Course "Empirical Evaluation in Informatics"

Quasi-Experiments

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

- Example 1: language comparison
- Method: Like controlled experiment
  - but with incomplete control
  - typically non-randomization
- Example 2: effects of the workplace
"Empirische Bewertung in der Informatik"

Quasi-Experimente

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

• Beispiel 1: Vergleich von Programmiersprachen
• Methodik: wie kontrolliertes Experiment
  • aber mit unvollständiger Kontrolle
  • meist fehlt Randomisierung

• Beispiel 2: Wirkung von Arbeitsplatzbedingungen
Example 1: Comparing 7 programming languages

- Lutz Prechelt:
  "An empirical comparison of seven programming languages",
  IEEE Computer, October 2000

- Question: How do many implementations of the same string processing program compare for C, C++, Java, Perl, Python, Rexx, and Tcl?

- Study format: Quasi-experiment
Approach

• Have several dozen different authors write an implementation for a given requirements specification
  • They use a programming language of their own choice
  • Independent variable: Programming language used

Dependent variables:
• Measure the time required by the programmers

• Measure various attributes of the resulting programs:
  • program length
  • run time
  • memory consumption
  • reliability
Task: Phonecode

- The program converts 'telephone numbers' into word sequences
- Words come from a 73,000 word dictionary
- Conversion is based on the following mapping:
  e jnq rwx dsy ft am civ bku lop ghz
  0 111 222 333 44 55 666 777 888 999
- When no completion of a partial word sequence exists, the program may insert one of the original digits between two words
- Input text files: dictionary, telephone numbers
- Output format:
  3586–75: Dali um
  3586–75: Sao 6 um
  3586–75: da Pik 5
Origin and number of programs per language

• C, C++, Java ("non-script group"): Created by subjects of a controlled experiment about the PSP method

• Perl, Python, Rexx, Tcl ("script group"): Created by volunteers found via a public call for participation in Usenet newsgroups
  - solutions submitted by Email

<table>
<thead>
<tr>
<th>Language</th>
<th>Number of programs</th>
<th>Compiler or execution platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tcl</td>
<td>10</td>
<td>Tcl 8.2.2</td>
</tr>
<tr>
<td>Rexx</td>
<td>4</td>
<td>Regina 0.08g</td>
</tr>
<tr>
<td>Python</td>
<td>13</td>
<td>Python 1.5.2</td>
</tr>
<tr>
<td>Perl</td>
<td>13</td>
<td>Perl 5.005_02</td>
</tr>
<tr>
<td>Java</td>
<td>24</td>
<td>Sun JDK 1.2.1/1.2.2</td>
</tr>
<tr>
<td>C++</td>
<td>11</td>
<td>GNU g++ 2.7.2</td>
</tr>
<tr>
<td>C</td>
<td>5</td>
<td>GNU gcc 2.7.2</td>
</tr>
</tbody>
</table>
Results: Program length

- tcl
- rexx
- python
- perl
- Java
- C++
- C

Program length [statement LOC]
Results:
Run time for loading/initilization

- tcl
- rexx
- python
- perl
- Java
- C++
- C

Run time for z0 [seconds]
Results:
Run time without loading/initialization
Results: Run time

- Not many differences are statistically significant,
  - because of the high variance within the groups
- If we aggregate as follows: 1. C/C++, 2. Java, 3. Script group, we can say the following with 80% confidence:
  - Initialization phase:
    - Java took at least 1.3 times as long as C/C++ (on avg.)
    - Scripts took at least 5.5 times as long as C/C++ (on avg.)
  - Search phase:
    - No significant differences in mean times
    - But variability in script group was smaller by a factor of 2.1 to Java and a factor of 3.4 to C/C++
  - Total run time:
    - C/C++ was at least a little faster than Java (p=0.07) and than scripts (p=0.15)
Results: Memory consumption

memory consumption for z1000 [MB]
Results: Memory consumption

- C/C++ was most memory-effective
- Java was least memory-effective
- Script programs (except Tcl) were not worse than the worse half of C/C++
- Python and Perl had less variability than C/C++
- With a confidence of 80%:
  - Java consumed at least 32 MB (297%) more than C/C++
  - and 20 MB (98%) more than the script programs
Results: Work time

- tcl
- rexx
- python
- perl
- Java
- C++
- C

total time for programming [hours]
Results: program design

A qualitative finding when looking at the data structures used by the programs:

- Most script programs used associative arrays
  - Map from a digit sequence to a word
  - Built into all script languages

- Essentially all non-script programs used either
  - one large array, indexed only by first digit
    - leads to very inefficient solution
  - or a 10-ary tree
    - very efficient, but also complicated
Validity problems

With respect to internal validity, there are two problems:

- **Programmer capabilities**
  - Are the programmers comparable (per language group)?
  - Or have the most capable ones prefered certain languages?
    - There is some indication that the Perl programmers may have been above average
    - As Java had been very young at the time (1996/97), the average Java language experience may be below average
    - The rest appears reasonably even

- **Work times of script group**
  - Maybe the script group has cheated about their reported work time?
  - Can we find the plausible cheating candidates from the data we have?
Work time validation

Source text productivity [LOC/hour]

tcl
rexx
python
perl
Java
C++
C
Summing up: Results

For the given problem(!):

- Script programs were only half as long as non-script programs
- For this reason, they also took only half as long to write
- They were much slower in the I/O-intensive init phase
  - but hardly slower in the actual search phase
- They consumed more memory than C/C++ programs
  - but not more than Java programs

- Note: Keep in mind that the Java data was produced using JDK 1.2 and Java-inexperienced programmers
Quasi-experiment general method

1. A quasi-experiment resembles a controlled experiment:
   1. One thing is varied
      - It is called the experiment variable or independent variable
      - (There could be more than one)
   2. The rest is kept constant
      - These things are called extraneous variables
      - If human beings are involved, repetition is used
   3. Some result variables are observed
      - They are called the dependent variables

2. But the control is incomplete
   - Some of the attributes may lack constancy
   - Typical control reductions are discussed on the next slide
Typical control reductions

- Lack of randomization in group assignment
  - Self assignment
    - Subjects chose a group based on personal preference [as above]
  - Historical assignment
    - Groups exist before the experiment is even planned
  - Assignment by an outsider
    - e.g. a project manager assigns people using project criteria

- Different handling
  - The groups may be instructed, supervised, equipped etc. in a different way [as script vs. non-script groups above]

- Less reliable measurement
  - e.g. data measured by participants rather than the experimenter [as work time for the script groups above]

- etc.
Example 2: Effects of the workplace


- Question: Do high-performer or low-performer programmers cluster in different organizations?

- Study type: Quasi-experiment
Approach

- 35 organizations participated with one or more pairs of programmers, 166 programmers overall
- Each programmer solved the same task
  - working in their usual programming language, working environment, and work hours
    - more than 8 different languages were used overall
- Each programmer kept track of the time until two milestones:
  - 1. First clean compile, 2. Work completed
    - The first 100 participants tested the program of their pair-mate, the others tested their own
- Time log includes periods and types of work and interruption
- Each answered questionnaire about workplace conditions
Task

- The task involved "syntactic and semantic edits on an input stream of calendar dates, followed by computation of day-intervals between specified dates as much as 8 centuries apart."

- Mean program length was 220 lines

- Mean time to milestone 1 was 280 minutes
  - ➔ 47 LOC/h on average
Results: Work time differences

- The slowest participant took 5.6 times as long as the fastest
- Average time was 2.1 times the fastest time
- The slower half took 1.9 times as long as the faster half
Similarity of pairs

- Work time of the members of a pair correlated strongly
  \[ r = 0.79 \]
  \[ r^2 = 0.62 \]

- 62% of the differences between people is explained by the pair (organization) they belong to

\[ \text{RED} \]

\[ \text{BLUE} \]
Similarity of pairs

- The fastest and second-fastest persons were in one pair
- The slowest and second-slowest were in one pair
- Of 13 that did not finish, 10 were paired with other non-finishers
Time versus quality

• Little coupling between time required and number of defects

• The fastest 25% had 30% less defects than the rest

• More than one third of programs had no defects even without testing
Comparison of workplace conditions

- 25% fastest versus 25% slowest participants:

<table>
<thead>
<tr>
<th>ENVIRONMENTAL FACTOR</th>
<th>TOP 25%</th>
<th>BOTTOM 25%</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dedicated floor space</td>
<td>78 sqft.</td>
<td>46 sqft.</td>
<td>63 sqft.</td>
</tr>
<tr>
<td>Acceptably quiet workspace</td>
<td>57% yes</td>
<td>29% yes</td>
<td>42% yes</td>
</tr>
<tr>
<td>Acceptably private workspace</td>
<td>62% yes</td>
<td>19% yes</td>
<td>39% yes</td>
</tr>
<tr>
<td>Can you silence your phone?</td>
<td>52% yes</td>
<td>10% yes</td>
<td>29% yes</td>
</tr>
<tr>
<td>Can you divert your calls?</td>
<td>76% yes</td>
<td>19% yes</td>
<td>57% yes</td>
</tr>
<tr>
<td>Do people often interrupt you needlessly?</td>
<td>38% yes</td>
<td>76% yes</td>
<td>62% yes</td>
</tr>
<tr>
<td>Does your workspace make you feel appreciated?</td>
<td>57% yes</td>
<td>29% yes</td>
<td>45% yes</td>
</tr>
</tbody>
</table>

Output variable: time

Input variables!
Validity problem

- Fact: One organization had 18 participants in similar conditions, plus further 6 working in a quiet "clean room"
  - These 6 outperformed the other 18 by 40%
  - Why?

- Cause and effect may be either way round:
  - Better workplace conditions result in faster performance
  - Better performers will be provided with a better workplace

But:

- Three organizations with nine or more pairs each all showed very little variation in workplace conditions
  - so at least there the conditions are a function of the organization, not the individual performance
Summing up: Quasi-experiments

- Quasi-experiments are like controlled experiments, but with reduced levels of control
  - typically via non-randomized group assignment

- Relaxing control allows for very interesting studies
  - that would not otherwise be possible

- Creative ways can often be found to strengthen credibility despite the reduced control
  - e.g. the worktime validation in the language comparison
  - or the use of pairs in the workplace study
Thank you!