The Role of Empirical Studies in Software Engineering

Dietmar Pfahl

University of Tartu
Estonia

EU member since 2004
Schengen Treaty since 2007
Euro as currency since 2011
Estonia

- Capital: Tallinn
- Population: 1.3 million
  - 70% native Estonians
  - 30% Russian-speakers
- Area: 45 000 km²
- Safe, stable society
- Foreign languages widely spoken
  (English, Russian, German)
- 3 million tourists (per year)
Academic Excellence since 1632
Milestones

• 1632: founded by Swedish King Gustavus II Adolphus

• 1802: reopened by Russian Tsar Alexander I as Imperial University at Tartu

• 1919: reorganized as the University of Tartu of the Republic of Estonia

• 1944: renamed Tartu State University (under Soviet Union)

• 1989: renamed as the University of Tartu
  • 2002 Implementation of Bologna Process
Hall of Fame

Friedrich Struve (Astronomy)
Moritz Hermann von Jacobi (Physics)
Karl Ernst von Baer (Embryology)
Hall of Fame

Wilhelm Ostwald (Chemistry, Nobel Prize 1909)
Jury Lotman (Semiotics)
Paul Ariste (Finno-Ugristics)
University in a Nutshell

- 9 Faculties
- 4 Colleges
- Students: ~17 500
- Staff: ~3 800
- Academic staff: ~1800
- Professors: 200
- Budget: 145.9 MEUR
- Research funding: 51.4 MEUR
University of Tartu: 9 Faculties, 4 Colleges

**Humaniora**
- Faculty of Theology
- Faculty of Philosophy
- Viljandi Culture Academy

**Medicina**
- Faculty of Medicine
- Faculty of Exercise & Sport Sciences

**Socialia**
- Faculty of Social Sciences & Education
- Faculty of Law
- Faculty of Economics & Business Administration
- Euro College

**Realia et Naturalia**
- Faculty of Science & Technology
- Faculty of Mathematics & Computer Science
  - Institute of Computer Science (SE Group)

Four IT Masters Study Opportunities @ University of Tartu (i.e. Masters of SE)

Pärnu College
Narva College
The Role of Empirical Studies in Software Engineering

Dietmar Pfahl

University of Tartu
Structure of Talk

- Characterisation of Software Engineering
- The Wallace-Model of Research
- Examples of Empirical Studies
  - Experiment
  - Survey
  - Case Study
- Validity Issues
- Summary
Engineering vs. Science

Science

- Detect (and prove) new 'laws' governing reality
- Proof that a solution to a known problem exists
- Invent new devices (prototypes) that demonstrate the existence of a solution to a problem
- Find new problems

Engineering

- Find solutions to known problems under given constraints (time and effort budgets, quality needs = functional and non-functional requirements)
SW Eng. vs. 'Other' Engineering

SW Engineering:
- Many engineers (up to 500) collaborate/cooperate to solve a problem
- Cannot rely on laws of nature (only math/logic and sociology/psychology)
- Material cost is negligible compared to personnel cost of engineers / Production is not a cost factor

Non-SW Engineering:
- Small teams of engineers solve a problem / design a solution
- Can rely on laws of nature (i.e., physics and chemistry)
- Many workers (and much material) involved in the construction/production process of (physical) artifacts
From Observation to Theory and back – Wallace Model

Theory is based on observation and reasoning (inductive & deductive)

Research methods are at the heart of Theory-building

Wallace, Walter L. (1971)
The Logic of Science in Sociology.
New York: Aldine
Types of Empirical Research Methods

- Survey
- Experiment
- Case Study
- (Action Research)
- (Ethnography)
- ...
Survey – Definition

- **Definition:**
  - A survey is a data collection method (e.g., interview) or tool (e.g., questionnaire) used to gather information about individuals

- **Characteristics:**
  - 'In-breadth' research (wide scope)
  - Can collect both quantitative and qualitative data
  - When to use? --> Either at start of research to get an understanding of the current situation / or at the end of a research phase to see the impact/acceptance/etc. of a new method/technique/tool

- **Issues:**
  - 'Superficial' --> no explanation / no causality --> not suitable for hypothesis testing
  - 'Generalisability' of results depends on the choice of population and 'response rate', as well as validity of data collection instrument
Survey – Example

What?

Research Questions:
- How is Agile practiced at Microsoft?
  - i.e. What do engineers do?
- How do engineers feel about it?
  - i.e. Do they like it?

Who, Where, and When?
- Microsoft (worldwide, 2006)
- Anonymous survey sent to 2821 engineers
  - 10% random sampling of all developers, testers, program managers at Microsoft in October 2006
- 487 valid responses
  - 44% developers, 28% testers, 17% program managers

Why?

Many agile approaches exist – what's in it for Microsoft?

Survey – Example (cont'd)

**Quantitative Results (Highlights)**

- 33% of respondents (spread across divisions) report their team uses Agile methodologies.
- They mainly use Scrum (68%).
- Used for many legacy products.
- Agile usage does not appear to depend on team co-location.
- Test-driven development and pair programming are not very common.

**Qualitative Results (Highlights)**

- MS engineers who have used Agile like it for their local team, but not necessarily for their organization.
- They worry about scale, overhead, and management buy-in.

---

<table>
<thead>
<tr>
<th>Perceived benefits (687 comments, 44 themes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Improved Communication and Coordination</td>
</tr>
<tr>
<td>2. Quick Releases</td>
</tr>
<tr>
<td>3. Flexibility of Design – Quicker Response to Changes</td>
</tr>
<tr>
<td>4. More Reasonable Process</td>
</tr>
<tr>
<td>5. Increased Quality</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Perceived problems (565 comments, 58 themes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Does not scale to larger projects</td>
</tr>
<tr>
<td>2. Too many meetings</td>
</tr>
<tr>
<td>3. Management buy-in</td>
</tr>
<tr>
<td>4. Unfamiliar with Agile</td>
</tr>
<tr>
<td>5. Coordination with other teams</td>
</tr>
<tr>
<td>6. Losing sight of the big picture</td>
</tr>
</tbody>
</table>
Survey – Example (cont'd)

Agile practice penetration at Microsoft

- Team coding standards
- Continuous integration of code
- System metaphor
- Simple design
- Sustainable pace
- User stories
- Small releases
- Direct interaction with customer
- Design improvement
- Collective code ownership
- Acceptance testing
- Whole team daily stand-up meeting
- Test-driven development
- Pair programming

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

- Yes
- Sometimes
- Planning to
- No
- Never will
Experiment – Definition

- Definition:
  - A (controlled or laboratory) experiment investigates a testable hypothesis by manipulating one or more independent variables (factor) to measure their effect on one or more dependent variables. Each combination of values (levels) of the independent variables is a treatment. All other variables are kept constant.

In the simplest case, one compares the performance of an experimental group to a control group:

- Experimental group: performs a task exposed to Level_A of Factor_1.
- Control group: performs the same task exposed to Level_B of Factor_1.

Example:
- Independent Variable: UML diagram usage (yes or no)
- Dependent Variable: Design Quality
- Treatments: A = use UML / B = don’t use UML
Experiment – Definition (cont'd)

- **Characteristics:**
  - Due to control over independent variables (in vitro = 'in glass', i.e. laboratory), helps detect causal relationships (--> tests hypotheses --> theory building)
  - Focus on quantitative data --> allows for use of statistics for hypothesis testing
  - When to use? --> When in-depth understanding about causal relationships is requested

- **Issues:**
  - Expensive, 'narrow' (in-depth)
  - Randomisation important
  - 'Generalisability'? ('artificial' lab setting --> test subjects, tasks, artifacts)
Experiment – Example

What?
Research Question:
- What is better – Pair Programming or Solo Programming?

Why?
Many studies with contradicting results – mostly conducted with students (not with professional developers)

Who, Where, and When?
- Norway, 2007
- 295 junior, intermediate and senior professional Java consultants from 29 companies were paid to participate (one work day)
- 99 individuals; 98 pairs
- The pairs and individuals performed the same Java maintenance tasks on either:
  - a "simple" system (centralized control style), or
  - a "complex" system (delegated control style)
- They measured:
  - duration (elapsed time)
  - effort (cost)
  - quality (correctness) of their solutions

Experiment – Example (cont'd)

Total Effect of PP

Difference from individuals

Duration Effort Correctness

-8 % 84 % 7 %
Experiment – Example (cont'd)

Effect of PP for Juniors

Duration | Effort | Correctness
--- | --- | ---
5 % | 111 % | 73 %

Effect of PP for Seniors

Duration | Effort | Correctness
--- | --- | ---
-9 % | 83 % | -8 %

Total Effect of PP

Duration | Effort | Correctness
--- | --- | ---
-8 % | 84 % | 7 %

Experiment – Example (cont'd)
Experiment – Example (cont'd)

**Effect of PP for Juniors**

- Duration: 5%
- Effort: 111%
- Correctness: 73%

**Moderating Effect of System Complexity for Juniors**

- Duration: CC (easy) 4%, CC (complex) 6%
- Effort: CC (easy) 109%, CC (complex) 112%
- Correctness: CC (easy) 32%, CC (complex) 149%

**Effect of PP for Seniors**

- Duration: -9%
- Effort: 83%
- Correctness: -8%

**Moderating Effect of System Complexity for Seniors**

- Duration: CC (easy) -23%, CC (complex) 8%
- Effort: CC (easy) 55%, CC (complex) 115%
- Correctness: CC (easy) -13%, CC (complex) -2%
So, when should we use PP?

<table>
<thead>
<tr>
<th>Programmer Expertise</th>
<th>Task Complexity</th>
<th>Use PP?</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Junior</td>
<td>Easy</td>
<td>Yes</td>
<td>Provided that increased quality is the main goal</td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>Yes</td>
<td>Provided that increased quality is the main goal</td>
</tr>
<tr>
<td>Intermediate</td>
<td>Easy</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>Yes</td>
<td>Provided that increased quality is the main goal</td>
</tr>
<tr>
<td>Expert</td>
<td>Easy</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>No</td>
<td>Unless you are sure that the task is too complex to be solved satisfactorily even by solo seniors</td>
</tr>
</tbody>
</table>

The question of whether PP is better, or not, is meaningless!

One should ask: In which situation is PP better to achieve a defined goal?

**Importance of Context:**
Helps construct/refine theory about when and how to do 'Pair Programming'
Case Study – Definition

- **Definition:**
  An empirical enquiry that investigates a contemporary phenomenon within its real-life context (in-vivo=in the living), especially when the boundaries between phenomenon and context are not clearly evident.

  - Descriptive Case Study
    - Purely observational / Focus on “What happens?”
  - Explorative Case Study
    - Initial investigation of some phenomena to derive new hypotheses and build theories / Focus on “What and Why?”
  - Confirmatory Case Study
    - Start out with a given theory and try to refute it, ideally with a series of case studies covering various contexts

- **Characteristics:**
  - 'Rich' (in-vivo=in the living),
  - More focus on qualitative data --> allows for better understanding of conditions under which a technique/tool works
  - When to use? --> When 'rich' information is requested

- **Issues:**
  - Important: Proper case selection / clearly stated research question / clearly defined framework for interpreting the observations
  - 'Generalisability' (1 case --> only 1 context')
‘Case Study’ – Example

What?

Research Questions:
- How many individuals to use in QA tasks where the primary goal is defect detection, e.g. inspections or testing, when having a fixed effort budget?

Who, Where, and When?

University, Sweden, 2007
- 19 MSc students at BTH (Blekinge Institute of Technology)
- Task: Inspection of requirements documents
  - Materials from a student project at Lund University, Sweden
- Detected defects: 31
- Defect detection times are recorded

Why?

Part of a larger research program studying the relevance of theories (laws) for configuring QA tasks.

Here: Linus' Law – Given enough eyeballs, all bugs are shallow!

Case Study – Example (cont'd)

Research Question illustrated

10 persons x 1 hour

1 person x 10 hours
How to estimate defect detection probability?

Assume we have k defects with associated average defect detection probabilities for a pool of reviewers at time t:

<table>
<thead>
<tr>
<th>Defect</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_1</td>
<td>0.4</td>
</tr>
<tr>
<td>D_2</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>D_k</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Then, the expected number of defects found by 1 reviewer is:

\[ 0.4 + 0.5 + \ldots + 0.8 = E(1, t) \]

... found by 2 reviewers is:

\[ (1 - (1 - 0.4)^2) + (1 - (1 - 0.5)^2) + \ldots + (1 - (1 - 0.8)^2) = E(2, t) \]

... found by n reviewers is:

\[ (1 - (1 - 0.4)^n) + (1 - (1 - 0.5)^n) + \ldots + (1 - (1 - 0.6)^n) = E(n, t) \]
Case Study – Example (cont'd)

If \( n \times t = c \) with \( c = \text{const} \): How to choose \( n \) and \( t \) to maximise \( E(n, t) \) for a given \( c \)?

Assume we have \( k \) defects with associated average defect detection probabilities for a pool of reviewers at time \( t \):

<table>
<thead>
<tr>
<th>Defect</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D_1 )</td>
<td>0.4</td>
</tr>
<tr>
<td>( D_2 )</td>
<td>0.5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>( D_k )</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Then, the expected number of defects found by 1 reviewer is:

\[
0.4 + 0.5 + ... + 0.8 = E(1, t)
\]

... found by 2 reviewers is:

\[
(1 - (1 - 0.4)^2) + (1 - (1 - 0.5)^2) + ... + (1 - (1 - 0.8)^2) = E(2, t)
\]

... found by \( n \) reviewers is:

\[
(1 - (1 - 0.4)^n) + (1 - (1 - 0.5)^n) + ... + (1 - (1 - 0.6)^n) = E(n, t)
\]
Case Study – Example (cont'd)

Empirical BTH data

- Since we know for each student (reviewer) the times at which defects were detected, we know:
  - Average defect detection probabilities for all defects (at end of review session)
  - Defect detection probability growth over time
Case Study – Example (cont’d)

Empirical BTH data

- Since we know for each student (reviewer) the times at which defects were detected, we know:
  - Average defect detection probabilities for all defects (at end of review session)
  - Defect detection probability growth over time

- Example for two defects D5 and D36 is shown in figure on the right
  - Average defect detection prob. of D5 (at t=100): 63%
  - Average defect detection prob. of D36 (at t=100): 58%
  - Expected number of total defects detected (at t=100): 1.21
Case Study – Example (cont'd)

1 individual (n=1):

\[ 1.21 = \]

\[ (1-(1-0.58)^1) + (1-(1-0.63)^1) = \]

\[ 0.58 + 0.63 \]

1 individual, \( t=100 \) (1*100)

1.21 defects
Case Study – Example (cont'd)

1 individual (n=1):
1.21 =
\((1-(1-0.58)^1) + (1-(1-0.63)^1) = 0.58 + 0.63\)

2 individuals (n=2):
1.35 =
\((1-(1-0.58)^2) + (1-(1-0.31)^2) = 0.8236 + 0.5239\)

0% 10% 20% 30% 40% 50% 60% 70%
0 20 40 60 80 100
D5
D36

2 individuals, t=50 (2*50) 1 individual, t=100 (1*100)

1.21 defects 1.35 defects
Empirical BTH data

- Example for two defects D5 and D36

General answer to the research question --> Optimisation problem:

\[ E(n, t) = \sum_{d \in \{1, \ldots, D\}} (1 - (1 - p(d, t))^n) \rightarrow \text{max} \]

- \( n \in \{1, \ldots, N\} \), with \( N \) maximum number of individuals,
- \( t \in (0, T) \), with \( T \) maximum duration of QA task,
- \( d \in \{1, \ldots, D\} \), with \( D \) total number of defects,
- \( p(d, t) \in [0, 1] \), average probability of detecting defect \( d \) at time \( t \) by any individual,
- \( t \times n = c \) with \( c \) is a constant effort budget
Case Study – Example (cont'd)

Results for 2 cases (simulation):

- Case 1: only inspection time (125 min maximum for single reviewer)
- Case 2: inspection + preparation time (165 min maximum for single reviewer)
- \(E(n,t)\): Inspection effectiveness for \(n\) reviewers at time \(t\)

<table>
<thead>
<tr>
<th>Case 1: fixed budget of 125 min (inspection time only)</th>
<th>Case 2: fixed budget of 165 min (inspection+preparation time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>(t/n) [min]</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
</tr>
<tr>
<td>2</td>
<td>62</td>
</tr>
<tr>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>16</td>
</tr>
</tbody>
</table>

Case 1

Case 2
Summary of results:

- **Case 1:** for all budgets it’s best to assign one additional reviewer to the same artifact (compared to the starting configuration: n=1)

- **Case 2:** only for larger budgets it’s best to assign one additional reviewer to the same artifact (compared to the starting configuration: n=1)

- Generally, if effort constraints apply, the optimal number of reviewers is rather small
  
  --> Linus' Law is only correct in the absence of time restrictions

- **Important:** Theory was the driver of the analysis (Linus' Law)
  
  --> Could there be other theories 'imported', e.g. from psychology?
List of Theories

- Yerkes-Dodson Law (Pressure)
  - Little pressure improves performance, but a lot of pressure hinders performance (people do choke under pressure specially in cognitive tasks).

- Anchoring
  - A cognitive bias that describes the common human tendency to rely too heavily, or "anchor," on one trait or piece of information when making decisions
  - E.g., the first few defects tester finds are likely to effect the type of defects tester will search for.

- Inattentional (or Perceptual) Blindness
  - Failure by a person to notice some stimulus that is in plain sight. This stimulus is usually unexpected but fully visible.
  - E.g., if instruction has been given to search for certain kinds of defects then it is very likely that other defects will be missed.

- Habituation
  - Repeating the same tasks over and over leads to decline in attention – and thus performance

- 'Tiring'

- 'Overspending' (Parkinson's Law)

- ...
Validity Issues (Threats)

Types of validity issues:
- Internal
- External
- Construct
- Conclusion
- ...

Influencing factors:
- Study design
- Sampling
- Population
- Statistical analysis
- Expectations
  - Researcher
  - Participant
- ...

Influencing factors:
Summary

- SE is engineering with special characteristics
  - Non-physical artifacts
  - Large teams of designers/developers
- Empirical studies are crucial
  - Understanding/Analysis
  - Theory-building
- Much work still to do
ISERN is a community that believes software engineering research needs to be performed in an experimental context. ISERN annual meetings are open for ISERN members, candidates and invited observers only (see ISERN Manifesto).

ISERN meetings are not conference style with refereed papers and presentations. Instead, meetings build on previous meetings and each session is supposed to foster collaboration, encourage discussions and contribute in building the knowledge in experimental software engineering. The continuous knowledge building will be formalized in ISERN Experience Factory.
Thank you very much!
International Masters of Software Engineering

Programme Director: Prof. Marlon Dumas

Institute of Computer Science
University of Tartu
marlon.dumas@ut.ee
IT Masters Study Opportunities @ University of Tartu

English-taught Masters programs:

- Masters of Software Engineering  
  http://www.ut.ee/software
- Masters of Cyber-Security (with Tallinn Univ. of Technology)  
  http://www.ip.ttu.ee/index.php?main_id=244
- Masters of Security and Mobile Computing (with Aalto Univ.)  
  http://nordsecmob.tkk.fi/

English + Estonian-taught Masters in Informatics

- Specializations in data mining, distributed systems, crypto, programming languages and Estonian language technology
Masters of Software Engineering
Structure

- Core Module (4 courses, 24 ECTS)
- Specialty Module 1: Enterprise Software (4 courses, 24 ECTS)
- Specialty Module 2: Embedded Software (4 courses, 24 ECTS)
- Electives+free Module (18 ECTS)
- Professional practice or entrepreneurship project (3-4 months)

Masters Thesis
Core Module

- Building the software **right**
  - Advanced Programming
  - Software Quality and Standards

- Building the **right** software
  - Software Economics
  - Systems Modeling
Masters of Software Engineering

**Core (24 ECTS)**
- Software Economics
- Systems Modelling
- Advanced Programming Techniques
- Software Quality and Standards

**Embedded Systems (24 ECTS)**
- Foundations of embedded real-time systems
- Real-time operating systems and programming
- Real-time systems development with UML/SDL
- Formal methods in embedded real-time systems

**Enterprise Systems (24 ECTS)**
- Enterprise System Integration
- Business Process Management
- Data Mining
- Enterprise Software Seminar
Truly International Experience

- 32 international students, 28 Estonian
- International staff with experience in 20+ universities worldwide
- World-class research groups (incl. top-10 most-cited software engineering researcher according to Google Scholar)
- Internship opportunities in international companies (e.g. Skype, Ericsson, Playtech, Nortal)
Check it yourself…

Publicly available course material & recordings

http://software.cs.ut.ee/study-program/

**Systems Modeling**

**Lectures**

- **06.09** - Class Modelling (M. Dumas)
  - Slides (Introduction)
  - Slides (Class Modelling)
  - Screencast
- **13.09** - Interaction Modeling (M. Dumas)
  - Slides
  - Screencast
- **20.09** - State Modelling (L. García)
  - Slides
- **27.09** - State Modelling (L. García)
  - Slides
- **04.10** - Modelling in Practice (T. Saarsen)
  - No slides this week
- **11.10** - Story-driven Modelling (D. Danilov)
  - Slides
- **18.10** - Story-driven Modelling (D. Danilov)
  - Slides
- **25.10** - Story-driven Modelling (D. Danilov)
  - Slides
Anything else I need to know?

50+ study places all with tuition-waiver

About 12 scholarships funded by IT Academy

Additional scholarships: Skype, Foreign Affairs Ministry

Application deadline: April 2014

http://software.cs.ut.ee