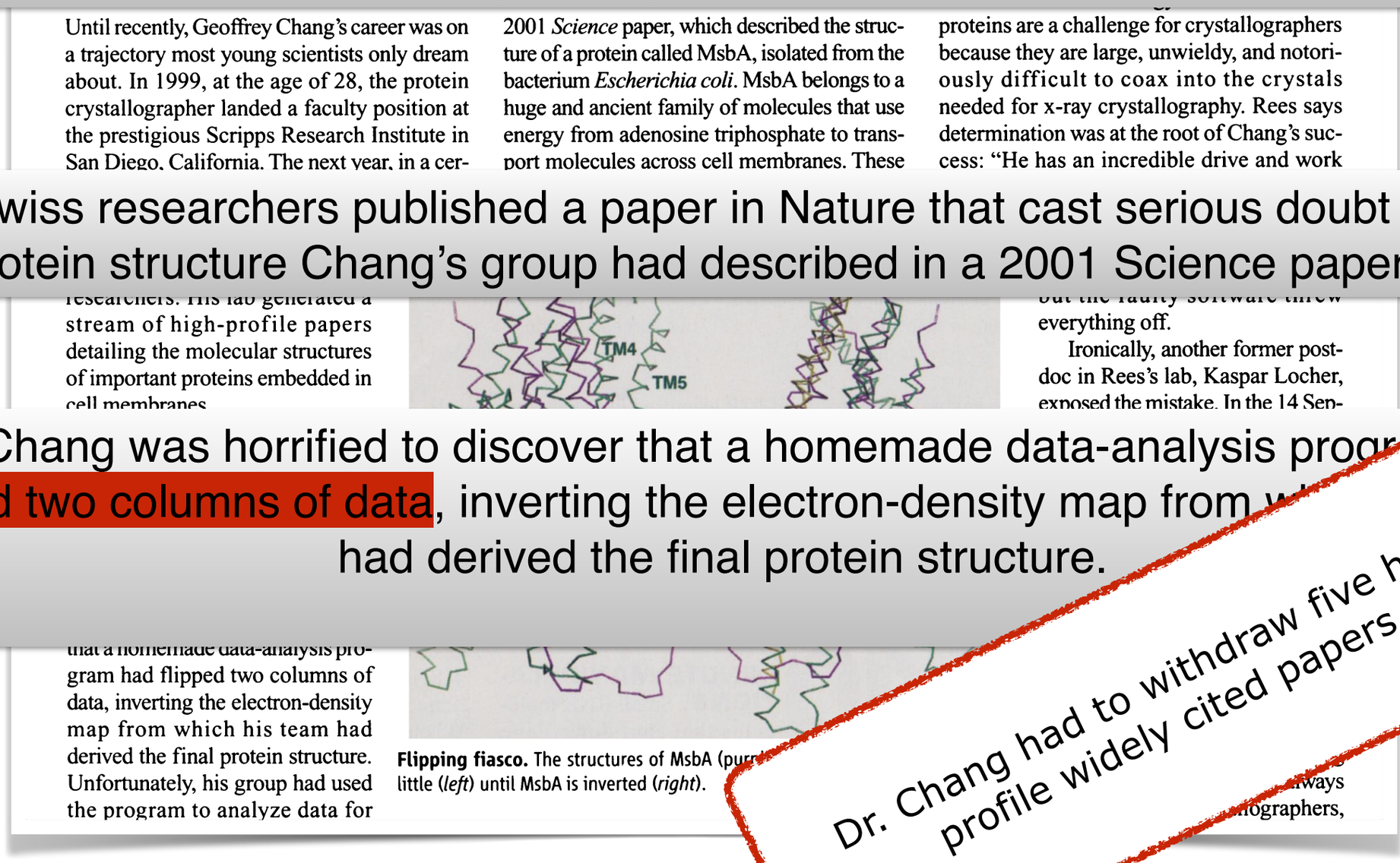
Example: Threat to Instrument Validity

[…] in a ceremony at the White House, Chang received a Presidential Early Career Award for Scientists and Engineers, the country's highest honor for young researchers. His lab generated a stream of high-profile papers detailing the molecular structures of important proteins embedded in cell membranes.

[…] Swiss researchers published a paper in Nature that cast serious doubt on a protein structure Chang's group had described in a 2001 Science paper.

[…] Chang was horrified to discover that a homemade data-analysis program had flipped two columns of data, inverting the electron-density map from which his team had derived the final protein structure.

Dr. Chang had to withdraw five high profile widely cited papers



Until recently, Geoffrey Chang's career was on a trajectory most young scientists only dream about. In 1999, at the age of 28, the protein crystallographer landed a faculty position at the prestigious Scripps Research Institute in San Diego, California. The next year, in a cer-

2001 *Science* paper, which described the structure of a protein called MsbA, isolated from the bacterium *Escherichia coli*. MsbA belongs to a huge and ancient family of molecules that use energy from adenosine triphosphate to transport molecules across cell membranes. These

proteins are a challenge for crystallographers because they are large, unwieldy, and notoriously difficult to coax into the crystals needed for x-ray crystallography. Rees says determination was at the root of Chang's success: "He has an incredible drive and work

researchers. His lab generated a stream of high-profile papers detailing the molecular structures of important proteins embedded in cell membranes.

TM4

TM5

everything off.

Ironically, another former postdoc in Rees's lab, Kaspar Locher, exposed the mistake. In the 14 Sep-

that a homemade data-analysis program had flipped two columns of data, inverting the electron-density map from which his team had derived the final protein structure. Unfortunately, his group had used the program to analyze data for

Flipping fiasco. The structures of MsbA (purp… little (*left*) until MsbA is inverted (*right*).

…ways …lographers,

http://www.jstor.org/sta…

Replication

Replicating MSR:
A study of the potential replicability of papers published in the Mining Software Repositories Proceedings

Gregorio Robles
GSyC/LibreSoft
Universidad Rey Juan Carlos
Madrid, Spain
Email: grex@gsyc.urjc.es

Results show that MSR authors use in general publicly available data sources, mainly from free software repositories, but that the amount of publicly available processed datasets is very low.

© 7th IEEE Working Conference on Mining Software Repositories (MSR 2010), 2010, pp. 171-180, doi: 10.1109/MSR.2010.5463348.

Data Management Plan

TEMPLATE HORIZON 2020 DATA MANAGEMENT PLAN (DMP)

- **Instructions and footnotes in blue must not appear in the text.**
- **For options *(in square brackets)*: the option that applies must be chosen.**
- **For fields in [grey in square brackets] (even if they are part of an option as specified in the previous item): enter the appropriate data.**

Introduction

This Horizon 2020 DMP template has been designed to be applicable to any Horizon 2020 project that produces, collects or processes research data. You should develop a single DMP for your project to cover its overall approach. However, where there are specific issues for individual datasets (e.g. regarding openness), you should clearly spell this out.

Guidelines on FAIR Data Management in Horizon 2020 are available in the Online Manual.

FAIR data management

In general terms, your research data should be 'FAIR', that is findable, accessible, interoperable and re-usable. These principles precede implementation choices and do not necessarily suggest any specific technology, standard, or implementation-solution.

This template is not intended as a strict technical implementation of the FAIR principles, it is rather inspired by FAIR as a general concept.

More information about FAIR:

FAIR data principles (FORCE11 discussion forum)

FAIR principles (article in Nature)

Helicopter View

(Ph.D.)
Research

How to *perform* research?
(and get "empirical" results)

How to *write* research?
(and get papers accepted)

How many of you have submitted a paper recently?

- Was the review helpful?
- Was there a rebuttal phase?
- How did you write the abstract?

The Reviewer

- volunteer
  + don’t waste his/her time

- curious
  + catch his/her interest

- constructive
  + supervises other Ph.D.

- influential
  + wants to support “valuable” papers

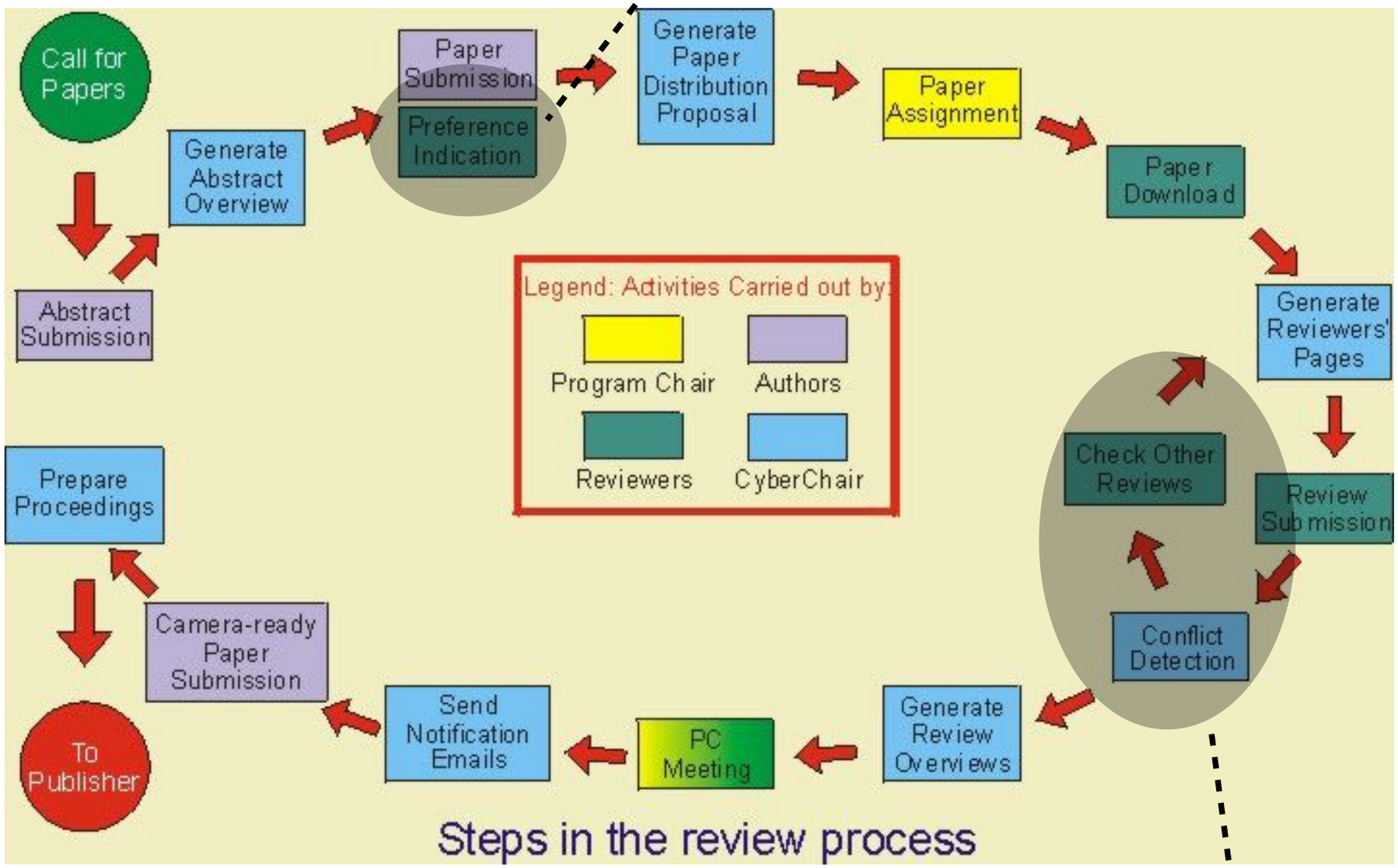
- anonymous
  + avoid tampering

… unfortunately …

- busy
  + read’s on train, bus, air-plane, …

Review Process Steps

Bidding for Abstracts
abstracts + key-words
= "first date" with your reviewer



Identify the Champion
your reviewer needs arguments to support your paper

source: CyberChair (http://www.CyberChair.org)

Providing Keywords

As many as possible?
vs. As few as possible?

	▪Automated reasoning techniques
	▪Component-based systems
	▪Computer-supported cooperative work
	▪Configuration management
	▪Domain modelling and meta-modelling
	▪Empirical software engineering
	▪Human-computer interaction
	▪Knowledge acquisition and management
	▪Maintenance and evolution
	▪Model-based software development
	▪Model-driven engineering and model transformation
	▪Modeling language semantics
	▪Open systems development
	▪Product line architectures
	▪Program understanding
	▪Program synthesis
	▪Program transformation
	▪Re-engineering
	▪Requirements engineering
	▪Specification languages
	▪Software architecture and design
	▪Software visualization
	▪Testing, verification, and validation
	▪Tutoring, help, and documentation systems

Writing Abstracts

Descriptive Abstract

- outlines the topics covered in a piece of writing
  + reader can decide whether to read entire document
- ≈ table of contents in paragraph form.

Informative Abstract

- provides detail about the substance of a piece of writing
  + readers remember key findings
  + reviewers find the claims
- ≈ claim and supporting evidence in paragraph form

≠ executive summary
(abstracts use *the same* level of technical language)

4-line abstract guideline

- source: Kent Beck “How to Get a Paper Accepted at OOPSLA”
  + https://ansymore.uantwerpen.be/system/files/uploads/courses/thesis_master/BeckAbstract.html
  + https://plg.uwaterloo.ca/~migod/research/beckOOPSLA.html

- 1) states the problem
  + WHO is suffering the problem?
  + Connect with your target audience
- 2) why the problem is a problem
  + WHY is it a problem?
  + Cost / Art rather than a science / …
- 3) startling sentence
  + WHAT is the claimed solution?
  + the one thing to say that will catch interest
    … and that you will actually demonstrate in the paper
      > must be falsifiable
- 4) the implication of my startling sentence
  + WHERE can we use this solution?
  + implications for society, community, other researchers, …

Identify The Champion (1/2)

- source: Oscar Nierstrasz, “Identify the Champion,” in Pattern Languages of Program Design 4

- Make Champions Explicit
  + A: Good paper. I will champion it at the PC meeting.
  + B: OK paper, but I will not champion it.
  + C: Weak paper, though I will not fight strongly against it.
  + D: Serious problems. I will argue to reject this paper.

    - *“The most important thing for a reviewer to decide is whether he or she thinks that the paper is worth defending at the PC meeting, not whether it is a great paper or not.”*

- Make Experts Explicit
  + X: I am an expert in the subject area of this paper.
  + Y: I am knowledgeable in the area, though not an expert.
  + Z: My evaluation is that of an informed outsider.
    > detect inexpert champion — expert fence-sitter

These scores are *not* revealed to the authors

Identify The Champion (2/2)

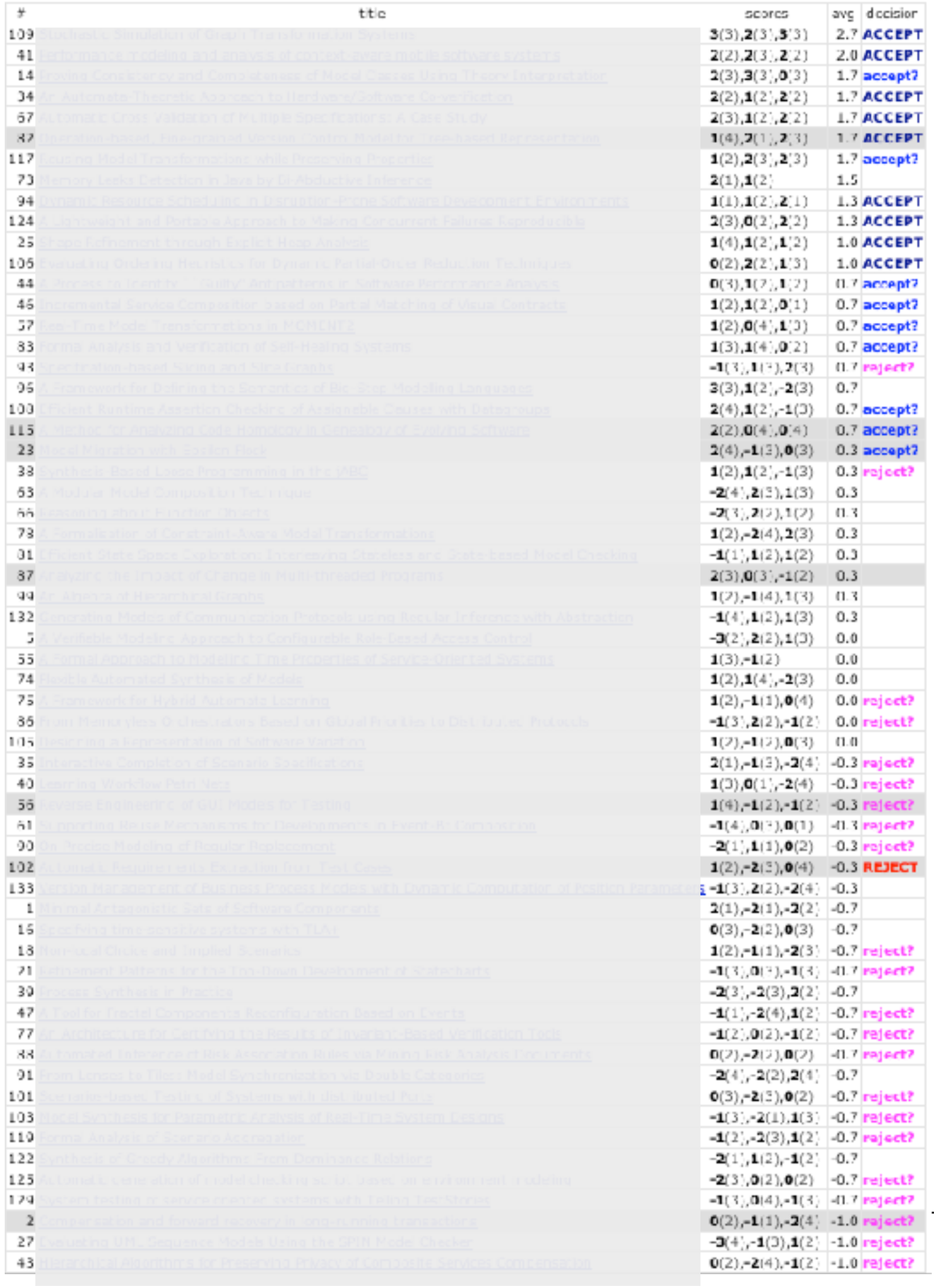
- Identify the Conflicts (classify according to extreme reviews)
  + AA, AB: All reviews are positive, at least one champion.
  + AC: Likely accept; at least one champion, and no strong detractor.
  + AD: This is a serious conflict, and will certainly lead to debate.
  + BC: Borderline papers, no strong advocate nor a detractor.
  + BD: Likely to be rejected.
  + CC, CD, DD: Almost certain rejects.

- inexpert champion
  + If all champions are Y (or Z)
  + If all reviews are Y or Z
    > solicit extra review

- expert fence-sitters
  + Experts tend to be more critical
    > B or even C ratings by X may turn out to be champions
    (remember: PC members want to influence the research)

Example: Easychair

- Clear accept at top
- Clear reject at the bottom (not shown)
- middle area: to discuss



#	title	scores	avg	decision
109	Stochastic Simulation of Graph Transformation Systems	3(3),2(3'),3(3')	2.7	ACCEPT
41	Performance modeling and analysis of context-aware mobile software systems	2(2),2(3'),2(2')	2.0	ACCEPT
14	Proving Consistency and Completeness of Model Classes Using Theory Interpretation	2(3),3(3'),0(3')	1.7	accept?
34	An Automata-Theoretic Approach to Hardware/Software Co-Verification	2(2),1(2'),2(2')	1.7	ACCEPT
67	Automatic Cross-Validation of Multiple Specifications: A Case Study	2(3),1(2'),2(2')	1.7	ACCEPT
87	Interaction-based, Fine-grained Version Control Model for Tree-based Representation	1(4),2(1'),2(3')	1.7	ACCEPT
117	Reusing Model Transformations while Preserving Properties	1(2),2(3'),2(3')	1.7	accept?
73	Memory Leak Detection in Java by Bi-Abductive Inference	2(1),1(2')	1.5	
94	Dynamic Resource Scheduling in Disruption-Prone Software Development Environments	1(1),1(2'),2(1')	1.3	ACCEPT
124	A Lightweight and Portable Approach to Making Concurrent Failures Reproducible	2(3),0(2'),2(2')	1.3	ACCEPT
25	Slope Refinement through Explicit Heap Analysis	1(4),1(2'),1(2')	1.0	ACCEPT
105	Evaluating Ordering Heuristics for Dynamic Partial-Order Reduction Techniques	0(2),2(2'),1(3')	1.0	ACCEPT
44	Approach to Identify Quality Anti-patterns in Software Performance Analysis	0(3),1(2'),1(2')	0.7	accept?
46	Incremental Service Composition based on Partial Matching of Visual Contracts	1(2),1(2'),0(1')	0.7	accept?
57	Real-Time Model Transformations in MOMENT2	1(2),0(4'),1(3')	0.7	accept?
83	Formal Analysis and Verification of Self-Healing Systems	1(3),1(4'),0(2')	0.7	accept?
93	Expectation-based Scope and Slice Grouping	-1(3'),1(3'),2(3')	0.7	reject?
95	A Framework for Defining the Semantics of Big Step Modeling Languages	3(3),1(2'),-2(3')	0.7	
103	Efficient Runtime Assertion Checking of Assignable Clauses with Datagroups	2(4),1(2'),-1(3')	0.7	accept?
115	A Method for Analyzing Code Ownership in Complex Software Systems	2(2),0(4'),0(4')	0.7	accept?
23	Model Migration with Epsilon Flock	2(4),-1(3'),0(3')	0.3	accept?
38	Synthesis Based Loop Programming in the XDC	1(2),1(2'),-1(3')	0.3	reject?
53	A Modular Model Composition Technique	-2(4'),2(3'),1(3')	0.3	
66	Reasoning about Function Objects	-2(3'),2(3'),1(2')	0.3	
78	A Formalization of Corrected-and-Aware Model Transformations	1(2),-2(4'),2(3')	0.3	
91	Efficient State Space Exploration: Interweaving Stateless and State-based Model Checking	-1(1'),1(2'),1(2')	0.3	
97	Analyzing the Impact of Changes in Multi-threaded Programs	2(3),0(3'),-1(2')	0.3	
99	An Analysis of Hierarchical Graphs	1(2),-1(4'),1(3')	0.3	
132	Generating Medium Grained Communication Protocols using Rely-by-Reference with Abstraction	-1(4'),1(2'),1(3')	0.3	
5	A Verifiable Modeling Approach for Configurable Role-Based Access Control	-2(2'),2(2'),1(3')	0.0	
55	A Formal Approach to Modeling Time Properties of Service-Oriented Systems	1(3),-1(2')	0.0	
74	Model-Automated Synthesis of Models	1(2),1(4'),-2(3')	0.0	
75	A Framework for Hybrid Automata Learning	1(2),-1(1'),0(4')	0.0	reject?
86	From Pleasure to Obligation: Based on Global Priorities to Distributed Protocols	-1(3'),2(2'),-1(2')	0.0	reject?
116	Describing a Representation of Software Variation	1(2),-1(2'),0(3')	0.0	
35	Interactive Completion of Scenario Specifications	2(1),-1(3'),-2(4')	-0.3	reject?
40	Learning Workflow Petri Nets	1(3),0(1'),-2(4')	-0.3	reject?
56	Reverse Engineering of GUI Models for Testing	1(4),-1(2'),-1(2')	-0.3	reject?
61	Supporting Reuse Mechanisms for Development in Event-B: Formalization	-1(4'),0(3'),0(1')	-0.3	reject?
90	On Process Modeling of Regular Replacement	-2(1'),1(1'),0(2')	-0.3	reject?
102	Automatic Requirements Prioritization from Test Cases	1(2),-2(3'),0(4')	-0.3	REJECT
133	Version Management of Business Process Models with Dynamic Computation of Position Parameters	-1(3'),2(2'),-2(4')	-0.3	
1	Minimal Antagonistic Sets of Software Components	2(1),-2(1'),-2(2')	-0.7	
16	Specifying time-sensitive systems with TLA+	0(3),-2(2'),0(3')	-0.7	
18	Non-visual Choice and Implied Scenarios	1(2),-1(1'),-2(3')	-0.7	reject?
21	Refinement Patterns for the Top-Down Development of Statecharts	-1(3'),0(3'),-1(3')	-0.7	reject?
39	Process Synthesis in Practice	-2(3'),-2(3'),2(2')	-0.7	
47	A Tool for Fractal Components Reconfiguration Based on Events	-1(1'),-2(4'),1(2')	-0.7	reject?
77	An Architecture for Certifying the Results of Invariant-Based Verification Tools	-1(2'),0(2'),-1(2')	-0.7	reject?
88	Automated Inference of Role Association Rules via Mining Text Analysis Documents	0(2'),-2(2'),0(2')	-0.7	reject?
91	From Lenses to Tiles: Model Synchronization via Double Categories	-2(4'),-2(2'),2(4')	-0.7	
101	Scenario-based Testing of Systems with Distributed Ports	0(3'),-2(3'),0(2')	-0.7	reject?
103	Model Synthesis for Parametric Analysis of Real-Time System Designs	-1(3'),-2(1'),1(3')	-0.7	reject?
119	Formal Analysis of Scenario Aggregation	-1(2'),-2(3'),1(2')	-0.7	reject?
122	Synthesis of Greedy Algorithms From Dominance Relations	-2(1'),1(2'),-1(2')	-0.7	
125	Automatic generation of model checking scripts based on environment modeling	-2(3'),0(2'),0(2')	-0.7	reject?
129	Search testing of service-oriented systems with Telling TestStories	-1(3'),0(4'),-1(3')	-0.7	reject?
2	Compensation and forward recovery in concurrent transactions	0(2'),-1(1'),-2(4')	-1.0	reject?
27	Checking UML Sequence Models Using the SPIN Model Checker	-2(4'),-1(3'),1(2')	-1.0	reject?
43	Hierarchical Algorithms for Preserving Privacy of Composite Services Compositions	0(2'),-2(4'),-1(2')	-1.0	reject?

Make it Easy for your Champion

- Select appropriate keywords
  + Why are you in the scope of the conference/journal/…?

- Test the abstract
  + Start early with the abstract
  + Ask for early (external) feedback

- Visible claims
  + Abstract + intro + conclusion have have visible claim(s)
  + Ask early feedback to summarize what reviewers think the claim is

- Clear validation
  + Champion is then able to defend it against detractors

- Write to the Program Committee
  + Target a PC member
  + Have a clear picture of your champion

Shadow PC / Junior PC

Shadow Program Committee Initiative: Process and Reflection

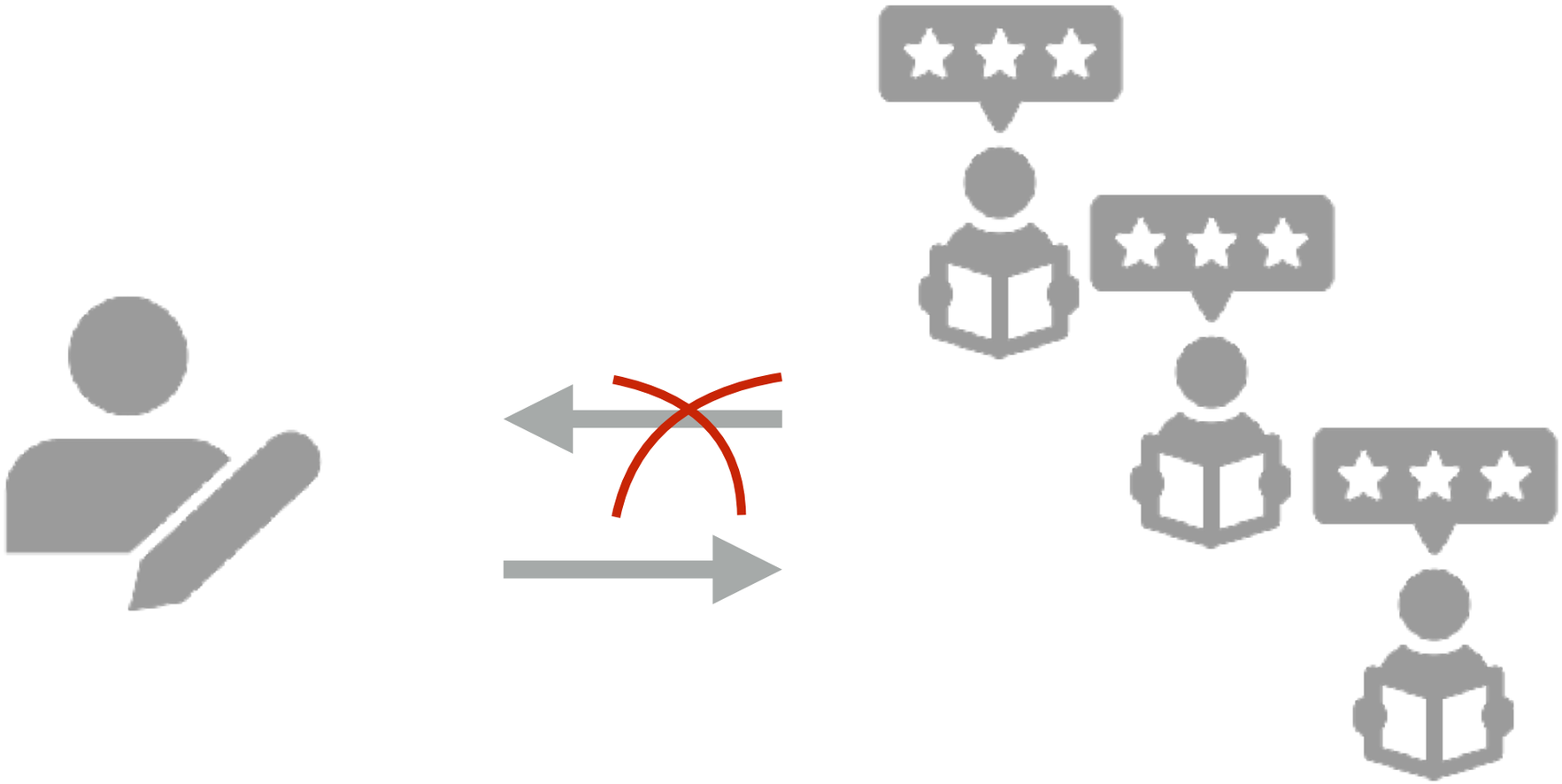
Authors: Patanamon Thongtanunam, Ayushi Rastogi, Foutse Khomh, Serge Demeyer, Meiyappan Nagappan, Kelly Blincoe, Gregorio Robles Authors Info & Claims

ACM SIGSOFT Software Engineering Notes, Volume 46, Issue 4 • October 2021 • pp 16–18 • https://doi.org/10.1145/3485952.3485956

Online: 28 October 2021 Publication History

Allows future PC members to learn first-hand about the peer-review process and gain experience as a reviewer and learn from the senior researchers on how to write a good review. The Shadow PC will provide reviews on a subset of submissions to the technical track of the conference (The authors will opt-in for their paper to be reviewed by the Shadow PC).

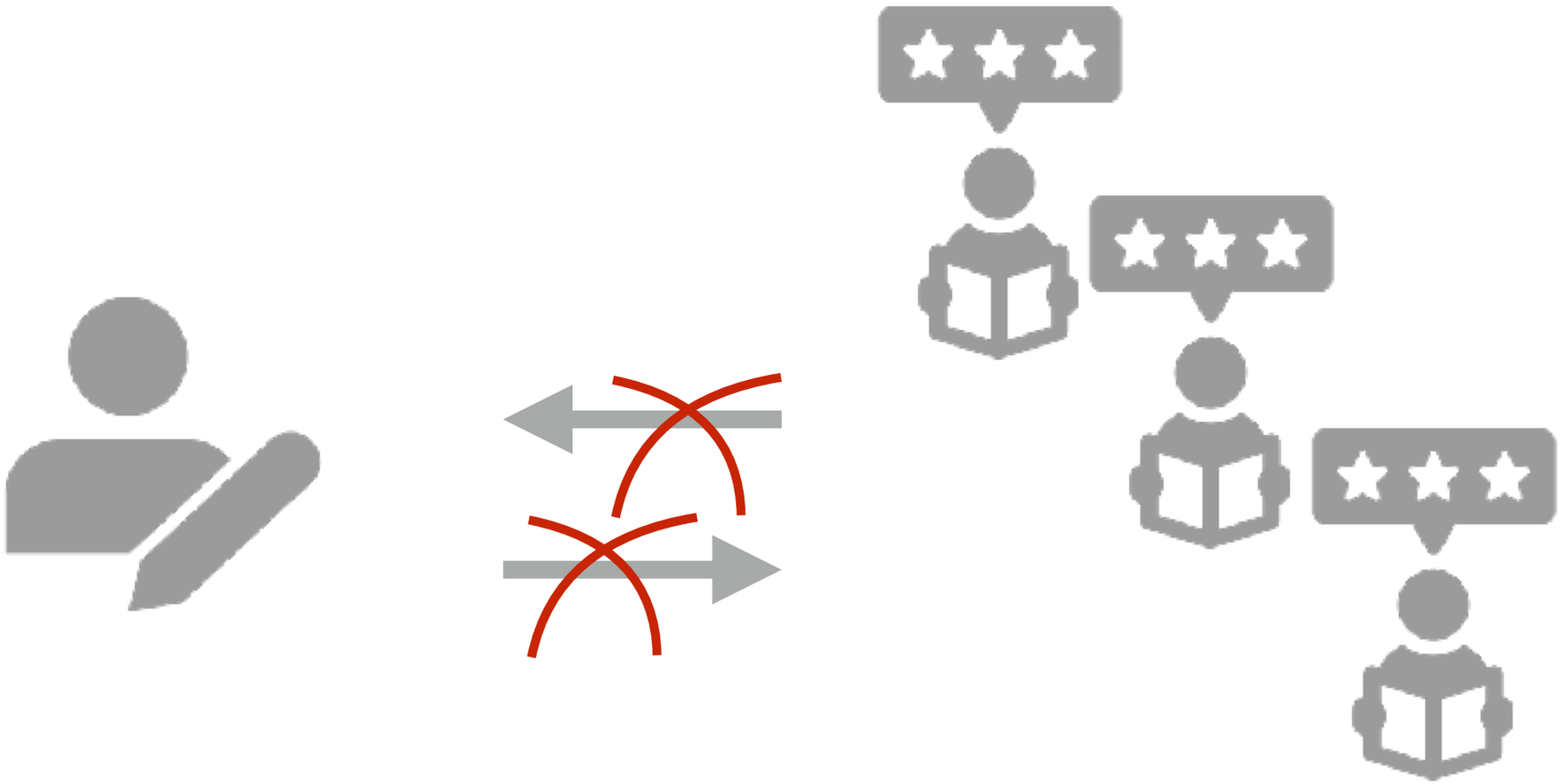
Single Blind Reviewing



Author is Known

Reviewers are Anonymous

Double Blind Reviewing

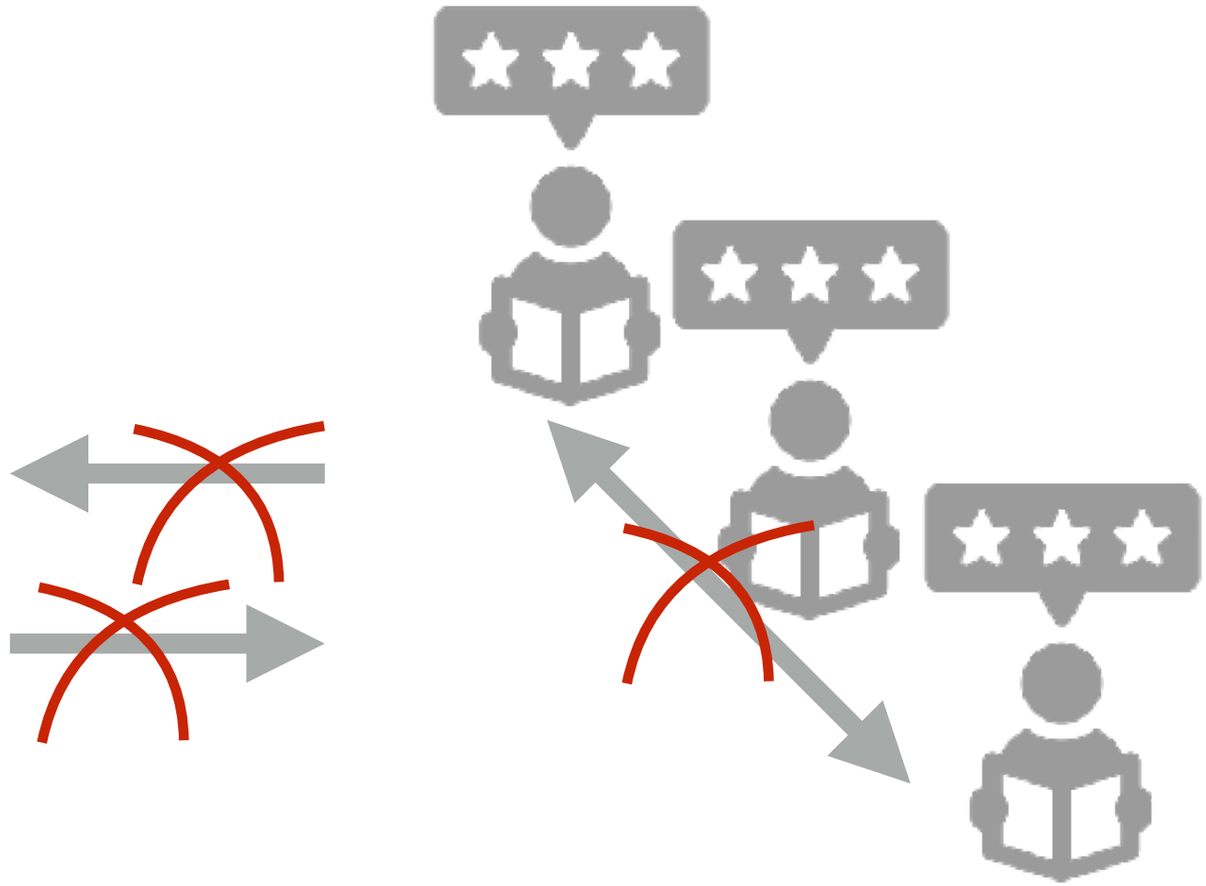


Author is Anonymous

Reviewers are Anonymous

Triple Blind Reviewing

Author is Anonymous



Reviewers are Anonymous
(Also to one another)

(Unconscious) Bias

Update

Research Focus

Double-blind review favours increased representation of female authors

Amber E. Budden [1, 2] ✉, Tom Tregenza [3], Lonnie W. Aarssen [4], Julia Koricheva [5], Roosa Leimu [6], Christopher J. Lortie [7]

Show more

Share Cite

https://doi.org/10.1016/j.tree.2007.07.008 Get rights and content

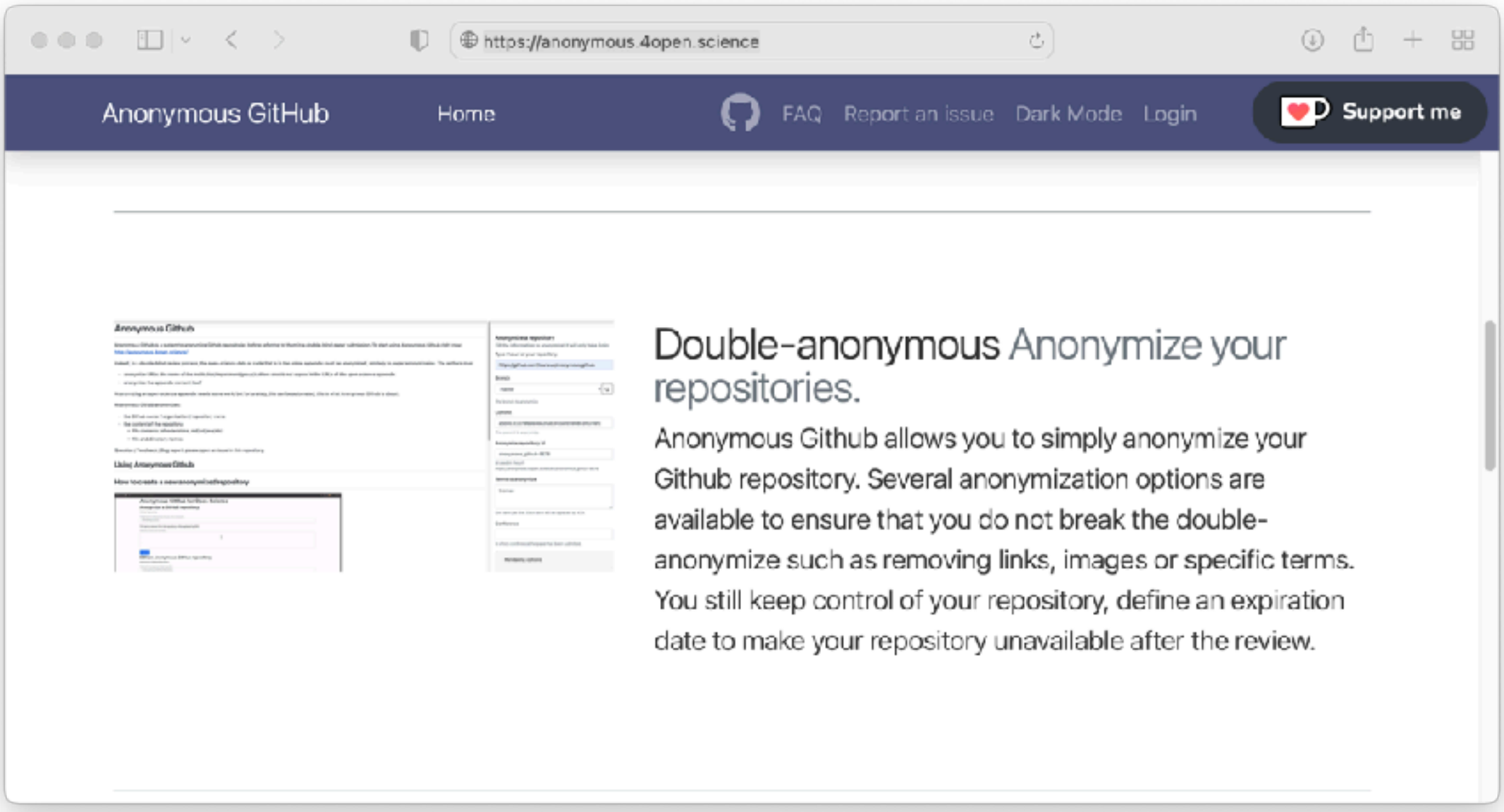
DOI:10.1145/3208157 C. Le Goues, Y. Brun, S. Apel, E. Berger, S. Khurshid, and Y. Smaragdakis

Viewpoint

Effectiveness of Anonymization in Double-Blind Review

Assessing the effectiveness of anonymization in the review process.

https://anonymous.4open.science



Rebuttal

Author Response Period
ICSE 2022 will offer a three day author response period. In this period the authors will have the opportunity to inspect the reviews, and to answer specific questions raised by the program committee. This period is scheduled after all reviews have been completed, and serves to inform the subsequent decision making process. Authors will be able to see the full reviews, including the reviewer scores as part of the author response process.

ESEC/FSE 2022
[…] Authors will have an opportunity to respond to reviews during a rebuttal period.

Good Advice

https://andreas-zeller.info/2012/10/01/patterns-for-writing-good-rebuttals.html

1 October 2012

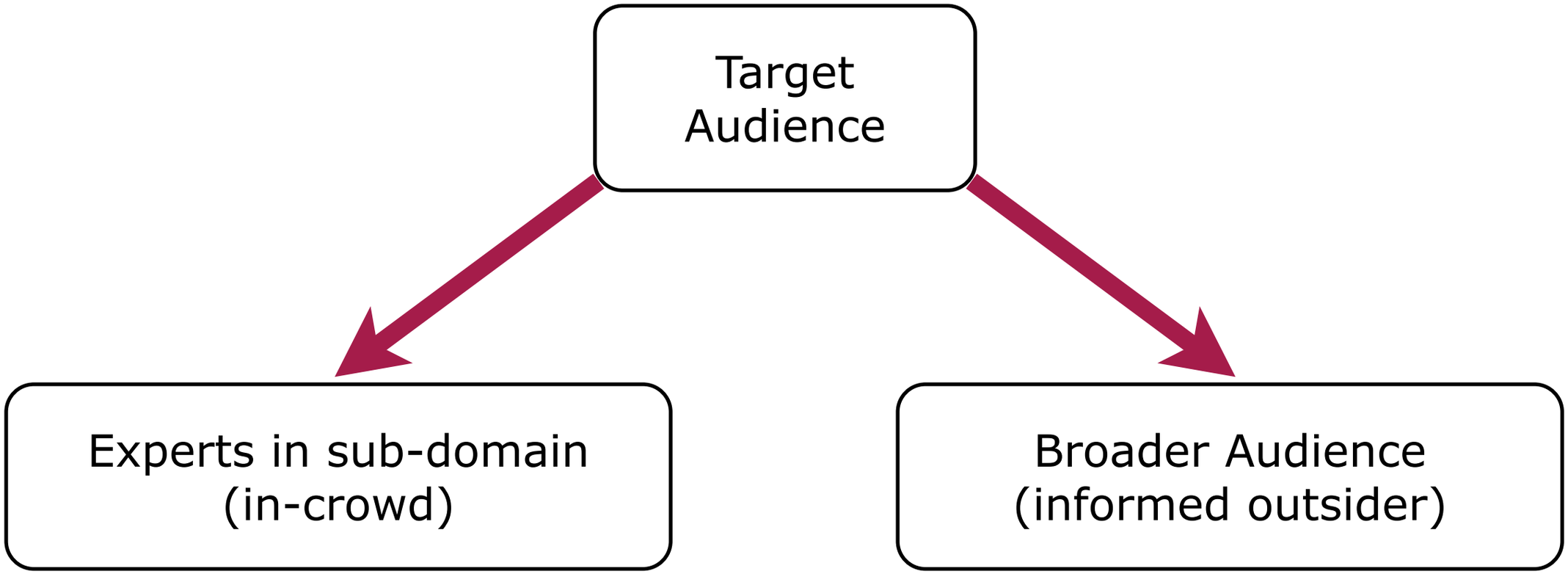
Patterns for writing good rebuttals

by Andreas Zeller

I compiled the following patterns for rebuttals (also known as author clarifications) for major software engineering conferences (ICSE, ESEC, FSE, ASE, ISSTA), having seen a number of rebuttals as PC chair of ESEC/FSE 2011 and having written a number of rebuttals for top conferences. These patterns may or may not be applicable in your context; use at your own risk.

- Understand the decision process
- Identify the undecided
- Identify the champion
- Arm the champion
- Identify the detractors
- Answer the questions
- Write for the PC chair
- Write for the committee
- Convince
- Choose comments wisely
- Organize your rebuttal
- No tricks
- Thank the reviewers
- Don’t expect too much

Target Audience

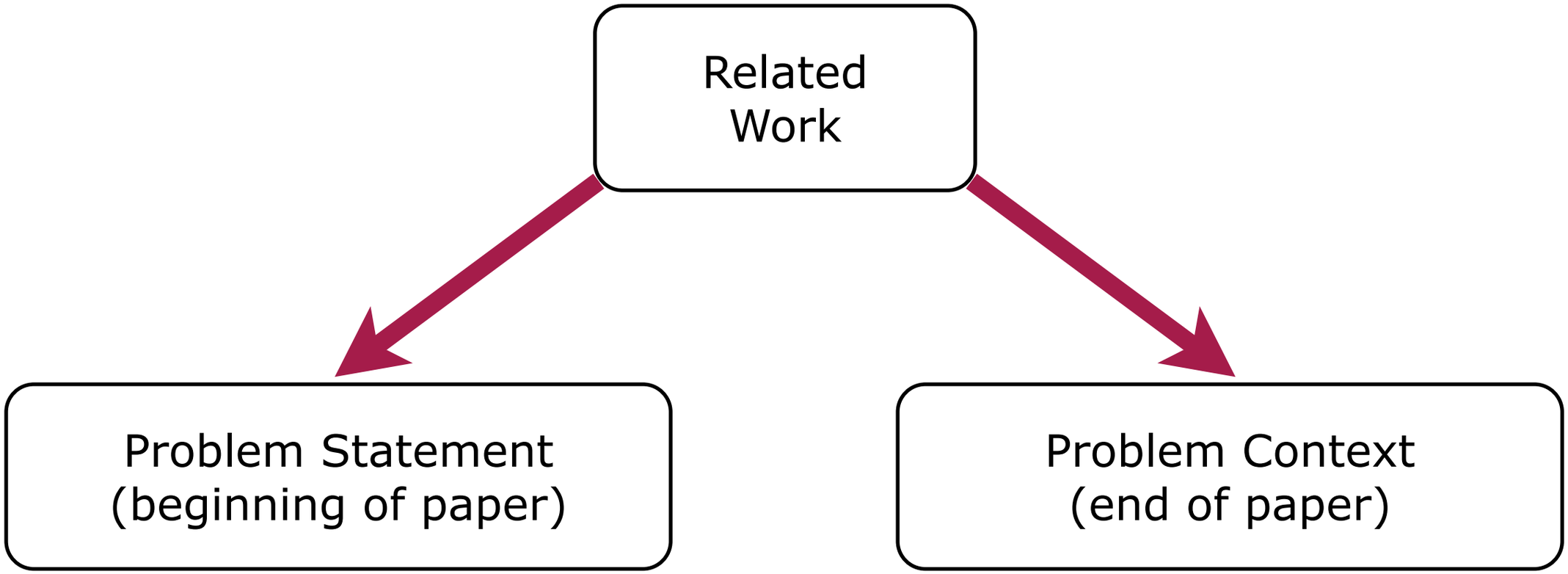


= preaching to the quire

= arguing the problem and inviting others to contribute

- Conferences: ICSE, ESEC/FSE
- Journals: TSE, TOSEM
- magazines: IEEE Software, IEEE Computer, Communications of the ACM

Role of "Related Work"



Other researchers define the research agenda

⇒ high entry barrier

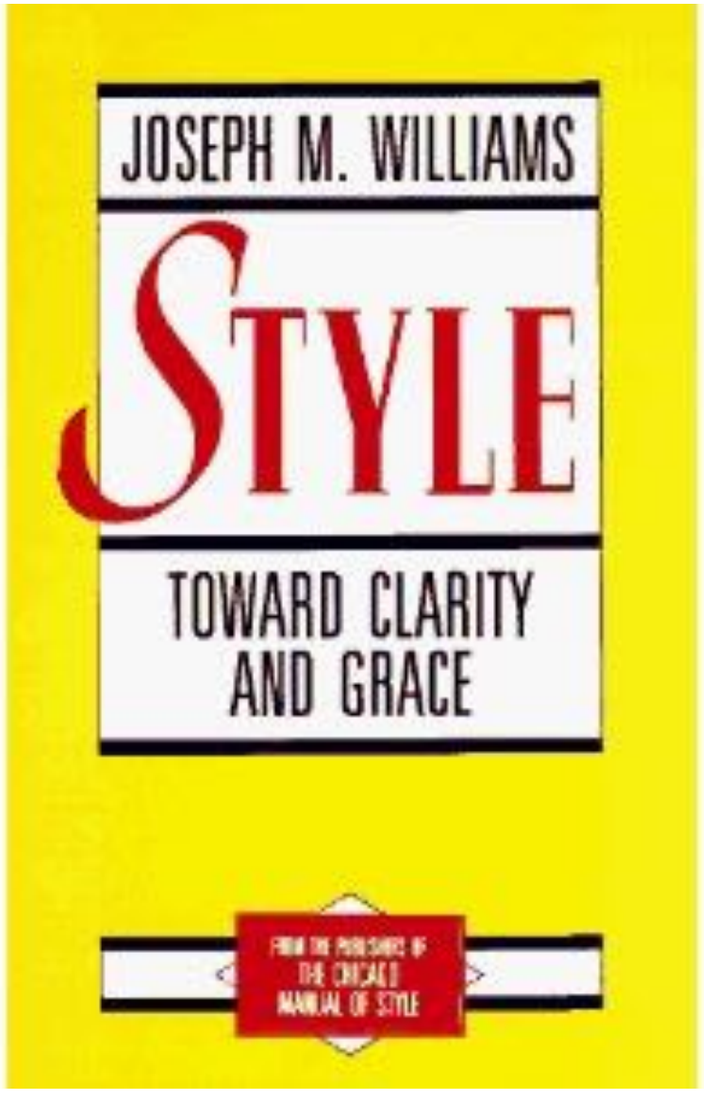
(for experts only)

Other researchers do complimentary work

⇒ crisp problem statement

(difficult to write)

Advice on writing



Style: Toward Clarity and Grace
Joseph M. Williams, Gregory G. Colomb

- guidelines
  + *refactoring* rules
- Give a man a fish and you feed him for a day. Teach a man to fish and you feed him for a lifetime.

Slide Deck - Full Tutorial

https://win.uantwerpen.be/~sdemey/Tutorial_ResearchMethods/