Announcements:

- Feedback on ETP Introduction from seminar leaders by tomorrow (27/11) late afternoon.
- Feedback seminar on Thursday 28/11 08:15.
  - Attending is optional if everything is “green”.
- Feedback on Academic English ca. 12 December by Shelley/Mikael and Brittany, possibility to ask questions in the feedback lecture 13 December

TDDD89

Research Methods
in Computer Science and Engineering

Christoph Kessler

LINKÖPINGS UNIVERSITET

What is a scientific research method?

- Try and error??
- Design, implement, evaluate?
- Acquire data, aggregate, visualise?
- Formulate theorems and prove them?
- …

Research Methods in Computer Science and Engineering

- **Theoretical/Analytical**
  - Defines and/or uses mathematical models of real or hypothetical systems
    - set theory, graphs, equations, constraints, probability, coding theory
  - Mathematically proves properties of abstract artifacts within the model
  - Typical for theoretical computer science
    (e.g. formal methods, complexity theory, type theory, coding theory, program analysis, …)
- **Design, Problem Solving, or Incremental Improvement of new technology**
  - Build a prototype to demonstrate/evaluate a new idea, or extend/improve a given system
  - Requires extensive experimental evaluation,
    comparing quantitatively to a well-chosen baseline
    to prove an improvement over the state of the art
  - Most algorithmic and computer systems / engineering thesis projects are here
- **Descriptive/Empirical**
  - Observe a phenomenon, describe it, compare, and extrapolate
  - Data analysis to statistically identify correlations and cause-effect relations
  - More typical for theses in software engineering, HCI, ML applications e.g. in healthcare
- **Systematic Literature Review / Systematic Mapping Study**

Each method type has its own specific techniques and specific threats to validity.

Let’s take a closer look…

Descriptive / Empirical Research Methods

Empirical Research: Different types of methods

- **Qualitative methods**: establish concepts, describe a phenomenon, find a vocabulary, create a model

Observations, interviews, ...: (Mostly) Qualitative data

Descriptive / Exploratory Research

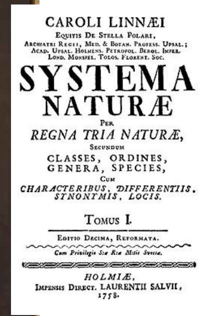
- **Quantitative methods**: make statistical analyses, quantify correlations, identify cause-effect relationships, ...

Surveys, controlled experiments, analysis: Quantitative data

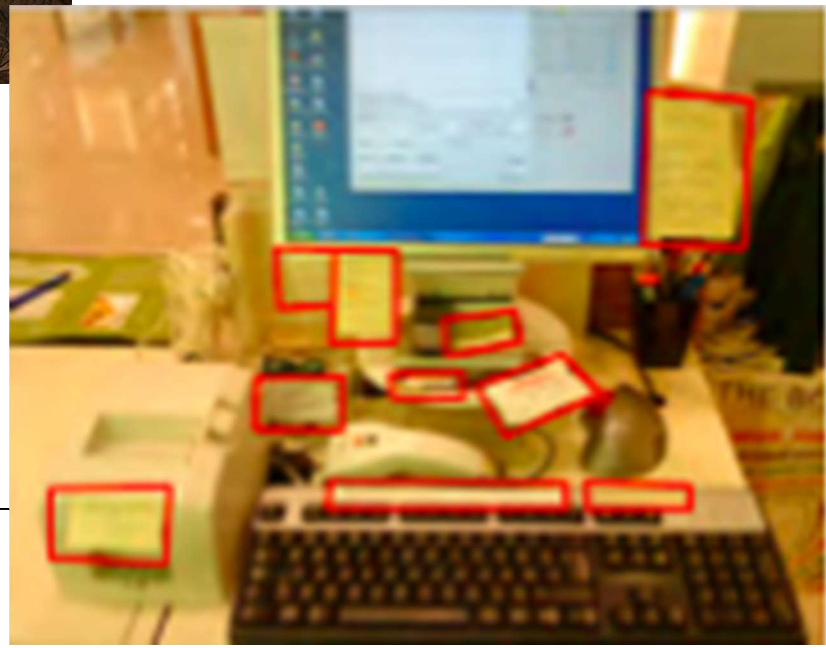
Explanatory Research

Empirical Research: Observations

By Friedrich Georg Weitsch - Karin März, Public Domain, https://commons.wikimedia.org/w/index.php?curid=61508



- Understand the context
- Write down what you see, hear, and feel
- Take pictures
- Combine with interviews
- Ask users to use systems if available



LINKÖPINGS UNIVERSITET

Empirical Research Methods

Techniques

Human-Centered Methods

- Observations
- Interviews
- Surveys
- Think-aloud sessions
- Competitor analysis
- Usability evaluation
- …

Experiment-Centered Methods

- Prototype / experiment design
- Experiments
- Quasi-experiments
- …

Also useful for the experimental evaluation in Design / Incremental Improvement based research

Interviews

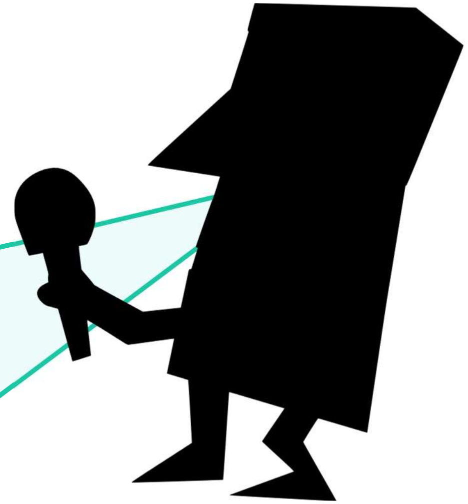
- Structured or unstructured?
- Group interviews (focus groups) or individual interviews?
- Telephone interviews

Hints:

- Use open-ended questions:
  ”Do you *like* your job?” vs ”*What do you think* about your job?“
- Active listening
- Record the interview
- Plan and schedule for that!

Four phases of an interview

1. Explain objectives of the interview and the study, ensure confidentiality
2. Introductory questions about the interviewee’s background
3. Main questions
   – based on research questions
4. Summarize the main findings to get feedback and avoid misunderstandings



P. Runeson, M. Höst: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14:131-164, 2009.

Interview analysis

- Transcribe or not?
- Categorize what has been said (encode)
- Easier for structured interviews

P. Runeson, M. Höst: Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14:131-164, 2009.

Surveys

- "A **survey** is a system for collecting information from or about people to describe, compare or explain their knowledge, attitudes and behavior."
  - A. Fink: *The Survey Handbook*, 2nd edition. SAGE, Thousand Oaks/London, 2003
- Gather qualitative and/or quantitative data
- **Questionnaire**
  - Keep it *short* and specific!
    - Not more questions than absolutely necessary
  - Anonymous, but also include some questions to collect relevant statistical data
    - for validation and correlation
  - Do a *dry-run* with a few colleagues before deploying at large scale
    - to avoid unclear questions / misunderstandings
- Choose a **sample group** that is *representative* for the **target group**
- Evaluate statistically to derive (possibly, explanatory) conclusions

Best questionnaire technology?
- Paper, Microsoft Forms, Google Forms, ...
- Depends on target group's preferences

Survey Example

Case: Find out about the current usage of programming languages for data-intensive HPC applications

- **Target group**: users / programmers in computational science and engineering, including data-driven methods using machine learning and data mining
- **Sample**: via members of a large EU project
- **Difficulties**: low number of answers, bias in the reply set of the sample group (too many CS professors) w.r.t. target group
  - Single-page Paper/Word/PDF form turned out to be most effective (10 questions, partly free-form)
  - Put effort in re-sampling, distributing, reminding
  - Be honest about impact of bias or small reply set

Survey

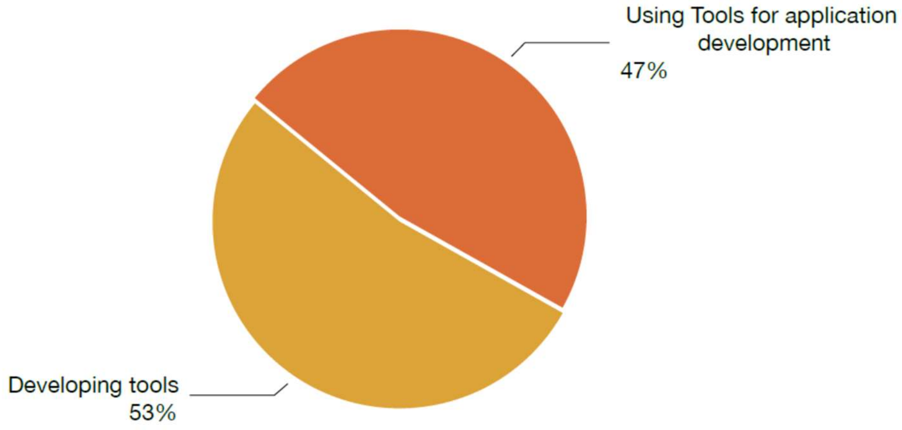
This survey is carried out within the scope of the article in preparation "***Programming Languages for Data-Intensive HPC Applications: a Systematic Mapping Study***", initiated by **Vasco Amaral** (Univ. Nova de Lisboa) and co-authored by the 19 contributors to the SLR/SMS study during the last 3 years.

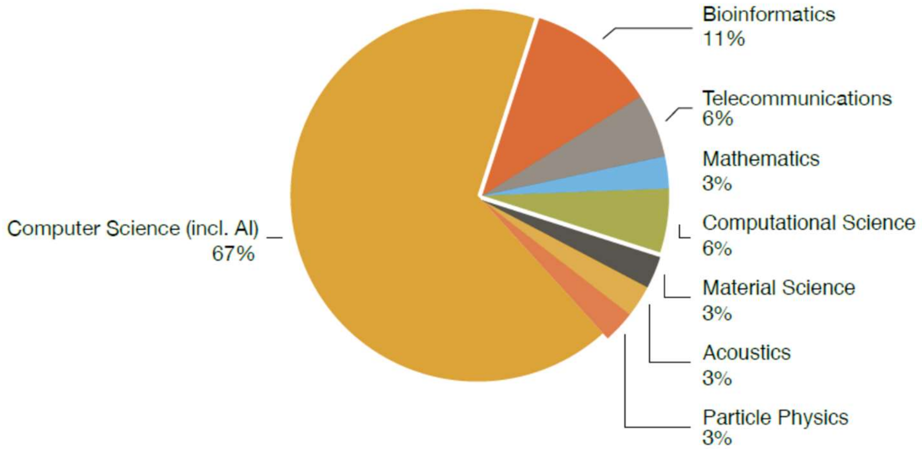
For complementation and validation of the literature review results, we would like to compare with the honest estimations of **experts** in data-intensive high-performance computing (that is, **you**). Please help us in collecting a sufficiently large and broad statistical basis for this validation by answering this survey form now at the Las Palmas meeting. It only takes 2-3 minutes, and the collected data will enable us to complete the article, producing a tangible outcome of *cHiPSet*.

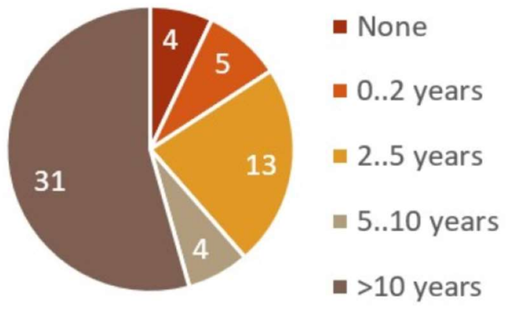
Please hand in the paper anonymously to **Christoph Kessler** or **Peter Kilpatrick** during the Las Palmas meeting. Many thanks in advance!

1. Were you involved in the SMS (Vasco's literature review program)?
   X Yes   O No
2. How long have you been working in High Performance Computing?
   O Not at all   O < 2 years   O 2 to 5 years   O 5 to 10 years   X > 10 years
3. In what areas of science or engineering have you worked?
   (e.g., computer science, bioinformatics, material science, telecommunications ...)
   computer science
4. Do your High Performance Computing related activities consist primarily of
   X *Developing* programming support tools, or   O *Using* existing programming tools?
5. How do you rate your level of technical knowledge about languages/frameworks for HPC?
   O Very Poor   O Poor   O Neutral   X Good   O Excellent
6. Which programming languages do you *use* for High Performance Computing?
   C, MPI, OpenMP, OpenACC, CUDA, OpenCL, Chapel
7. What are, in your view, the *key advantages* of these languages (in relation to the alternatives you know)?
   (this may include language properties, performance, programmability, etc.)
   control, performance, stability, flexibility
8. What *actually* made you use these languages? (if not already covered in 7.)
9. Which other programming frameworks (e.g., library-based) and tools do you *use* for HPC?
   MPI, OpenMP, OpenACC, CUDA, OpenCL
   (relation to question 6 is unclear to me)
10. Which other HPC programming languages / frameworks / tools do you know about (but do not use)?
    SAC, S-Net, X10, Hadoop, Spark, Tez, SkePU, Muesli, skeletons in general, BSP, FastFlow, gpH, StarPU, OmpSs, TensorFlow, Manticore, etc

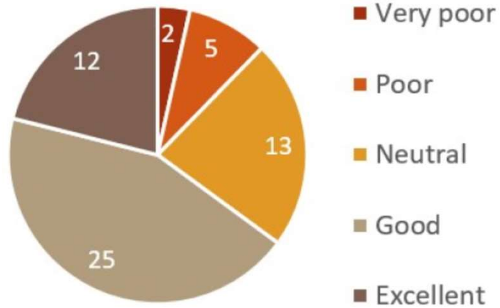
Survey Example, continued







14: Expert sample: Level of experience of working in HPC



sample: Self-estimation of technical knowledge in HPC pro-

Bias in sample detected thanks to the collected background information

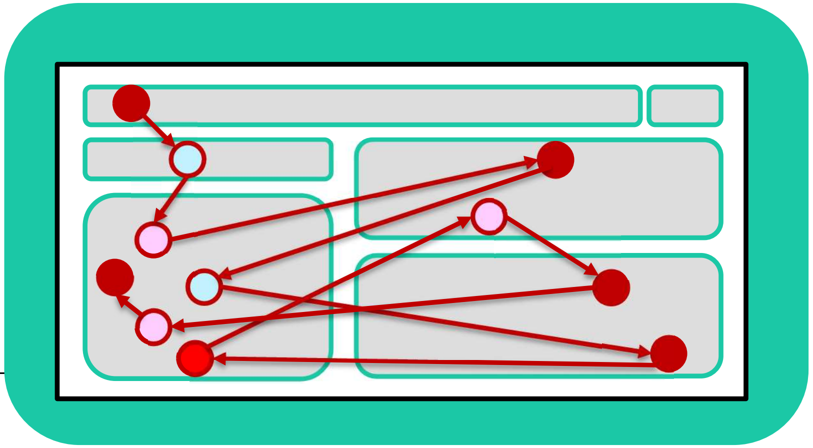
Collect more answers from actual HPC users to rebalance the bias (as far as possible)

LINKÖPINGS UNIVERSITET

V. Amaral *et al*.: Programming Languages for Data-Intensive HPC Applications: a Systematic Mapping Study. *Parallel Computing* 91, Elsevier, March 2020. DOI: 10.1016/j.parco.2019.102584

Usability Evaluation

- Heuristic evaluation – few persons, early in the development process
- System usability scale (SUS) →
- Post-Study System Usability Questionnaire (PSSUQ)
- Heuristic evaluations
  - with fewer test persons, done earlier in the development process
- Eye tracking
  - e.g. for GUI usability evaluation
- First-click Testing
- …



System Usability Scale (SUS)

Note the differences in positivity orientation

Recommended: **Alternating the interpretation of the scale** to enforce more reflection about the answer

		Strongly Disagree				Strongly Agree
1.	I think that I would like to use this website frequently.	☐	☐	☐	☐	☐
2.	I found this website unnecessarily complex.	☐	☐	☐	☐	☐
	I thought this website was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need assistance to be able to use this website.	☐	☐	☐	☐	☐
5.	I found the various functions in this website were well integrated.	☐	☐	☐	☐	☐
6.	I thought there was too much inconsistency in this website.	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this website very quickly.	☐	☐	☐	☐	☐
8.	I found this website very cumbersome/awkward to use.	☐	☐	☐	☐	☐
9.	I felt very confident using this website.	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this website.	☐	☐	☐	☐	☐

Usability Performance Measurement

- Task success
- Time (time/task)
- Effectiveness (errors/task)
- Efficiency (operations/task)
- Learnability (performance change)

Case Study

Example in Software engineering: “*Do weekly code reviews in ABC-type programmer teams improve the code quality of an XYZ-type application?*”

A **case study** investigates a **phenomenon** in its real-life **context**,

- with multiple sources of information,
- where the boundary between context and phenomenon may be unclear
- Uses predominantly **qualitative** methods to study a phenomenon

Different from *experiment*

- Experiments *sample* over the parameters being varied
  - more control, can e.g. identify interdependent factors
- Case studies *select* a parameter setting representing a *typical* situation

- Can, like experiments, be applied as a **comparative research strategy**
  - E.g., compare the effects of using a specific method, improvement etc. to a *baseline* method (e.g., project vs. comparable “sister project”)

P. Runeson and M. Höst, “Guidelines for conducting and reporting case study research in software engineering,” *Empirical Softw. Eng.*, vol. 14, pp. 131–164, Apr. 2009.

Experimental Studies

Experimental Study

- Control over the situation
- Manipulate behavior directly, precisely and systematically

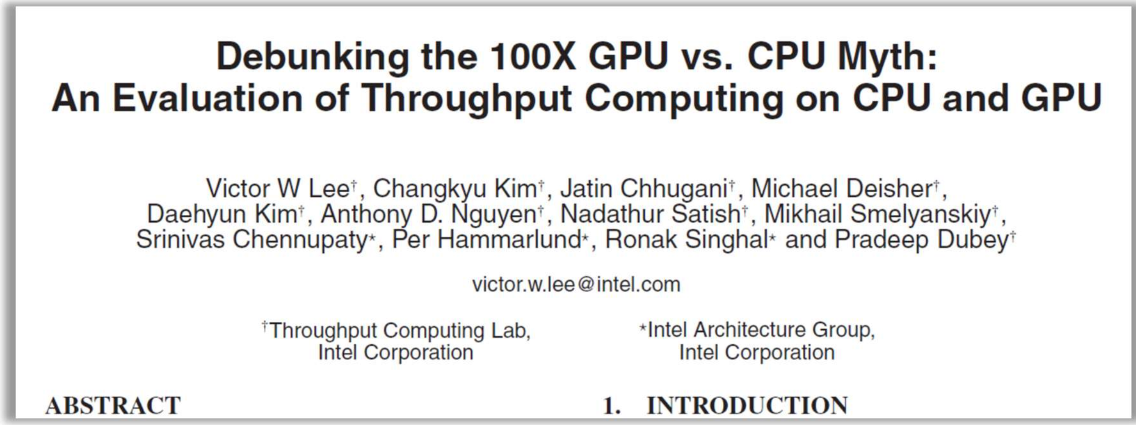
- Off-line experiment, e.g. in laboratory
- On-line experiment, e.g. in deployed system – more difficult

- Human-oriented experiment
  - needs test persons, less control, order-dependent, less deterministic
- Technology-oriented experiment
  - needs benchmark problems, more deterministic, more reproducible

Experimental Study

Possible experiment purposes:

- Confirm theories
- Confirm conventional wisdom →
- Explore relationships
- Evaluate the accuracy of models
- Validate measurements
- Quantitative comparisons or analyses:
  - "Where does technique ABC lead to better performance than technique DEF?"
  - "How well does this parallel program scale with the number of CPU cores?"



Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU

Victor W Lee[†], Changkyu Kim[†], Jatin Chhugani[†], Michael Deisher[†], Daehyun Kim[†], Anthony D. Nguyen[†], Nadathur Satish[†], Mikhail Smelyanskiy[†], Srinivas Chennupaty[*], Per Hammarlund[*], Ronak Singhal[*] and Pradeep Dubey[†]

victor.w.lee@intel.com

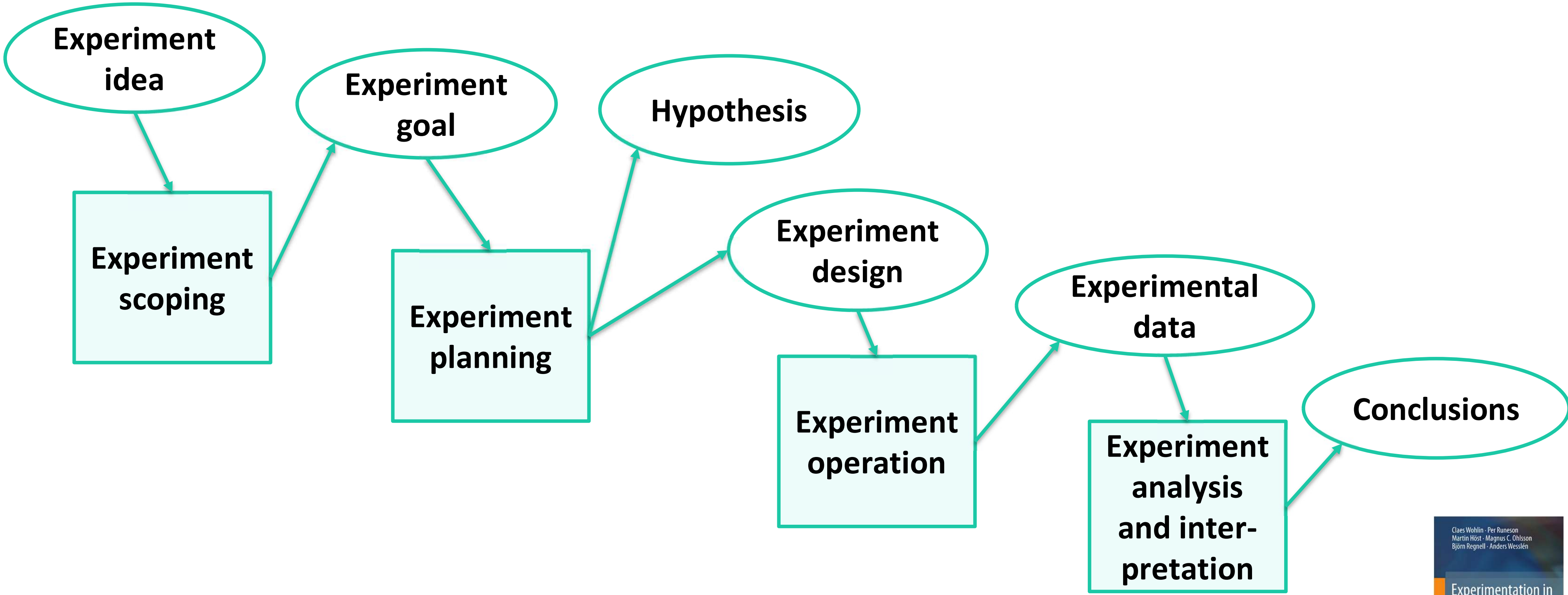
[†]Throughput Computing Lab, Intel Corporation

[*]Intel Architecture Group, Intel Corporation

ABSTRACT

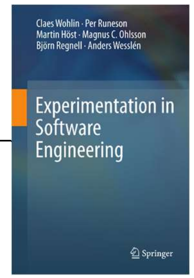
1. INTRODUCTION

Experimental study design



LINKÖPINGS UNIVERSITET

C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Berlin Heidelberg, 2012.



Experiment Goal

Template: [Basili, Rombach]

”Analyze <Object>
for the purpose of <Purpose>
with respect to their <Quality>
from the point of view
of the <Perspective>
in the context of <Context>"

Write it down!

	Example
Object: What is studied?	Product, process, resource, model, metric, …
Purpose: What is the intention?	evaluate choice of technique, describe process, predict cost, …
Quality: Which effect is studied?	effectiveness, cost, …
Perspective: Whose view?	developer, customer, manager, end user, …
Context: Where is the study conducted?	Subjects (personnel) and objects (artifacts under study)

V. Basili, D. Rombach: The TAME project: Towards improvement-based software environments. *IEEE Trans. Softw. Eng*. 14(6):758-773, 1988

Experiment Goal

Example [Wohlin *et al.*]

”Analyze perspective-based vs. checklist-based inspection techniques in SW requirements
for the purpose of evaluation
with respect to their
effectiveness and efficiency
from the point of view
of the researcher
in the context of M.Sc. and Ph.D. students reading requirements documents”

	Example
Object: What is studied?	Product, process, resource, model, metric, …
Purpose: What is the intention?	evaluate choice of technique, describe process, predict cost, …
Quality: Which effect is studied?	effectiveness, cost, …
Perspective: Whose view?	developer, customer, manager, end user, …
Context: Where is the study conducted?	Subjects (personnel) and objects (artifacts under study)

C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer Berlin Heidelberg, 2012.

Experimental Research Methods

Specific Threats to Validity

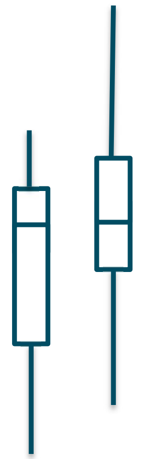
Method-Critical Questions	Engineering Aspect	Scientific Aspect
Can I trust your work?	Have you properly tested and evaluated your solution in different settings/scenarios?	Have you verified that you obtain the same data in different settings/scenarios?
Can I build on your work?	Can I run/create the same system somewhere else?	Can I replicate the results of the study?

Experiment Design Principles

For statistical analyzability of collected / experimental data:

- **Randomization**
  - All statistical methods used for analyzing the data require that the observations be from independent random variables
  - Randomization applies to the allocation of objects, subjects and order of test application
  - Random selection of sample can average out bias
- **Blocking** (grouping) subjects based on confounding factors
  - Eliminate systematically the effect of a factor that does have an effect on the result but is not considered central for the study,
  - e.g., distribute test persons with previous experience with a technique being studied
- **Balancing** – aim for equal group sizes in test and control groups
  - simplifies the statistical analysis of the data

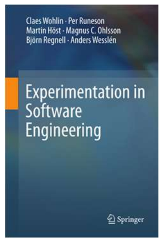
Statistical Evaluation of Data



- **See your statistics course book**
- A few hints anyway:
  - Use *boxplot* or *violine diagrams* to visualize distribution of data variation
  - Separate *correlation* and *causality*
  - Enough data points to statistically support a conclusion?
    - Unless $\geq$ 95% *confidence*, there is no correlation
  - Always include the *Null-Hypothesis* as a possible outcome!
    - Null-Hypothesis = there is no (statistically significant) difference between two data sets
      here: no statistically significant effect of the technique under study
    - Null-hypothesis significance testing (calculate *p*-value, ...)
      - Null-hypothesis can be rejected only if $p < 0.05$ → statistically significant effect
  - Threat to validity: HARKing = Hypothesizing After the Results are Known → (e.g., cherry-picking of benchmarks to show desired success)
    - Tempting, because negative results are often not accepted for publication

---

See also:
Chapter 10 of: C. Wohlin et al., *Experimentation in Software Engineering*. Springer, 2012.



HARKing

- Hypothesizing After the Results are Known

Figure courtesy of Dirk-Jan Hoek, used under CC 2.0 / original was cropped

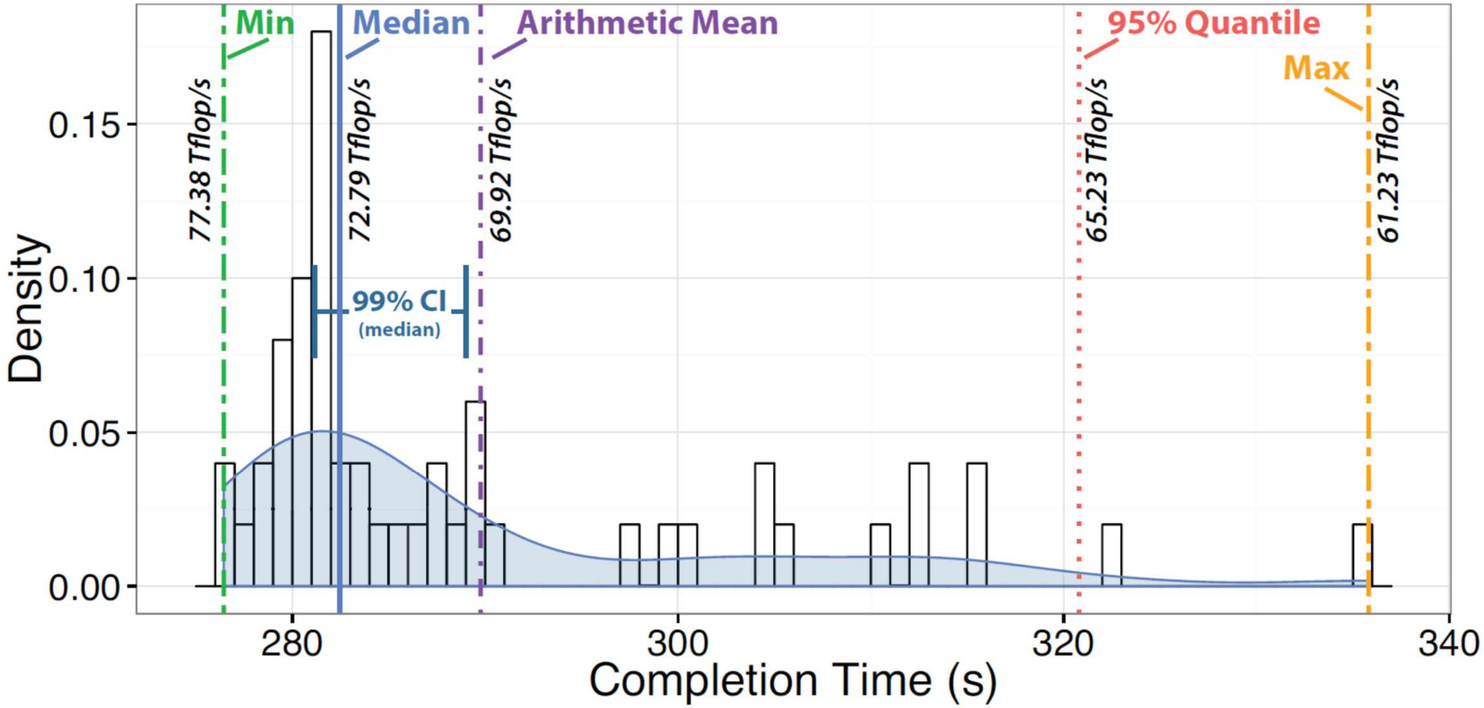
Experiments using Benchmarks

- A **benchmark** is a (usually, de-facto) standard *workload* (= program + input data) for the *comparison* of competing systems, components or methods according to specific characteristics, such as*
  - Relevance
  - Reproducibility
  - Fairness
  - Verifiability
  - Usability
- "To benchmark" = to *compare* by measurements for a standard workload.
- A single benchmark is not enough – there exist *benchmark suites* covering multiple application characteristics, e.g. SPEC for CPU benchmarking

*Adapted from: J. Kistowski, J. Arnold, K. Huppler, K. Lange, J. Henning, P. Cao: How to build a benchmark. Proceedings of 6th ACM/SPEC International Conference on Performance Engineering (ICPE), 2015.

Example: Measuring CPU time (and resulting performance)

Problem: On modern CPUs, execution time can vary considerably for the same input (due to, e.g., OS noise)



Example: Distribution of completion times for 50 runs of the HPL (High Performance Linpack) benchmark, from: T. Hoefler, R. Belli: Scientific Benchmarking of Parallel Computing Systems - Twelve ways to tell the masses when reporting performance results. Proc. SC '15, Nov. 2015, Austin, TX, USA. (c) ACM.

Evaluation Techniques in Machine Learning Research

- See your favorite ML textbook
  - E.g., E. Alpaydin: *Introduction to Machine Learning*, Second Edition, MIT Press, 2010

Cross-Validation

- To estimate generalization error, we need data unseen during training. We split the data as
  - Training set (50%)
  - Validation set (25%)
    - After having it used to choose the best model, it effectively becomes part of the training data
  - Test (publication) set (25%)
- Analogy from real life:
  - Exercise questions – training set
  - Exam questions – validation set
  - Problems in professional life – test set
- Resampling when there is few data

Evaluation in Deep Learning / DNN Research

Training Data Labeling and Augmentation

- **Where do we get labeled training data** for new problems?
  - Examples: Frame drivable area, bridges, motorcycles, humans on the road, traffic lights, car plates, ...
  - Usually need **human** labelers
    - expensive – this training data is the real IP of the companies, not the software
    - crowdsourcing in some cases, e.g. Oxford cats-and-dogs dataset [Parkhi *et al*. 2012] → →
- Risk with large DNNs and (too) few labeled training images: **Overfitting**
  - Overfitting = the DNN just memorizes the training set but it does not do a good job in generalizing classifications for previously unseen input
- **Training Data Augmentation**
  - applies scaling, rotation, translation, distortion, and other modifications to the set of available labeled training images

    → more training data, better generalization (and more work...)

    → more robust inference

LINKÖPINGS UNIVERSITET

C. Kessler: Programming Frameworks for Deep Learning. 2018.

Summary: Threats to Validity in Experimental Research

Type of Validity	Common Threats
Construct validity • Relation between theory and observation • Generalizability of experiment results based on underlying concepts/theory	• Premature experiment design (theory not entirely clear) • Incorrect setup of measurement equipment or unclear questionnaries • Unawareness of / ignoring accuracy issues, e.g. measurement noise • Unawareness of interactions between multiple experiments for a subject (test p.) • Errors in result-data logging, storage, postprocessing, visualization, interpretation • Positive effects observed and documented, but possible negative effects ignored

Wohlin et al.: *Experimentation in Software Engineering*. Springer, 2000

Summary: Threats to Validity in Experimental Research

Maybe useful for the ***risk analysis*** *in Seminar 5?*

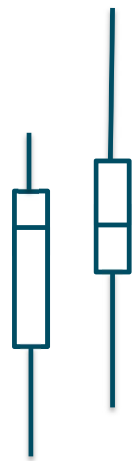
Type of Validity	Common Threats
Construct validity • Relation between theory and observation • Generalizability of experiment results based on underlying concepts/theory	• Premature experiment design (theory not entirely clear) • Incorrect setup of measurement equipment or unclear questionnaries • Unawareness of / ignoring accuracy issues, e.g. measurement noise • Unawareness of interactions between multiple experiments for a subject (test p.) • Errors in result-data logging, storage, postprocessing, visualization, interpretation • Positive effects observed and documented, but possible negative effects ignored
Internal validity • Causality in observed results (Absence of hidden factors impacting the results)	• Misinterpretation of causality direction (does A→B, or B→A, or X→A and B?) • Ignoring confounding factors • Biased selection of subjects etc. based on availability • Selection of subjects for control group and experiment groups is biased • Maturation of subjects (order/number of multiple experiments matters for the observed result for a subject) • Bias introduced by subjects with a conflicting interest in the study outcome • (Biased) drop-outs of subjects/systems/… from the study
External validity • Generalizability of experiment results to other environments than the one used in this study	• Selection of subjects/systems/settings/benchmarks/… is not representative for the target domain of the study • Selection interacts with the treatment or evaluation method • Results biased due to very recent events, e.g. security attack
Conclusion validity • Generalizability of experiment results based on statistical properties	• Established statistical methods are not used or applied wrongly • Null-Hypothesis not considered in evaluation • Low statistical power, low number of samples/test persons/data points

Wohlin et al.: *Experimentation in Software Engineering*. Springer, 2000

Final Remarks on Experimental Evaluation

Especially, for Design/Improvement based projects:

- Plan sufficient **time** for extensive evaluation.
- **Compare** quantitatively to the main competing algorithms/techniques.
- Use established **benchmark** problems representative for the application domain.
- Describe the experimental **setup** and measurement **method** thoroughly.
- Create ***readable* diagrams**.
  - Readable also on paper:
    - Font size should be between caption font size and normal text font size,
    - Not too light colors, …
  - Display measurement variations (e.g. boxplots), …
- **Archive** your program code used for the evaluation.
- Include (information about) own test programs/data etc.
  - e.g., in an appendix or on github, if OK with the company
- **Confidential** results to be de-identified before publication.



Systematic Mapping Studies and Literature Reviews

Systematic Mapping Study (SMS)

- *Broad* and *shallow* literature review
- Charts and structures a research area
- Discovers research trends
- Systematic search method, search scope, and criteria for inclusion / exclusion of literature items must be clearly specified
- May be implemented as a combination of automatic analysis (e.g. keyword-based) and manual reviewing with guiding questions

B. Kitchenham and S. Charters. **Guidelines for performing systematic literature reviews in software engineering**. Technical report, Ver. 2.3 EBSE, 2007.

K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. **Systematic mapping studies in software engineering**. *Evaluation and Assessment in Software Engineering*, vol. 8, pp. 68–77, 2008.

B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. **Systematic literature reviews in software engineering: a systematic literature review**. *Information and Software Technology*, 51(1):7–15, 2009.

Systematic Literature Review (SLR)

- *Narrow* and *deep* literature review for a well-defined specific area.
- Built on *focused questions* to aggregate evidence on a very specific goal
- Quality assessment of primary studies is more crucial
  - E.g., primary studies without empirical/experimental evidence should not be included.

What is a *Research Method Description*?

- ”To implement a Flux controller, I first needed to learn about Flux”

??? Don’t write a diary!

Write what convinces someone that you have done a good job:

”The Flux controller was evaluated using the Flux controller evaluation protocol [1]”

Research Methods – Concluding Remarks

- Know your research method(s), their specific techniques and validity threats
  - Theoretical Research
  - Design/Prototyping/Incremental Improvement based Research
  - Empirical Research
  - Statistical Data Analysis based Research
  - Experimental Research
  - Systematic Literature Studies
- Cite (and read) a few relevant methodology papers
  to show that your work follows the established practices in the field
- Critically evaluate your research method choice(s)
  in the Discussion/Conclusion part of your thesis
- Plan sufficient time for data collection (interviews, surveys, experiments, …)
  and evaluation

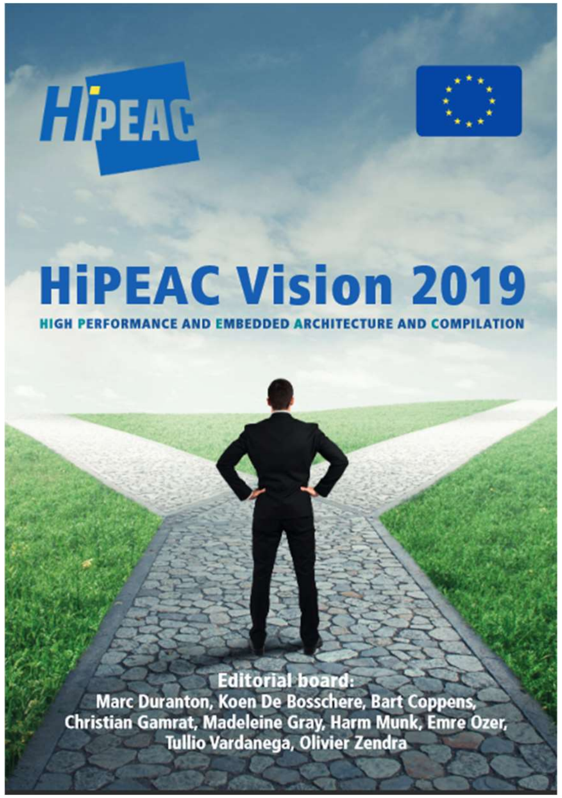
Many thesis projects require a combination of several of these

TDDD89
“Your Work In a Wider Context”

→ Seminar 4

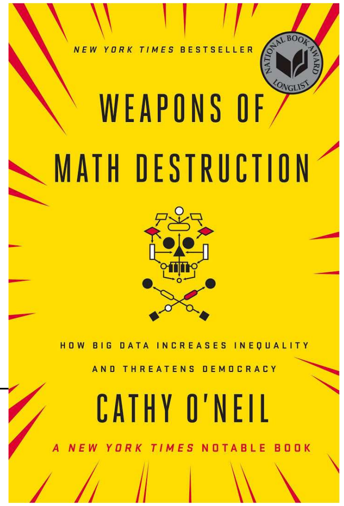
LINKÖPINGS UNIVERSITET

Resources



- Section 2.6 (The Societal Dimension) of the ***HiPEAC Vision 2019***
  https://www.hipeac.net/vision/2019/

- C. O'Neil: *Weapons of Math Destruction - How Big Data Increases Inequality and Threatens Democracy*. New York, NY, USA: Broadway Books, 2017.



Your work in a wider context

Why do we as humans have to solve this problem?

United Nations Development Programme www.undp.org
2015 Sustainable Development Goals

Your work in a wider context

Direct effects

System effects

Social effects

Stress, Awareness, Trust, Engagement

Economic effects

Job opportunities, Market dynamics

Ecological effects

Emissions, Resource use

C. Becker, R. Chitchyan, L. Duboc, S. Easterbrook, B. Penzenstadler, N. Seyff, and C. C. Venters, “Sustainability design and software: the Karlskrona manifesto,” in IEEE International Conference on Software Engineering (ICSE), vol. 2, pp. 467–476, IEEE, 2015.

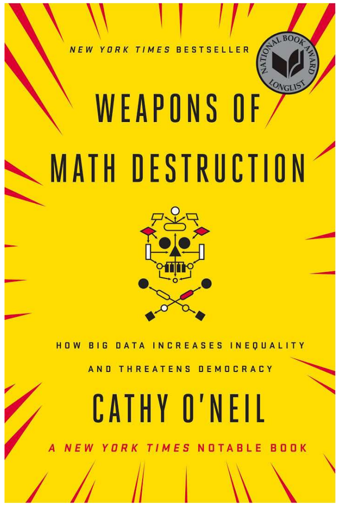
Example: The Effects of Big Data and Machine Learning

- A level 1 non-linear, chaotic dynamic system:
  the climate system, turbulence, population dynamics
- A level 2 chaotic system: Human activities such as stock markets

System behavior (model) may be based on (biased) training data.
System behavior affects reality, which generates new training data, which confirms the biased model
→ bias at system deployment reinforced by system's behavior

My behavior implies the system's behavior and vice versa

Stuff I like ⇄ My inputs to search engine



Example

Stocks shall always be traded based on quantitative information about prices

The most rational prices should be derivable from a mathematical model

What does reality say about this?

Option Pricing Model by Black-Scholes 1973:

$$C = S \cdot N(d_1) - Xe^{-r\tau} \cdot N(d_2)$$

$$d_1 = \frac{\ln\left(\frac{S}{X}\right) + (r + \sigma^2/2)\tau}{\sigma\sqrt{\tau}}; \qquad d_2 = d_1 - \sigma\sqrt{\tau}$$

The Pricing of Options and Corporate Liabilities - EconPapers
https://econpapers.repec.org › RePEc:ucp:jpolec:v:81:y:1973:i:3:p:637-54
by F Black - 1973 - Cited by 38639 - Related articles
The Pricing of Options and Corporate Liabilities. Fischer Black and **Myron Scholes** · Journal of Political Economy, **1973**, vol. 81, issue 3, 637-54. Date: **1973**

Example (cont.)

Constructing a Market, Performing Theory: The Historical Sociology of a Financial Derivatives Exchange[1]

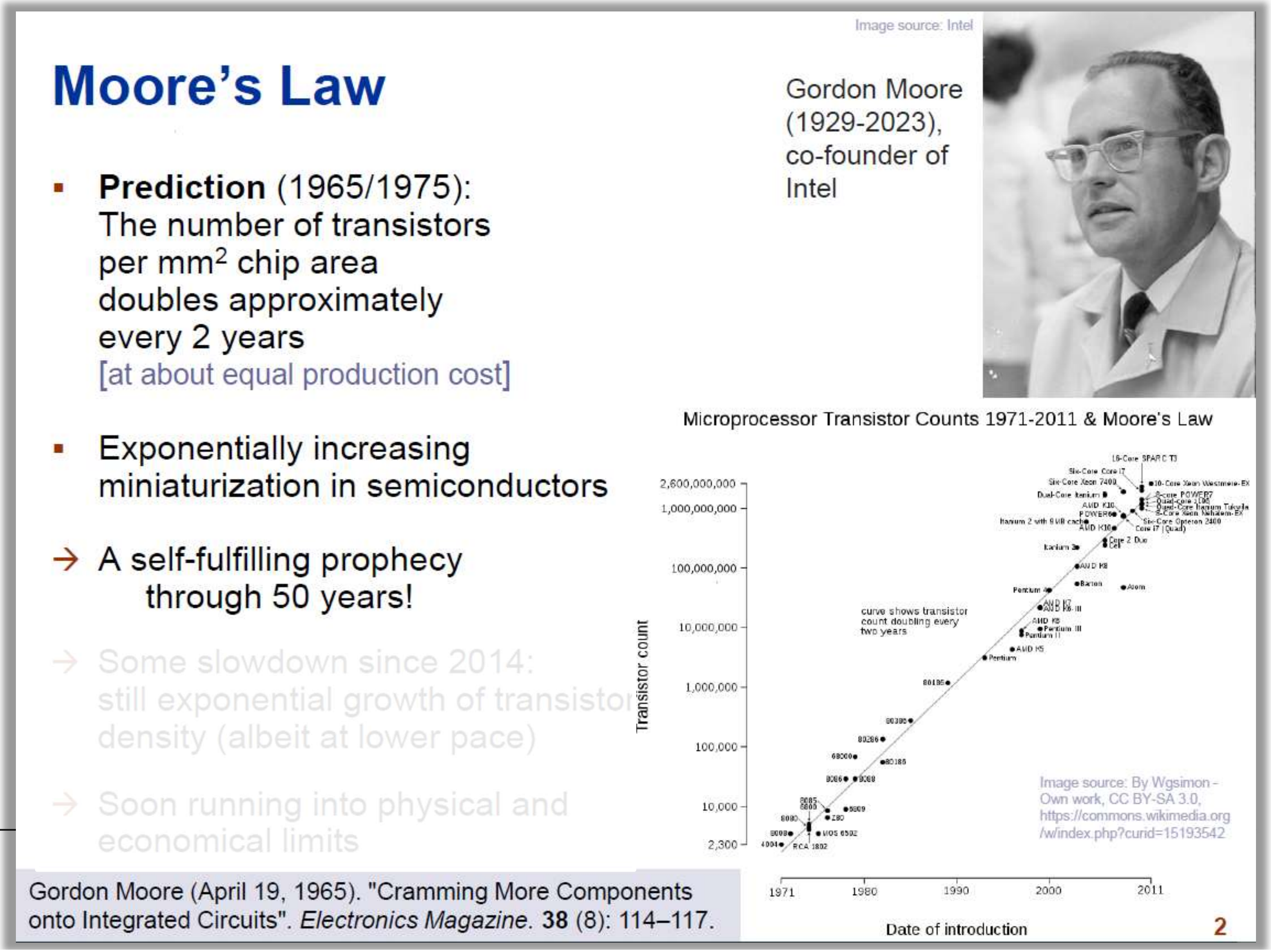
the 20th century. Option pricing theory—a "crown jewel" of neoclassical economics—succeeded empirically not because it discovered preexisting price patterns but because markets changed in ways that made its assumptions more accurate and because the theory was used in arbitrage. The performativity of economics, however,

→ **Research can create self-fulfilling prophecies that eventually interfere with the target of research itself!**

D. MacKenzie, Y. Millo: Constructing a market, performing theory: The historical sociology of a financial derivative exchange. *American Journal of Sociology* 109(1): 107-145, July 2003.

Self-Fulfilling Prophecies in Computer Engineering …

- Example ?



Slide from TDDD56 (2023), C. Kessler, Linköping University

Further Examples

- “Automating the classification of fMRI images for oncologists”
- “Directed media content through topic modeling”

Acknowledgments

Some slides are based on a previous lecture by Ola Leifler, IDA, Linköping University

Literature (1)

On specific types of research methods in Software Engineering:

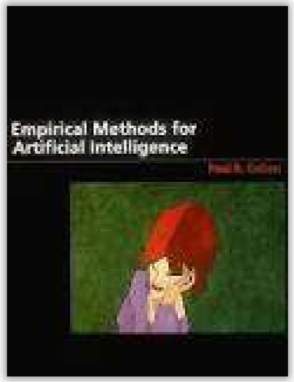
- P. Cohen: ***Empirical Methods in Artificial Intelligence***. The MIT Press, 1995.
- C. Wohlin *et al.*: ***Experimentation in Software Engineering***. Springer, 2012.
- P. Runeson *et al.*: ***Case Study Research in Software Engineering***. John Wiley & Sons, Ltd., 2012.

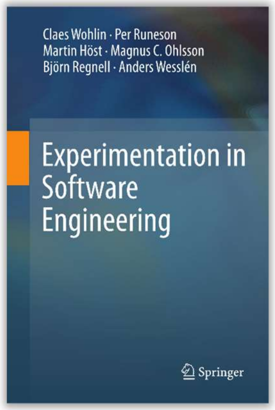
On experimental evaluation in HPC:

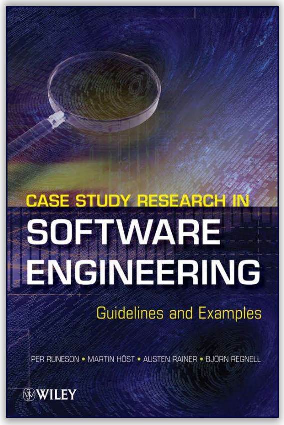
- T. Hoefler, R. Belli: **Scientific Benchmarking of Parallel Computing Systems** - Twelve ways to tell the masses when reporting performance results. Proc. SC '15, Nov. 2015. ACM.

On (lack of) statistical evaluation in empirical computer science:

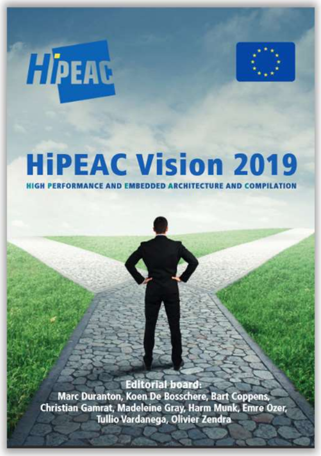
- A. Cockburn, P. Dragicevic, L. Besancon, C. Gutwin: **Threats of a replication crisis in empirical computer science**. *Communications of the ACM* 63(8), Aug. 2020. DOI: 10.1145/3360311







Literature (2)



On societal impact of IT:

- Section 2.6 of the ***HiPEAC Vision 2019***, https://www.hipeac.net/vision/2019/

And more on the perils of using opaque models and Big Data:

- C. O'Neil, ***Weapons of Math Destruction - How Big Data Increases Inequality and Threatens Democracy***. New York, NY, USA: Broadway Books, 2017.