Empiric studies about programming language design

I'm deeply annoyed by the sorry state of decision making in programming. It's OK to rely on intuition in a creative field like software development, but given how critical software is for todays society it is horrible that we don't even know fundamentals like whether types improve quality enough to warrant their cost or whether parentheses are less readable than Pascal — beyond anecdotal evidence, the effect of funding, and feelings influenced by what you already know.

I can't just fix that on my own, but as a first step, I'm gathering some resources with overviews of empiric studies and takeaways I see.

A Summary of Summaries of Summaries.

The current summaries are pretty old (around 2012). They are, for example, missing Rust and Typescript. I did not yet check the papers about these and know no empirical studies about their impact on code. When I stumble upon more recent research, I add it to the appendix.

Contents

1 Stefik's Evidence page

→ Programming Languages and Learning

"A quick primer on human-factors evidence in programming language design"

- Summarizes how few randomized controlled experimental studies there were from 1976 to 2012. Around 1 per year.
- Static typing with good documentation reduces development time when using a new API.
- Naming of keywords matters.
- Learning block languages makes it easier to go to text-based languages later. Except for state initialization.
- Quality of compiler error messages matters.

2 Dan Luu static types literature review

→ Literature review on the benefits of static types

A review of 16 publications of empirical studies about static types.

> Of the controlled experiments, only three show an effect large enough to have any practical significance.
> ...
> Unfortunately, they all have issues that make it hard to draw a really strong conclusion.

3 Robert Smallshire: The Unreasonable Effectiveness of Dynamic Typing for Practical Programs

A Python advocate talking at a Java conference.

Video — Video with Slides on the side

- The state of empiric knowledge is weak (cites two studies).
- Only 2% of the errors in dynamic languages on Github were type errors.
- Dynamic typing takes less time to type.
- Types couple distinct parts of the program.

There's a text summary of the talk.

I would summarize it as "the unreasonable story of a trillion dollar industry built on sand".

4 Summary of Summaries of Summaries

Not applicable.

See the page by Stefik for the little robust empirical evidence that is known for now.

5 Appendix

5.1 Random papers

Some related papers I read out of interest, not in any particular systematic structure.

5.1.1 To Type or Not to Type? A Systematic Comparison of the Software Quality of JavaScript and TypeScript Applications on GitHub

Justus Bogner, Manuel Merkel, 2022: https://arxiv.org/abs/2203.11115

TS in popular Github repos has:

- 5x lower cognitive complexity (but that metric does not include the cognitive load of types, so it may just mean that typescript of the same calculated complexity value as Javascript is harder to understand)
- fewer code smells
- 2x higher bugfix commits ratio
- 3x as many bugs, takes the same time to fix an issue (do external factors dominate? E.g. when people have time to **start** working on them?)

  TS projects in our sample were **more bug prone** and required **more time to fix bugs** ... decreased usage of the any type was not correlated with fewer bugs.

5.1.2 On the Impact of Programming Languages on Code Quality (re-analyzing a previous study, documenting pitfalls and best practices for such research)

E. Berger, C. Hollenbeck, P. Maj, O. Vitek, J. Vitek, https://arxiv.org/abs/1901.10220

> We uncover a number of **flaws that undermine the conclusions** of the original study ...

Combining corrections for all of these aforementioned items, the reanalysis revealed that only 4 out of the original 11 languages correlated with abnormal defect rates, and even for those the **effect size is exceedingly small**.

Best practices:

- Automate, document, and share
- Apply domain knowledge (e.g. V8 is written in C++)
- Grep considered harmful (accuaracy of classification was 36%)
- Sanitize and validate ("Real-sworld data is messy")
- Be wary of p-values