NPRG075
Human-centric language design

Tomáš Petříček, 309 (3rd floor)
✉ petricek@d3s.mff.cuni.cz
📍 https://tomasp.net | @tomaspetricek

Lectures: Monday 12:20, S7
📍 https://d3s.mff.cuni.cz/teaching/nprg075
Research methods

Human-computer interaction
HCI perspective

Are programming languages user interfaces?

The means by which the user and a computer system interact (…)

Shifts focus on users and interaction
Desktop metaphor

Created in the 1970s at Xerox

Metaphor as a design principle

Move from solving problems to building new interfaces
Human factors

Equipment interaction incidents by trained users in World War II

Design equipment to minimize potential for problems

Lab testing and experimental psychology
Research methods
What to study and how

- What is the most effective way of doing X?
- What mistakes programmers make and why?
- Can we solve X and Y in a unified way?
- Do systems enable new user experiences?
Methodological bias

Hierarchy in science

- Theoreticians over experimentalists
- Everyone knows Einstein's equation
- Nobody Michelson–Morley experiment

Biases in computing

- Proofs are the most fundamental!
- Can we measure something objective?
- Running a rigorous user experiment?
- All other evaluation is "too soft"!

Representing and Intervening: Introductory topics in the philosophy of natural science
Ian Hacking
Controlled experiments
Evidence-based language design
Evidence-based language design

For each language feature, determine the best option experimentally.

How to make user studies as rigorous as possible?
Randomized controlled trials

Gold standard in medicine

- Compare treatments or with placebo
- Random allocation of participants
- Blinding and study pre-registration

Limitations of RCTs

- Very hard to do properly
- Answers only very limited questions
- Even this may not be rigorous enough!
Case study: Perl vs. Randomo

An Empirical Investigation into Programming Language Syntax (Steffik, Siebert, 2013)
Getting it right

Study setup

- Copy and modify code sample
- Never programmed before
- Age, gender, language balance

Statistical evaluation

- Verified manual rating of accuracy
- Mauchly's sphericity test
- Repeated-measures ANOVA test
Perl vs. Randomomo

While users of Quorum were able to program statistically significantly more accurately than users of Perl (p = .047), and users of Randomomo (p = .004), Perl users were not able to program significantly more accurately than Randomomo users (p = .458).
Experiments

Studying languages experimentally

≠ Typing discipline, syntax, errors, inheritance

🛡 Compare two structurally similar alternatives

👥 Study participants with similar backgrounds

🧰 Does not help with fundamentally new designs
Empirical studies
Software repository analysis
Software repository analysis

Study existing codebases

- Lots of projects on GitHub
- Commit history, bug reports, etc.

What can we study?

- What leads to fewer bugs?
- How OSS contributors behave
- How code gets duplicated and reused?
- Code quality and code structure
Does strong typing matter?

Large scale corpus study

"[It] appear[s] that "strong typing is modestly better than weak typing, and among functional languages, static typing is also somewhat better than dynamic typing.""
Does strong typing matter?

Attempt to reproduce the study mostly failed

"I believe [it does] in my heart of hearts, but it's kind of an impossible experiment to run."
Repository analysis
How to and limitations

- Lots of code on GitHub is useless
- Focus on somewhat sensible projects!
- Many hidden factors to account for
- Avoid comparing apples and oranges
- Studying semantics and runtime is hard
Usability evaluation
Considered harmful
Cultural adoption
(Greenberg et al. 2008)

"Usability evaluation is appropriate for settings with well-known tasks and outcomes. They fail to consider how novel systems will evolve and be adopted by a culture over time."
Tricky to evaluate

Early designs
- Purely explorative sketches
- Getting the right design vs. Getting the design right

Cultural adoption
- Hard to imagine future uses
- First radio and automobiles
- Memex, Sketchpad and oNLine System
Evaluating user interface research (Olsen, 2007)

Lively research field in the 1970s and 1980s

Ubiquitous computing challenges the classic desktop metaphor

Increasing number of non-expert programmers!
User interfaces
New system and languages

- Reduce time to create new solutions
- Least resistance to good solutions
- Lowering skills barrier of users
- Power in common unified infrastructure
Simplifying programming
Data exploration tools
Data transformations using various online data sources

Too hard for Excel, too complex in Python or R

Getting it right is very time-consuming!
Demo
Data exploration in The Gamma
Evaluating The Gamma

Can non-experts actually use it?

Is it better than spreadsheets?

What desirable design characteristics does it have?
Case study: The Gamma

Evaluating programming systems

- Programming tool for journalists
- Olsen's framework for UI systems
- tinyurl.com/nprg075-ui

Design questions

- What possible claims can we make?
- What evaluation errors to avoid?
Methods review
Evaluating programming systems
Evaluating HCI toolkits
(Ledo et al., 2018)

Research claims made in publications about UI toolkits, etc.

The same works for languages, libraries, tools, frameworks, ...
Evaluation types

What claims can we make?

- **Demonstrations** - show what is possible
- **Usage** - study actual system use
- **Performance** - evaluate how well it runs
- **Heuristics** - expert rules of thumb
Demonstrations

- **Showing a novel example**
  
  Can do something previously unthinkable

- **Replicating past examples**
  
  System makes previously very hard thing easy

- **Presenting case studies**
  
  Show usability of a system in a range of situations
Demo or Die!

MIT Media Lab paraphrasing of "publish or perish".

Aspen Movie Map
The 1978 precursor of Google Street View

Demo of a radically new technology
Varv programming system evaluation
(Borowski et al., 2022)

Makes all information visible and modifiable

Affects the whole developer workflow

Case studies to illustrate the effects

Figure 1: Varv Examples: (a) A todo list web application that is inherently extensible. Here, a basic todo list is extended with the ability to complete and delete todos by adding two new concept definitions and new modified template definitions. (b) A board game toolkit that defines abstractions for board game logic. The games "Checkers" and "Othello" were implemented with the toolkit and then merged into a new "Checkers-O-Thello" game with the addition of a short concept definition. As Varv applications are represented as data structures, higher-level tooling can be developed including a block-based editor (right), an inspector to go from an element in the view to the corresponding template or data (context menu to the left), and a data inspector for live editing application state (middle).
Varv evaluation

Demonstrate workflow

- Two concrete usage scenarios
- Step by step description of work
- Using personas for concreteness

Potential of the system

- Implications of the design
- Debugging, authoring, tools
- Notebooks, blocks, VS Code, etc.
Usage evaluation of The Gamma (Petricek, 2022)

Can non-programmers really use the system?

Get non-programmers, ask them to try and watch and note!

The Gamma: User experience study

worldbank byCountry, 'United Kingdom'
  'Economy & Growth', 'GDP per capita (current US$)'
worldbank byCountry, Germany
  'Economy & Growth', 'GDP per capita (current US$)'
worldbank byCountry, 'Czech Republic'
  'Economy & Growth', 'GDP per capita (current US$)'

The code is good and returns data we can visualize!
The Gamma evaluation

13 participants from business team of a research institute

Asked to complete 1 of 4 different tasks

Evaluated using activity logging, observation and interview

<table>
<thead>
<tr>
<th>Task</th>
<th>Kind</th>
<th>Done</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>expenditure</td>
<td>•</td>
<td>Obtained one of two data series</td>
</tr>
<tr>
<td>#2</td>
<td>expenditure</td>
<td>•</td>
<td>Explored further data series independently</td>
</tr>
<tr>
<td>#3</td>
<td>expenditure</td>
<td>•</td>
<td>Explored further data series independently</td>
</tr>
<tr>
<td>#4</td>
<td>expenditure</td>
<td>•</td>
<td>Completed following a hint to use another member</td>
</tr>
<tr>
<td>#5</td>
<td>expenditure</td>
<td>•</td>
<td>Explored further data series independently</td>
</tr>
<tr>
<td>#6</td>
<td>worldbank</td>
<td>•</td>
<td>Completed after a syntax hint about whitespace</td>
</tr>
<tr>
<td>#7</td>
<td>worldbank</td>
<td>•</td>
<td>Completed very quickly</td>
</tr>
<tr>
<td>#8</td>
<td>worldbank</td>
<td>•</td>
<td>Completed, but needed longer to find correct data</td>
</tr>
<tr>
<td>#9</td>
<td>lords</td>
<td>•</td>
<td>Struggled with composition of operations</td>
</tr>
<tr>
<td>#10</td>
<td>lords</td>
<td>•</td>
<td>Completed very quickly</td>
</tr>
<tr>
<td>#11</td>
<td>lords</td>
<td>•</td>
<td>With a hint to avoid operations taking arguments</td>
</tr>
<tr>
<td>#12</td>
<td>olympics</td>
<td>•</td>
<td>With a hint to avoid operations taking arguments</td>
</tr>
<tr>
<td>#13</td>
<td>olympics</td>
<td>•</td>
<td>With hints about 'then' and operations taking arguments</td>
</tr>
</tbody>
</table>

Table 1: Overview of work completed by individual participants in the study.
The marks denote: • = completed, • = required some guidance, • = partially completed
Usage evaluation

Possible setup
- Complete a given task
- Observe, log & record
- A/B comparison of variants
- In the lab or in the wild

Collecting feedback
- Complete a questionnaire
- Ask to comment (Think aloud)
- Semi-structured interview afterwards
Studying usage in the wild

Widely used to understand use of commercial systems

What language or editor features are used, performance, project profiles
Heuristics

Rules of thumb for evaluating designs written by experts

Evaluation without direct human involvement!

Example: Match between system and the real world

Olsen's criteria for user interface systems
Heuristic evaluation

Nielsen's usability heuristics

- Characteristics of a good interface
- General usability guidelines
- Consistency, visibility of state, ...

Cognitive dimensions of notation

- Heuristics for assessing notations
- Broad-brush understandable evaluation
- Viscosity, visibility, abstraction, ...
Technical performance

Baseline or improves over state of the art

Efficiency, lines of code

Not about usability, but an easy thing to show
Technical performance

Getting it right

Claims, comparison, benchmarks, metrics, setup, presentation

See SIGPLAN Empirical Evaluation Checklist
Conclusions

Usability and evaluation
Usability evaluation

Evaluating and comparing with existing systems

Evaluating usability can inspire new designs

The danger is designing with focus just on effective evaluability
Reacting

Reactive programming

- Introduction to RxJS concepts
- Available at:
  https://www.learnrxjs.io/learn-rxjs/concepts/rxjs-primer

Why read this

- Widely used practical library!
- But what exactly is going on?
- Does it always behave "intuitively"?
Conclusions

Human-centric language design

- Evaluation methods from the HCI field
- Controlled experiments, empirical studies
- Demos, usage, heuristics & performance

Tomáš Petříček, 309 (3rd floor)

✉ petricek@d3s.mff.cuni.cz
📍 https://tomasp.net | @tomaspetricek
📍 https://d3s.mff.cuni.cz/teaching/nprg075
References (1/2)

Methodology

- Arnold, K. (2005). Programmers are People, Too, ACM Queue

Heuristics

Examples

- Ray, B. et al. (2014) *A Large Scale Study of Programming Languages and Code Quality in Github*, FSE
- Borowski, M. et al. (2022). *Varv: Reprogrammable Interactive Software as a Declarative Data Structure*. CHI

Books