Freie Universität Berlin

Course "Empirical Evaluation in Informatics"

Introduction

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

- The notion of Empirical Evaluation
- Theory, Construction, Empiricism
- Status of empiricism in Informatics
- Hypothetical examples
- Quality criteria:
  - credibility
  - relevance
- Note on scale types

"Empirische Bewertung in der Informatik"

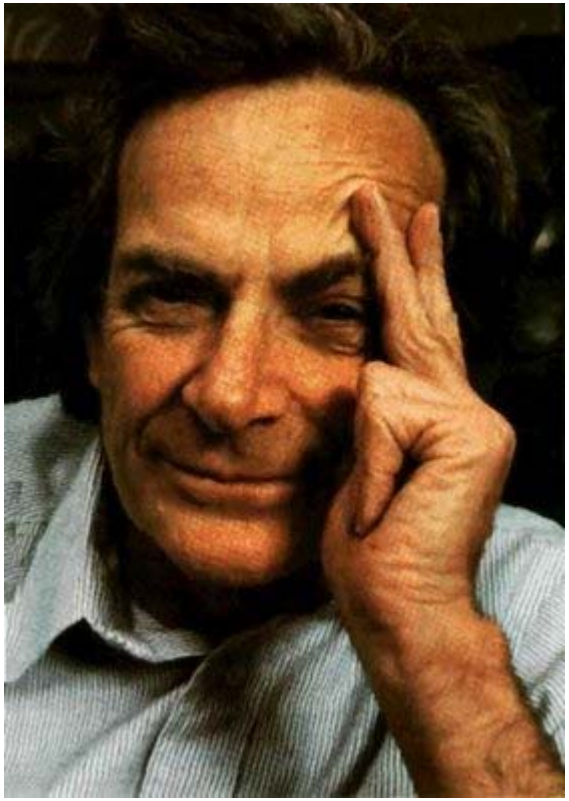
Einführung

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

- Begriff
- Theorie, Konstruktion, Empirie
- Status der Empirie in der Informatik
- Hypothetische Anwendungsbeispiele
- Qualitätsmaßstäbe:
  - Glaubwürdigkeit
  - Relevanz
- Hinweis: Skalentypen

Science

- Science is the belief in the ignorance of experts.
  - Richard Feynman (1918-1988)
    (The pleasure of finding things out,
    1999, p.187)



  - (Ignorance: Unwissen)

Technology

Freie Universität Berlin

- For a successful technology,
  reality must take precedence over public relations,
  for nature cannot be fooled.
  - Richard Feynman
    (last sentence of the Rogers Commission Report into the Challenger Crash, Appendix F: *Personal Observations on the Reliability of the Shuttle*)
    - http://science.ksc.nasa.gov/shuttle/missions/51-l/docs/rogers-commission/Appendix-F.txt

- (Things are as they are.
  Just claiming something does not make it true.)

"Empirical" / "empirisch"

Freie Universität Berlin

- Based on observation
  - (greek-latin origin)

- As opposed to being based on
  - theoretical considerations
  - intuition
  - random selection

"Evaluation" / "Bewertung"

Freie Universität Berlin

- "to evaluate": auswerten, bewerten

- Assigning measures of goodness to something
- The purpose is typically making some kind of decision:
  - deciding **yes or no**
    - Should I do it or not (in my context)?
    - e.g. starting or stopping a project, introducing a technology or not, etc.
  - **selecting** among solution candidates
    - Which one is best for my purposes?
    - e.g. systems, methods
  - **understanding** characteristics and priorities
    - What characteristics does a certain method/tool have in my context?
    - Which of these are most relevant for me?
    - e.g. in business process automation, user interface design, etc.

"Informatics" / "Informatik"

Freie Universität Berlin

- The science and engineering of information and information processing systems

- Very broad area

Note:

- Many of the same principles presented in this course can be applied to other areas as well
- Hence, this course is relevant far beyond where software is created or selected

Partitioning Informatics: Technical, Applied, Theoretical

Freie Universität Berlin

Work in Informatics is often discriminated into being either

- Technical ("Technische Informatik")
  - having to do more or less closely with hardware
- Applied ("Praktische Informatik")
  - having to do mostly with software
- Theoretical ("Theoretische Informatik")
  - having to do mostly with mathematics
- Furthermore: Applications ("Angewandte Informatik")
  - having to do mostly with software in actual usage contexts

- Problem: Discrimination hardly relevant for practitioners
- Problem: The boundaries have long disappeared

Partitioning again: Theory, Construction, Empiricism (T,C,E)

Freie Universität Berlin

A better discrimination would be **by work method**:

- Theory
  - produces formalisms, derives results about them, revolves around logical issues
- Construction
  - produces systems designs, constructs systems, revolves around practical issues
- Empiricism
  - produces observations of systems and interprets them, revolves around behavior in and of the real world

- At any one time, any work in Informatics is primarily in only one of these modes
  - whether practitioner work or research work
  - but good work switches mode frequently

T, C, E: What they are used for

Freie Universität Berlin

(Our focus is solving practical Informatics problems)

- Theory:
  - structuring and understanding a domain
  - if done well, can much simplify construction
    - "There is nothing more practical than a good theory"
- Construction:
  - build a useful technical artifact, such as a software system
- Empiricism:
  - in an early phase: Understanding requirements
  - in a later phase: Understanding the characteristics of a system
    - This is the topic of the present course!

Example 1: Algorithms

Freie Universität Berlin

1. Theory
   - Specify the problem to be solved
     - e.g. linear programming: minimize linear function given constraints
   - Specify an algorithm for solving it (e.g. simplex algorithm)
   - Maybe prove the algorithm correct, etc.

2. Construction
   - Implement the algorithm as a concrete program
     - often much longer than the theoretical algorithm because of optimizations, input/output, limitations of machine arithmetic, error handling, external interfaces, etc.

3. Empiricism
   - Determine actual characteristics of the program for different kinds of inputs
     - execution time, memory behavior, etc.
     - for heuristic or approximation algorithms: quality of results

Example 2: Software design methods

Freie Universität Berlin

1. Empiricism
   - Determine the weaknesses of current design methods
2. Theory
   - Maybe define some new terminology
   - Maybe pose new design principles
3. Construction
   - Formulate a new design method
   - Perhaps construct support tools
4. Empiricism
   - Evaluate the behavior of the method for concrete problems
   - Probably in comparison to other methods

Informatics until recently

- Until the 1990s, Informatics research publications were often short on empirical evaluation
  - they often provided designs of systems
  - and claims for their properties
  - but little or no no data on actual behavior

- In comparison, other engineering fields were much better in this respect

- Tichy, Lukowicz, Prechelt, Heinz: "Experimental Evaluation in Computer Science: A quantitative study", Journal of Systems and Software 28(1), 1995.

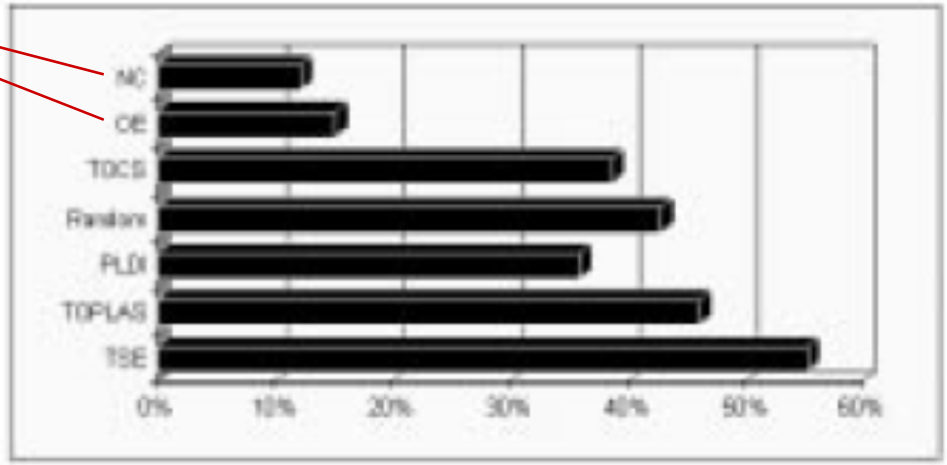


Figure 5: The percentage of design & modeling articles without any experimental evaluation.

Informatics today

Freie Universität Berlin

- This situation has become a lot better
- It is now generally understood that proposed systems and methods need to be evaluated empirically

- Compared to research, practitioners' work has always been more empirical
  - However, evaluation was often very implicit, hardly conscious and often not very systematical.
  - Systematic empirical evaluation is becoming more and more common there as well

A glimpse of the improvement

- M. Zelkowitz and D. Wallace: "Experimental Models for Validating Technology", IEEE Computer 31(5), May 1998.

- Categorizes 612 research articles in software engineering (from 1985, 1990, 1995) according to the empirical evaluation method that they used (if any)

- 2 of the 14 categories describe articles providing
  - no proof ("no experimentation") or
  - only visibly biased proof ("assertion")
  - 1985: 67% of the articles were in one of these two
  - 1995: 47% of the articles were in one of these two
  - still far from great, but a big improvement
  - has presumably continued to improve

Frequency of evaluation methods (in SW Eng.)

Freie Universität Berlin

Table 2. Classification of 612 evaluated papers.

Method	1985 ICSE	1985 *IEEE Software*	1985 TSE	1990 ICSE	1990 *IEEE Software*	1990 TSE	1995 ICSE	1995 *IEEE Software*	1995 TSE	Total
Not applicable	6	6	3	4	16	2	5	7	1	50
No experimentation	16	11	56	8	8	41	10	3	14	167
Replicated	1	0	0	0	0	1	1	0	3	6
Synthetic	3	1	1	0	1	4	0	0	2	12
Dynamic analysis	0	0	0	0	0	3	0	0	4	7
Simulation	2	0	10	0	0	11	1	1	6	31
Project monitoring	0	0	0	0	1	0	0	0	0	1
Case study	5	2	12	7	6	6	4	6	10	58
Assertion	12	13	54	12	19	42	4	14	22	192
Field study	1	0	1	0	0	1	1	1	2	7
Literature search	1	1	3	1	5	1	0	3	2	17
Legacy data	1	1	2	2	0	2	1	1	1	11
Lessons learned	7	5	4	1	4	8	5	7	8	49
Static analysis	1	0	1	0	0	0	0	0	2	4
Yearly totals	56	40	147	35	60	122	32	43	77	612

Examples

- Let us look what empirical evaluations may be used for
- and what approach we might use in each case:

  - Scenario 1: Introducing inspections in a SW project organization
  - Scenario 2: Switching the development world
  - Scenario 3: Selecting an Application Server product

In each case, we look at:

- What are the results of interest?
- What are the constraints?
- What are the most promising evaluation approaches?

Scenario 1: Introducing inspections

Freie Universität Berlin

- A SW organization is thinking about introducing inspections into their process
- Constraint: is an IT service company doing information systems projects for varying customers
  - requirements inspections, design inspections, code inspections
- Interest:
  - which and how many defects will be found?
  - cost of inspections
  - how much effort is saved by finding the defects early?
    - return-on-investment (ROI)

(Scenario 1)
Code Inspections

Freie Universität Berlin

- Possible approach:
  - inspect a number of modules; do not fix the defects found
  - measure testing and debugging cost of the same modules
  - (A **quasi-experiment** approach)

- Difficulties
  - What if author and tester are the same person?
    - unfair disadvantage for inspectors
  - What if measurement is too difficult?
    - due to interruptions, mixing with other tasks (e.g. debug with fix), etc.

- Other approaches
  - search the literature on this subject
  - controlled experiment: many people inspect or test the same program

(Scenario 1)
Requirements and design inspections

Freie Universität Berlin

- Approach:
  - ???

- Difficulties:
  - Cannot afford not to fix problems found
  - If we fix them, cannot measure would-be cost for comparison
  - Estimates may be very imprecise

Scenario 2:
Switching from C to Embedded Java

Freie Universität Berlin

- A SW organization is producing car control devices
  - motor control, ABS, ESP, distance control, etc.
- Development is done in C (since many years)
- The organization considers switching to Embedded Java
  - object-oriented rather than procedural,
    garbage collection rather than manual memory mgmt,
    VM rather than bare machine, etc.
- Interest:
  - What/how much would our engineers have to learn?
  - What would become better or easier?
  - What would become worse or more difficult?
  - What risks are involved?

(Scenario 2)
A laboratory study approach

Freie Universität Berlin

- Approach:
  - Select one team, let them build one prior system again
    - or a part of it
  - Evaluate relative to the experience from the original project
  - (A **case study** approach)

- Problems:
  - The team may be too inexperienced with Embedded Java
  - Many important differences can hardly be observed, because fundamental problems with requirements or design need not be solved again
    - (Alternative: build the system in both languages side-by-side; also a case study approach)
  - The retrospective view of the previous project may be too imprecise

(Scenario 2)
Field study approach

Freie Universität Berlin

- Approach:
  - Introduce Embedded Java in a (real) pilot project
  - Compare to 'usual' projects
  - (A **case study** approach)

- Problems:
  - The issues may be too project-specific for a sound comparison
  - All participants may be too busy doing the project
    - and will perhaps not really create a useful comparison
  - High risk of project failure (because of lack of experience)
    - Compensating this risk may result in an overly well-staffed, well-budgeted project and produce overly optimistic results

Scenario 3: Selecting an Application Server product

Freie Universität Berlin

- A SW organization building distributed information systems wants to start using a Java Enterprise Edition (Java EE) application server

- An application server is a very complex middleware product (a programming platform)
  - Several vendors offer such servers
  - In principle, they all conform to the Java EE standards and should be interchangeable

- Interest (for each product):
  - How scalable and efficient is it?
  - How stable, robust, easy-to-use is it?
  - How will it evolve in the future?

(Scenario 3)
How scalable and efficient is it?

Freie Universität Berlin

- Possible approach:
  - Use several existing applications
  - Set them up for load test measurements
  - Measure and compare across the various Application Servers
  - (A **benchmarking** approach)

- Problems:
  - Where to get the applications from
    if this is the first application server we will use?
  - How to make sure each server is configured well?

(Scenario 3)
How stable, robust, easy-to-use is it?

Freie Universität Berlin

- Possible approach:
  - Write one application on each server, then port it to all others
  - Protocol all interesting events (defects detected, usability and documentation traps, etc.)
  - Mostly qualitative rather than quantitative
  - (A **case study** or **quasi-experiment** approach)

- Problems:
  - This is a huge effort
  - The results depend critically on the (prior?) knowledge of the engineers
    - many will not apply to routine usage

(Scenario 3)
How will it evolve in the future?

Freie Universität Berlin

- Answering this question requires a market analysis
- It cannot be answered technically-empirically

Lessons learned from examples

Freie Universität Berlin

- There is a range of different empirical techniques
  - e.g. controlled experiment, quasi-experiment, case study, benchmarking, literature study
- Each evaluation problem has different characteristics
  - Often suggesting a particular technique
  - Or several techniques, with different tradeoffs
- There are usually some difficulties in an evaluation problem that cannot be fully solved
  - Most evaluations are only an approximation to the ideal one
  - But good approximations are *much* more useful than bad ones

Quality measures for empirical evaluations

Freie Universität Berlin

The two primary quality dimensions of empirical evaluations are:

- **Credibility**
  - How trustworthy are the results?
- **Relevance**
  - How interested are we in these results?
    How beneficial is it to have them?

- Subsequently we will often assume we are looking at a technical report about the study

Where does credibility come from?

Freie Universität Berlin

1. Authors are open towards any result
   - rather than "We will now show that our new X is superior."
2. Setup is described in detail and is easy to understand
3. Work has been performed carefully
4. Description discusses the limitations of the evaluation
   - rather than glossing over even its obvious flaws
5. There is no leap-of-faith or jumping to conclusions
6. Purpose and results are clear
   - rather than vague or, for results, weak
7. Results are easy to grasp ("anschaulich")
   - rather than abstract or contrived

- (Some of these aspects are known as "Internal Validity")

Where does relevance come from?

Freie Universität Berlin

1. The target of the evaluation (i.e. the question asked) is of sufficient interest
   - rather than overly specialized

2. We can generalize the results from the specific setup of the evaluation to those situations where we want to apply them
   - this is also known as "External Validity"

Qualitative vs. quantitative

Freie Universität Berlin

- Empirical evaluations need not always be quantitative
  - i.e. counting and measuring something; providing numbers, graphs and calculations
- They can also be qualitative
  - describing non-quantifiable characteristics; describing contexts; describing events and their consequences; providing subjective judgements obtained from relevant people
- or can combine both approaches
  - which is almost always a good idea

Qualitative vs. quantitative: Examples

Freie Universität Berlin

E.g. for applying a design method:

- **Quantitative questions**:
  - How long does it take?
    - time in minutes
  - How many mistakes are made in the process?
    - number of changes during work
  - How good is the result?
    - number of defects

- Corresponding **qualitative questions**:
  - What activities is the work time spent on?
  - Which kinds of mistake happen frequently? Why?
  - What are the typical kinds of flaws in the result?
    Why do they occur?
    What might be done to prevent them?

A word of warning: scale types

Freie Universität Berlin

- An advantage of quantitative data is that it can be processed using mathematical operations
  - This can allow easy summarization or can provide additional insights
- However, not all computations are valid for all kinds of quantitative data
  - The data must be on a sufficient scale type
    - otherwise, an operation may be invalid and not make sense
  - See the next slide

The scale types

Freie Universität Berlin

- Categorical/categorial scale (nominal scale)
  - Qualitative data: The values are just names
  - Example: Design method A, design method B
- Ordinal scale (rank scale)
  - Ordered nominal data: One value is larger than another, but we cannot characterize the size of the difference
  - Example: very good, good, OK, not so good, bad
- Difference scale (interval scale)
  - We can compute differences, but 0 is not equal to 'nothing'
  - Example: degrees centigrade
- Ratio scale ("Verhältnisskala")
  - We can compute ratios: 20 is twice as much as 10
  - Most physical quantities, degrees Kelvin
- Absolute scale: 1 is also special (counting)

What often goes wrong with scale types

Freie Universität Berlin

- *"Oh, 20 degrees. That's twice as warm as yesterday."*
  - A difference scale is not a ratio scale
  - 20 degrees centigrade is 293 Kelvin.
    That is only 3.5% more than 283 Kelvin.

- When something qualitative is measured using an ordinal scale
  - e.g. *"How well did you like using the tool?"*
    very well, well, OK, not so well, did not like it
  - Often such scales are coded with numbers: 5, 4, 3, 2, 1
  - Wrong (however tempting and common it may be):
    "average satisfaction was 3.8"
  - Even worse: "*average satisfaction in group B was 30% higher than in group A"*
    - This is utter nonsense!
    - Assume you would have coded using 2, 1, 0, -1, -2?

Primary nonsense candidates

Freie Universität Berlin

- There are a number of important attributes in informatics for which no good ratio scales are known
- These are frequent places of scale type mis-use

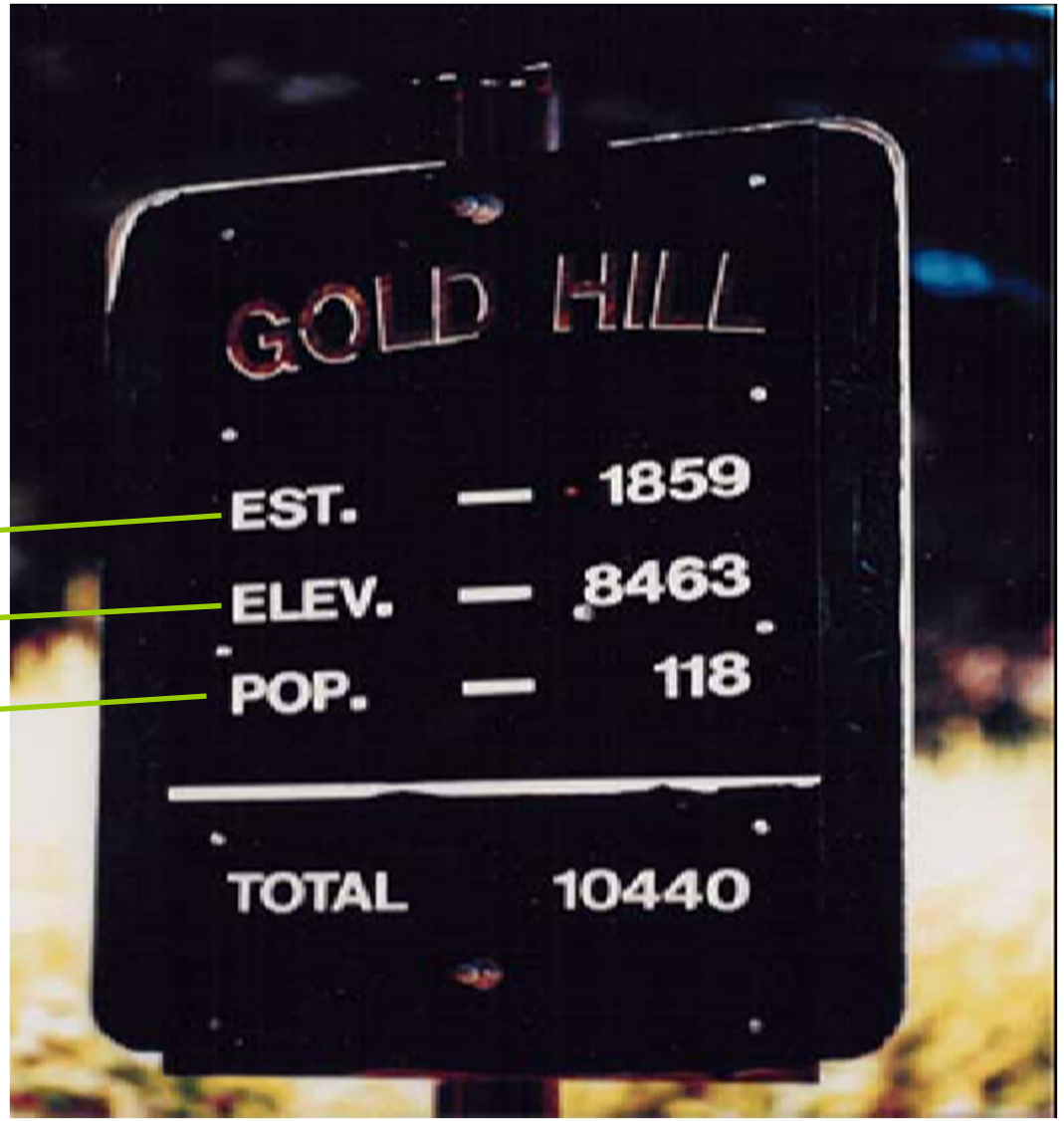
Namely

- Quality
  - And all quality attributes such as comprehensibility, usability, portability, maintainability etc. etc,
- Complexity

"There's more than one way to do it"

Freie Universität Berlin

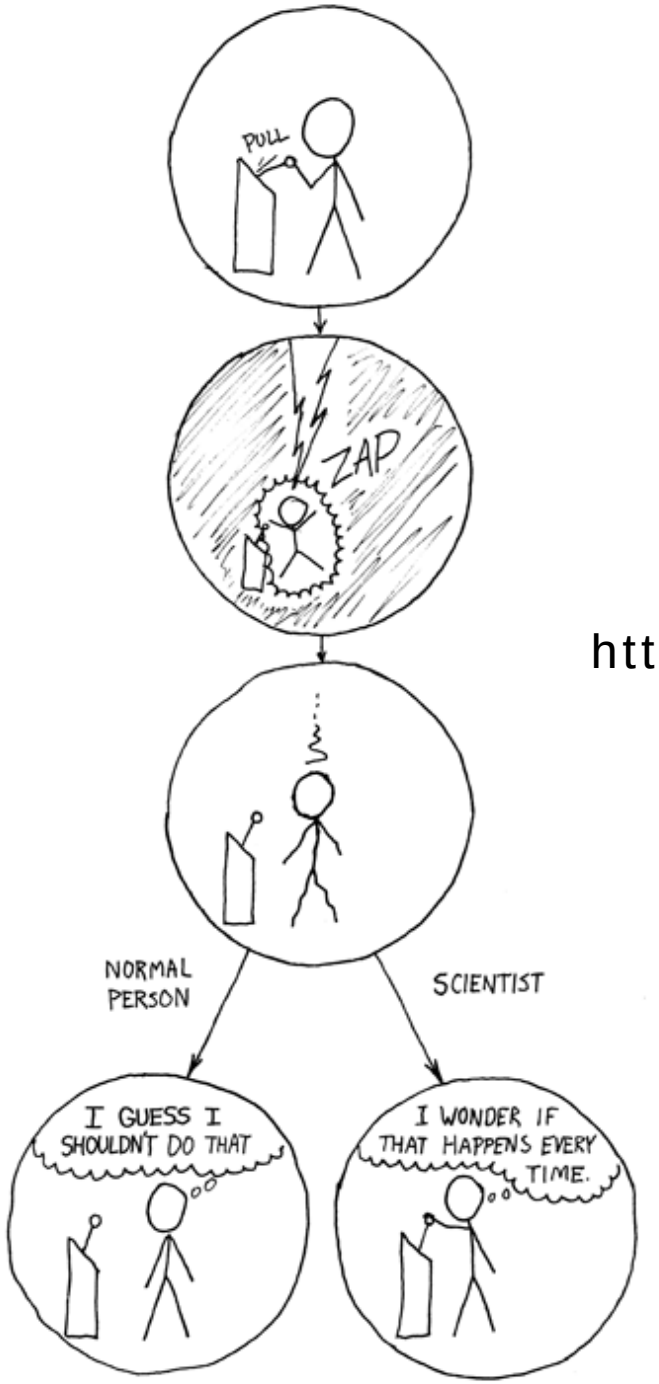
- Even if you do have a ratio scale, things may go wrong:



Summary

Freie Universität Berlin

- The three basic modes of informatics are **theory, construction, and empiricism**
  - all three are essential for successful work
  - empiricism is slowly gaining ground
- There are many examples where **sound decisions** can be made on empirical basis only
  - in particular when selecting technology or methods
- The main **quality criteria** for empirical work are
  - credibility and
  - relevance
- **Scales** are often mis-used
  - Make sure you have a sufficient scale type
    - For SW quality or complexity you almost never do



http://xkcd.com/c242.html

Thank you!