Human-Centered Evaluation of Software Artefacts in Computer Science: Introduction, State-of-the-Art, and Perspectives

Stefan Hanenberg
University of Duisburg-Essen, Germany

Santiago de Chile, Chile, 14.09.2012
What is this talk about?

- Tries to argue that human-centered / empirical studies are necessary
- Introduces into some basic terms
- Gives an overview of techniques required to perform experiments
- Shows pitfalls of experiments
- Gives an example of an experiment
Motivation

- Two different targets for research in CS
  - Machines
    - Execution speed, memory consumption, etc.
  - Human
    - Development speed, development errors, etc.
- Nowadays research methods mainly address machines
- Human plays rather minor role
- Usability (human interaction) rarely tested
Why should we care about humans?

- Humans are one of the main audience for CS constructs
- Usability of
  - Programming languages
  - APIs
  - User interfaces
  - ...
- Extensibility
- Maintainability
Current situation

- Example: Programming Language
  - Typical statement from the community:
    - „If a language is good, people will use it“
  - Questions:
    - „How many people must use a language so that it becomes good?“
    - „What about the moment when a language was initially developed?“
    - „What about marketing effects?“
    - „What should be the motivation of the first developer using a new PL?“
- Strange
  - Later on hardly tested whether PL was being used
  - „There is a community...so the language must be good“
- Example: well.....many, many

Typical situation: anecdotes instead of applied research method
Claim

- Artifact design is (often) about developers
- Current dominating approach
  1. Find example
  2. Build construct
  3. Claim that construct helps developers

This leads to nowhere

- Research methods needed that consider developers / users ... involved humans
Why not the traditional way?

• Machine / algorithm / etc.
  • Formal models, formal proofs, etc.

• Human
  • No formal model
    => no formal reasoning
    => traditional approaches cannot be applied
Overview of CS Research Methods

Taken from [Hanenberg, Onward'10]
Structure

- Need for experimentation (here: controlled experiments with humans)
  - What means experimentation?
  - What is required to run experiments?
- State-of-the-art
- Challenges in experimentation
- Example: Experiment on type systems
- Conclusion
Why experiments?

- Problem (again)
  - No formal model available how humans work
- Experiments
  - Observations as tests what really happens
  - Approximation (examples) of actual behavior
- What is a test?
  - There must be a statement which says when a test fails (hypothesis)
  - There must be an objective way to check, whether test has failed (falsification)
Logic of experimentation

• An experiment...
  • does not provide a proof for a theory
  • can NEVER consider all existing variables
  • can hardly reflect on real world situations
  • can only provide some evidence that a new construct helps (apart from developer's subjective impression)

• Why should it be useful?
  • Test: „Does the artifact really help in situations the inventor had in mind“?
  • Result: „Uselessness of artifact can be shown!“
Structure of Experiment

- Measurement of impact of
  - Independent variable (e.g. PL) on
  - Dependent variable (e.g. development time)
- A variable has a number of different treatments
  - Example: Comparison between Java, C++, and C
    => Indep. Variable PL with three treatments
- Experiment typically suffers from confounding factors (variable which are not controlled)
Background of Experiments (Karl Popper)

• Scientific argumentation
  – Falsification of hypothesis
    (use of statically typed language decreases development time)
  – More often
    • Exploratory analysis (let’s see what happens if…)

  – NO PROOFS / NO GENERALIZABILITY
    • But always the hope that repeated observations reveal some truth
Background of Experiments  (Karl Popper)

• Validity of hypotheses
  – Evidence for hypotheses increases the more often they could not be rejected

• Assumption
  – Massive execution of experiments

• Hope...(as practical researcher)
  • the more data available, the more probable it is, that we finally „see some rules“
Single vs. Multiple Runs

- General idea of experimentation
  - It shows, that hypothesis does not hold

- Single run experiments (in physics)
  - Example: Galilei's Pisa experiment
    => Single run falsified existing theory
    => Boolean statement from single run => **Boolean logic**

- With humans: **Multiple runs**
  - Humans differ too much
    => Multiple runs required
    => How often do runs need to falsify theory?
    => Argumentation based on analysis of sample => **Statistics**
Remaining questions

• How to design / perform experiments?

• How to analyse experiments?

...let's discuss it the other way around
Statistics in 5 minutes....

- Descriptive Statistics
  - Arithmetic mean, medians, variance, etc.
  - Relatively easy to understand, but inappropriate

- Inductive Statistics
  - Consideration of probabilities
  - Not that intuitive to understand, but state-of-the-art
Example: Descriptive Statistics

- Software development times with techniques A and B (in hours), 10 subjects
  - A: 1, 2, 3, 4, 1000 (mean: > 200, median: 3)
  - B: 10, 20, 30, 40, 50 (mean: 30, median 30)

- Problem
  - Argumentation based on mean or median?
  - Is 1000 an outlier that should not be considered?
  - Problems of descriptive statistics well known...
Inductive Statistics

• General idea: compare distribution / density functions of samples A and B

![Graph showing distributions A and B with an effect size arrow and test A < B]
Inductive Statistics\(^{(1)}\)

- General idea: compare density functions

![Diagram showing two normal distributions with annotations for effect size, deviation, and overlap.](image-url)
Inductive Statistics (1)

- General idea: compare density functions

- Computation of overlap between density function
Inductive Statistics (2)

- **P-value**: (Error-) Probability that a sample does NOT show A<B

- Arbitrarily(!) chosen alpha-level as „significance level“ (typically: 0.05, 0.01, …)

- Example:
  - „The difference turned out to be significant under an alpha-level of 0.05“
  => p<0.05
Inductive Statistics (3)

- Sample typically does not show perfect curve
  => approximation of density function required
  => sometimes, not even the kind of density function is known

- Standard mechanisms (significance tests) to compute p-values for different scales and sample sizes
  - T-Test, Wilcoxon-Test, Mann-Whitney-U-Test,

- Standard mechanisms to determine, whether a certain distribution can be assumed
  - Shapiro-Wilk-Test, K-S-Test, etc.

- All these tests are implemented in standard statistic software (R, SPSS, S, MiniStat, SAS, ...)
Inductive Statistics (4)

- Comparison of multiple curves (ANOVA): Impact of 1, 2, 3 on measurement

- Again: p-value (error probability that difference does not depend on 1-3)

- Partial-Eta-Square: How much of the variation can be explained by the variable (with the treatments 1-3)
Inductive Statistics (5)

- Quasi-endless different kinds of tests for different number of treatments and variables

**Take away:**
- Determination of error-probability $p$
  - Different standard significance tests
- Value of $p$ depends on
  - Effect size
  - Sample size
  - Scale
  - Applied significance test
  - Deviation (breadth of curve)
Remaining question

- How to design / perform experiments?
  - What kinds of experimental design are possible / desirable?
  - What is the impact of a certain design on the results?
  - What kinds of measurements can be applied?
  - ...

Experiment Design (1)

- Two-group between-subject design
  - One independent variable with two treatments
  - One subject tested under one treatment
  - Two different groups, each contains subjects with same treatment
- Example (Language A, B):
  - A: 1, 2, 3, 4, 1000
  - B: 10, 20, 30, 40, 50
- Problem
  - Both groups require subjects with "the same characteristics"
  - Problem: requires "very large" effect size in order to measure difference (for small sample sizes)
Experiment Design (2)

- Four-group between-subject design
  - Two independent variables with two treatments
  - One subject tested under one treatment
  - Four different groups, each subject assigned to treatment pair
- Example (Language A, B; Programming Task 1, 2)
  - G1 (Language A, Task 1): 1, 2, 3, 4, 1000
  - G2 (Language A, Task 2): ...
  - G3 (Language A, Task 3): ...
  - G4 (Language A, Task 3): ...
- Problem
  - Groups still require subjects with „the same characteristics“
  - Still: requires „very large“ effect size in order to measure difference (for small sample sizes)
Experiment Design (3)

- Large variety of further designs
  - Repeated measures designs, factorial designs, block designs, ...
  - Between vs. within-subject designs, ...

- General problems / considerations
  - Does design match hypotheses?
    - Difference hypotheses, correlation hypotheses, ...
  - Does design permit to determine effect?
    - Effect size, deviation, sample size, statistical power of required significance tests, ...
Experiment Design (4)

- General problem: **No measured effect**
  - Possible interpretations:
    - Sample size too small
    - Deviation too high
    - Inappropriate design
    - Non-exact measurement
    - ....
  - Alternative interpretation
    - Well, maybe the effect does not exist

- Pure technical problems
- Easy to run into these problems!!!
- NO (!) indicator that main effect does not exist
Experiment Design Example

Example

- 2 group experiment, 10 subjects, comparison of Java and Assembler
- Subjects: First year students
- Task:
  - Write an algorithm that computes a strongly connected component with $O(n^3)$
  - ...without using a book on algorithms
- Assumed result:
  - Average solution requires more than a year development time
  - No measured difference between Java and Assembler
    => very large deviation, small sample size, unbalanced groups,...

=> actual task has a huge impact on measurements
=> be careful when having an experiment without measured effect
(p > alpha-level)
Experiment Design: \( p > 0.05 \)

- But
  - if the significant effect of variable is „obvious“ (common community believe)
  - if the number of subjects is „high“ (whatever that means)
  - chosen tasks are the „killer-examples“ for the measured technique
  - …then...

\[ \Rightarrow \] Non-significant results are still interesting

(\textit{but only! then})
Experiment Design (6)

- Take away: Experiment design
  - ...must match research question
  - ...influences the final result (p-value)
  - ...requires appropriate analysis (t-Test, ANOVA, …)
  - ...results highly depend on actual task
  - ...be careful when no effect has been measured
Ok, let's do experiments

... but where and how to start?
Challenges of Empirical Studies

(remember: typically neither hypotheses nor concrete scenario available)
Challenges of Empirical Studies (1)

- Find / invent a hypothesis
- Find situations where hypotheses should be tested
- Find a good design

- Typical problem
  - „ Fighting the deviation / effect-size beast“
Challenges of Empirical Studies (1)

• Scientific approach
  – Observation of singular events (sample)
    (e.g. developers using a dynamically/statically typed programming language)
  • Formulation of hypothesis
  • Identification of dependent / independent variables
    (e.g. development time depending on type system)
  • Construction of environment
    (IDEs, tasks, languages, machines, …)
  – Collection of subjects
  – Measurements (e.g. development time to solve a certain task)
  – Analysis (mainly inductive statistics)
Challenges of Empirical Studies (2)

- Find / invent a hypothesis
- Find situations where hypotheses should be tested
- Find a good design

- Typical problem
  - “Fighting the deviation / effect-size beast”
Problems: Experiment Design

• Comparison between two samples

Example 1: Same effect size, different deviation
Problems: Experiment Design

• Comparison between two samples

Example 1: Same effect size, different deviation

Large overlap
=> no (significant) difference
Problems: Experiment Design

• Comparison between two samples

Example 1: Same effect size, different deviation

Large overlap => no (significant) difference

Small overlap => (significant) difference
Problems: Experiment Design

- Comparison between two samples

Example 2: Different effect size, same deviation
Problems: Experiment Design

• Comparison between two samples

Example 2: Different effect size, same deviation

Large overlap => no (significant) difference

Small overlap => (significant) difference
Problem(s) in Experimentation

Conclusion

Experimenter should try to
- reduce deviation, and/or
- increase effect size

● Possible ways

● Adaptation of experimental design
  (e.g. within-subject design) => Reduction of deviation

● Adaptation of tasks
  (no development „from scratch“) => Increase effect size
Example: Static Type System

[Kleinschmager, Hanenberg, Robbes, Tanter, Stefik; ICPC’12]

- Background: 4 experiments, „mixed results“
- Idea: Static type systems help when using an undocumented API
- Experiment
  - Java / Groovy as PLs
  - 9 programming tasks (designing tasks took about 2 month)
    - 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
Example: Static Type System

• Task 4,5: Semantic errors
• 1,2,3,6,8: New class usage
• 7, 10: Type errors
Example: Static Type System

- Potential problems
  - Artificially constructed API
    - parameter names do not reflect on type names (but on names chosen from the domain)
    - Is it representative?
  - Artificially constructed environment
  - Artificial programming tasks
  - Java type system
- Maybe we measured something else
  - "Existence of type annotations in the code help....no matter whether they are statically type checked or not"
- Maybe „in the wild“ positive effect of static type system „vanishes“
  - There is no generalizability
Example: Static Type System

• How to go on?
  • Traditional way
    – „We have done an experiment on type systems and found differences, let's go to the next topic“
  • Alternative way
    – Go on with experimentation on type systems
      • Variations on type systems, IDE support, replication of experiments, etc.
    – Try to find correlation hypothesis that survives falsification trials
Where to Start?

- Relatively few textbooks available specific to software engineering
Where to Start?

• Huge bunch of textbooks outside the domain of software engineering

  • Psychology
  • Social Sciences
  • Medicine
  • …

• Why not just use these books?
• Problem: Different domains have different problems…
Problem of different domains

- What is the difference between measuring blood pressure and software development time?
Problem of different domains

- **Blood pressure**
  - You will hardly find two (living) human subjects on this planet whose blood pressure differs by factor 10 (even factor 5 is unlikely)

- **Software development time**
  - It is hard to find a sample of human subjects where development time between best and worst developer is less than factor 5

=> Large set of experimental designs / statistical methods from for example medicine cannot be (directly) used in software construction
State of the Art: Empirical SE
State of the Art: Empirical SE (1)

- Empirical approach typically not taught to students
  - ...how can students check whether a statement „static type systems are good for developer hold“?
  - ...how can students understand an empirical study they are reading about?
  - ...how can a student perform such a study?

- There are empirical studies / controlled experiments (ok, not that many)
State of the Art: Empirical SE (2)

- Typically, a large number of experiments suffer from general problems (experiment design as well as analysis)

- A lot of techniques come up without a hypothesis / proposed measurement
  - Example: "Eclipse is quite a mature IDE and helps developer a lot"

  => Experimenter becomes "inventor of hypothesis to be tested"
State of the Art: Empirical SE

- Theories mainly describe existence of a difference
  - “…static type systems better than dynamic type systems“
  - …empirical knowledge rather low

- Theories typically do not try to quantify differences (for some good reasons)
  - …empirical knowledge rather low

- Experimenter currently have to „invent situations for language constructs on their own“
  - Example: Java vs. Assembler....
Empirical SE: Open issues

- Endless list of open issues
  - How can we distinguish good from bad developers upfront?
    - Fundamental question for certain experiment designs (factorial design, block design, etc.)

- What kind of programming tasks are worth being studied?
  - What tasks do have small deviations, which represent „daily programming tasks“?

- What tool support should be delivered in an experiment?
  - Most often, no data for tools is available...

- ...
Long term goal of SE

- Theories
  - Descriptions of situations where certain constructs dominate others (size of difference part of theory)
  - Large number of experiments that try to falsify theories
  - Example (very first initial step):
    - “When using an undocumented API, ..... ....static typing reduces development time“

- General kind of theory:
  - “When the code is of kind X, ..... ...the use of construct A leads to C ...which differs to construct B by factor...“
Discussion & Conclusion

• Controlled experiments as a research method
  • Statistics, experiment designs
• Many, many problems
  • Missing experimentation in the past, basics, organizational issues
  • ...


Conclusion

- **We must teach experimentation**
  - Help people/students to understand what's going on
  - Students need to know methods which permit to identify techniques which are „bad, time consuming, error prone“

- **We need to integrate experimentation in our courses**
  - The SE course should not say „Pair programming is good“, it should also introduce the experiments which revealed that effect

- **We must do experimentation**
  - We want to improve the life of developers & users
  - This does NOT mean that we should ignore the machines
Human-Centered Evaluation of Software Artefacts in Computer Science: Introduction, State-of-the-Art, and Perspectives

Stefan Hanenberg
University of Duisburg-Essen, Germany

Santiago de Chile, Chile, 14.09.2012