Empirical Methods in Programming Language Research and Software Engineering

Stefan Hanenberg
University of Duisburg-Essen, Germany

Universität Koblenz, Germany, 26.01.2012
Structure

• Reasoning in Software Science
  – Need for human-centered studies

• Empirical Software Engineering
  – Principles, Problems
  – Experimental Designs, etc.

• History/Future of Empirical Software Engineering
  – Why are Human-Centered Studies hardly practiced?
  – What needs to be done?

• Conclusion
Reasoning in Software Science
Software Crisis and Software Science

• Software crisis [Dijkstra, The humble programmer, 1972]
  - High costs of software construction
  - Large number of failed projects

• Approach in software science
  - Construction of new artefacts
    • Development processes, concepts, Programming Languages, Abstractions
  - Standardization of artifacts
  - New „paradigms“

GOAL: Improvement of Software Development
Research Methods in Software Science

- Large variety of different research methods

[Hanenberg, Onward 2010]
Research Methods (1)

• **Classical Approach** (e.g. deterministic algorithms)
  – Characteristics
    • formal artifact
    • formally described statement
    • Proof for statement

• **Stochastical Approach** (e.g. probabilistics algorithms)
  – Characteristics
    • formal artifact
    • formally described statement
    • Proof for statement
Research Methods (2)

• Benchmark-Based Approach (e.g. virtual machines)
  – Characteristics
    • Concrete artifact
    • Checkable statement
    • Experimental evaluation (benchmark)
      – empirical approach!

• ...

Previous approaches have a clear argumentation for / against the subject of research
Goal of Approaches

• Reduction of subjective impressions
  
  A statement is not true, just because someone is convinced that it is true

• Liberation from speculations
  
  A statement is not true because someone thought hard about it

• Liberation from authorities
  
  A statement is not true because it is done by an authority
Other approaches… - Example 1

• Example: [Dijkstra'76]*

„It is a characteristic of intelligent thinking to study in depth an aspect of one’s subject matter in isolation … for separation of concerns…“

– Phrase „separation of concerns“ dominating motivation for aspect-oriented community
– Increase of separation of concerns used as argumentation for certain technique (such as aspects)

Other approaches… - Example 1

• Characteristics
  – Human characteristics in focus of argumentation
  – No formalization
  – Argumentation based on
    • plausability / speculation
    • authority
  – No scientific reference backs up original statement

=> Invalid argumentation
Other approaches… - Example 2

- Example: [MaMøPed '88]*

  Object-oriented Programming „…is close to the natural perception of the real world: viewed as consisting of objects with properties and actions”.

  - Motivation / argumentation for object-oriented programming
  - Argumentation heavily influenced literature about object-oriented programming

Other approaches… - Example 2

- Characteristics
  - Human characteristics in focus of argumentation
  - No formalization
  - Argumentation based on
    - plausability / speculation
    - authority
  - No scientific reference backs up original statement

=> Invalid argumentation
First summary

Whenever* humans are part of the argumentation, the scientific argumentation is abandoned

* At least very often.
Further argumentations…

- A new type system TS+ appears …
- Goodness of TS+ argued by type soundness proof code example where type error occurred
- The new type system
  - requires additional annotations by the developer
  - is an alternative to code reviewing techniques, etc.
- Type soundness proof does not
  - permit to make statement about usability
  - permit to determine, whether it helps in software development
Further argumentations…

- Discussion
  - Prevented type errors might be irrelevant
  - Type annotations might increase developer’s effort
  - Code reviews might cost less than costs for maintaining type annotations
  - Code reviews are still necessary
  - …

Type proof does not help answering the most important question

Does it improve software development?
What’s wrong?

• New artefact requires additional interaction with developers
• Additional interaction might increase / decrease software development effort
• Necessity to study effort for additional interaction
• Classical approaches (mathematical reasoning) cannot make a statement about humans (effort)
Second summary

If a new artifact requires new interaction with developers, usability becomes an important (dominant?) question

Classical approaches do not help to study usability

Techniques necessary that take the developer into account
Empirical, Human-Centered Studies (Empirical Software Engineering)
Where to start?

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

- Psychology
- Social Sciences
- Medicine
- ...
Empirical Software Engineering

• Different facets
  – Controlled experiments, case/field studies, …

• Motivation
  – Humans part of the research method

• Main problem
  – Humans part of the research method
Controlled Experiments (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)
Controlled Experiments (2)

• 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
Problems in Experimentation

• Main Problem
  – Variability within/ among subjects
  – Huge bunch of possible (unknown) influencing factors
  – „No measured effect“ can always be the result of a rather inappropriate experiment setup

• Counteractions
  – Experimental Design
    • Within- / between subject design, Repeated measurement, Blockdesign, Latin Square, etc.
  – Task definition
Problem(s) in Experimentation

- Comparison between two samples

Example 1: Same effect size, different deviation
Problem(s) in Experimentation

- Comparison between two samples

Example 1: Same effect size, different deviation

Large overlap => no (significant) difference
Problem(s) in Experimentation

- Comparison between two samples

Example 1: Same effect size, different deviation

![Graph showing large overlap and small overlap](image)
Problem(s) in Experimentation

- Comparison between two samples

**Example 2: Different effect size, same deviation**
Problem(s) in Experimentation

• 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

...many more problems need to be addressed
**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**
Problem(s) in Experimentation

• Between-Subject Design: Each subject measured once
  – Problem
    • Deviation among subjects potentially hides effect
    • Requires balancing between groups (for small groups)
  – Benefit
    • No learning effect, Lower costs than within-subject-design

• Within-Subject Design: Each subject measured twice
  – Problem
    • Obvious learning effects
  – Benefit
    • Individual deviation not that important
Between-Subject vs. Within-Subject Design
Within-Subject Design: Example

• Question: Do type Casts Matter? [Stuchlik, Hanenberg DLS 2011]
  – 21 subjects (~ 5 h/subject)
  – Programming Languages: Groovy & Java
  – 5 simple programming tasks
  – Measurement: time until completion
  – Within-subject design
    • Low number of subjects
    • High variability between subjects
  – Expectation: devTime(Groovy) < devTime(Java)
  – Problem
    • Does learning effect matter?
Within-subject measurement

Groovy first

1st

Duration
Within-subject measurement

Groovy first

Learning

1st

Duration
Groovy First
Within-subject measurement

Duration

Language

Effect
Within-subject measurement

Groovy first

1st Language effect

2nd Duration

Learning
Within-subject measurement

Groovy first

measured difference

Duration
Within-subject measurement

Groovy first

Java first

Duration

1st
Within-subject measurement

Groovy first

Java first

Learning

1st
Within-subject measurement

Groovy first

Java first

Duration

Language effect

Learning

1st
Within-subject measurement

Groovy first

Java first

measured difference

1st

Language effect
Within-subject measurement

Groovy first

Java first

Small effect in Group „Groovy First“
Large effect in Group „Java first“
Within-subject measurement

• Problem
  – If learning effect larger than language effect
    => no measured difference

• Experiment succeeded
  – Large effort put into task definition and pilot-tests
Experiment Results

[Stuchlick, Hanenberg@DLS'11]

Results/Interpretation

Type casts are not that important
Between-Subject Design: Example

- Question: Do static type systems matter?  
  [Hanenberg, OOPSLA 2011]
  - 49 subjects (42 h/subject)
  - New Programming Languages
  - 2 tasks (scanner & parser)
  - Measurements:
    - time for minimal scanner
    - Test case coverage for parser
- Between-subject design
  - High number of subjects
  - …within-subject not practical (>80 hours/subject!)
Between-subject measurement

• Problem
  – If deviation is too large, nothing will be measured

• Experiment only partially succeeded
  – We „think“ that deviation was too large
Results

- Dynamically typed language faster for scanner
- No measured difference on parser
Results [Hanenberg@OOPSLA 2010]

- Potential problem
  - Second measurement revealed very large deviation
  - …within-subject would have been the better choice
Influence of Experiment Task
Simple vs. Complex Tasks

Simple Tasks
- Quick response times
- Low error rates
- Rather low deviation among subjects
- ...put maybe effect size (still) too small

Complex Tasks
- Long response times
- High error rates
- Very high deviation among subject
Previous experiments

- Large deviation potentially result of task
Simple vs. Complex Tasks

Goal

- Finding as simple as possible tasks that reveal effect
- Find more complex tasks that reveal whether effect potentially get lost
- Find even more complex tasks that might show whether effect becomes negative
Previous experiments

- Slightly more complex tasks hide effect
Conclusion
Controlled Experiments
What does experimentation require?

• Experimentation requires time
  – Finding the right experimental design
  – Finding the right tasks
  – Finding the right balance between effect size and deviation

• Experimentation requires some (basic) skills
  – Knowledge in experimental design
  – Knowledge in techniques (inductive statistics)
  – …knowledge in measurement techniques…

• Experimentation requires some luck
  – Large bunch of potentially influencing factors potentially hide existing effect
  – …the beauty of experiments is that they can go wrong…
But in the end…

It is not that hard and it is worth to be done (see motivation)
Why is Experimentation With Human Subjects Hardly Done in SE?
The student‘s perspective

• Experimental Design almost completely missing in SE education
  => No training in reasoning on SE artefacts

• Big focus of SE education
  – known and practiced techniques
  – new state-of-the-art techniques

• „Computer Science is not statistics“

=> What should be the motivation of students to do an empirical as e.g. bachelor thesis?
The young researcher’s perspective

• …still not trained in experimental design
  – Initial effort for getting into the topic
• No infrastructure for experimentation
  – Where to get the subjects from?
  – From which budget should the subjects be paid?
• Time constraints
  – Personal experience: max 2-3 experiments / year possible
• Paper output
  – Hard to get experiments published….
  – High competition on publication market

=> What should be the motivation of a young researcher to do a phd in that topic?
Why is Experimentation With Human Subjects Hardly Done in SE?
Consequence

Even in the future knowledge about usability of software engineering artifacts will be rare...
What needs to be done?

- **Student’s duty**
  - Ask for validations of techniques being taught
  - Try to be as critical as possible and as objective as possible

- **Teacher’s duty**
  - Implicit & explicit teaching of empirical methods / research methods
  - Adaptation of lectures
  - Adaptation of curricula
  - Motivate students to contribute to experiments / to perform experiments

- **Publisher’s duty**
  - Make explicit
    - Hypothesis, research method, discussion of appropriateness of research method to address hypothesis
What needs to be done?

• PC chairs
  – Distinguish reviewers according to their skills in
    • Subject of research
    • Research method

• Reviewers
  – Do not accept papers without an evaluation
  – Check whether research method and hypotheses match
  – Make explicit if you are not familiar with the subject of research or the research method
  – Refuse to review papers whose research method you are not familiar with
Conclusion
Conclusion

- Motivation for urgent need for empirical methods
- Rough impression of „doing empirical studies“
- Rough impression of influencing factors in empirical studies
- What needs to be done to get empirical knowledge in SE
Empirical Methods in Programming Language Research and Software Engineering

Stefan Hanenberg
University of Duisburg-Essen, Germany

Universität Koblenz, Germany, 26.01.2012