Usability of Programming Languages

Special Interest Group (SIG) meeting at CHI'2016

Brad A. Myers
Andreas Stefik
Stefan Hanenberg
Antti-Juhani Kaijanaho
Margaret Burnett
Franklyn Turbak
Philip Wadler

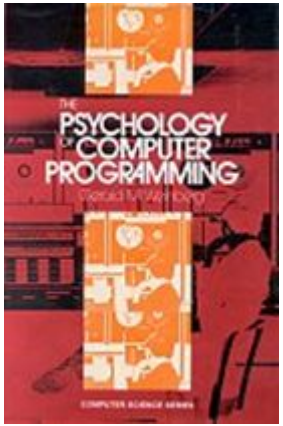
www.ProgrammingLanguageUsability.org

What is this SIG about?

- Programmers are people too
- More **usable** programming languages would be better
  - **Learnability**: People could learn programming easier
  - **Error proneness**: Programmers would make fewer errors
  - **Efficiency**: Programmers could create code faster
  - **Accessibility**: Various populations could be better included
- HCI methods can evaluate and improve programming languages

Studying Programmers has always been a CHI topic

- Original HCI! But many names
  - 1971 “Psychology of Computer Programming”
- “Software Psychology”
  - Ben Shneiderman book, 1980
- “Empirical Studies of Programming” (ESP)
  - Workshops from 1986 through 1999
- “Psychology of Programming Interest Group (PPIG)”
- The “International Conference on Program Comprehension“
- “Evaluation and Usability of Programming Languages and Tools“ (PLATEAU) at SPLASH/OOPSLA
- “Cooperative and Human Aspects of Software Engineering” (CHASE) at ICSE



Today's Focus: On the Programming Languages Themselves

- What is *known* about the usability of various design decisions?
  - Both low-level and high-level
  - Type Systems
  - Syntax
  - Other
- What *methods* are known to be successful for evaluating and improving the usability of a programming language?
- What, in particular, should future research focus on?

Why is this Needed?

- Few programming language designs are based on sound HCI principles
  - Java with JDK 8 and 9, C++ 11 or 14, ECMAScript 6, etc. have **not** been vetted from a human factors point of view
- Only **22 randomized controlled trials** of features of textual programming languages between the early 1950s through 2012 [12]
- The HCI and the Programming Language communities generally do not collaborate. Despite this:
  - Many people use programming languages (e.g., scientists, programmers, students)
  - K-12 education in the U.S. (and elsewhere) is increasingly including programming
  - Evidence in the literature suggests language designs very hard to use for many people (e.g., novices, pros under certain conditions, people with disabilities)

Only 22 RCTs? What's that about?

- The result of a systematic mapping study [12] conducted 2011-2014.
  - Research question: "What scientific evidence is there about the efficacy of particular decisions in programming language design?" (p. 109)
    - PLs restricted to textual general purpose languages.
  - Broad literature search and initial inclusion criteria (141 primary studies included).
    - Studies published after 2012 not considered.
- The 22 studies in question
  - Compare at least two language designs differing in the design of a feature or in the presence of a feature
  - Using some measure of usefulness to programmers (e.g. error proneness)
  - Assigning participants to (all sequences of) treatments using a random process
    - In case of a within-subjects design, full counterbalancing was required

On determining what is, and what is not, known

- *Much controversy and argument, even among the organizers of this SIG*
  - We have exchanges over 250 emails debating these topics!
- Current best practice in many fields is a **Systematic Literature Review**
  - Asks and answers specific questions relevant to practitioners
- Variant for scoping the literature is a **Systematic Mapping Study**
  - Asks and answers broad questions relevant to researchers
- Deliberately planned and meticulously conducted **with an audit trail**
  - Reports should describe search, inclusion/exclusion, quality appraisal, analysis, and synthesis processes **in detail**
  - Goal is a transparent secondary study whose reliability is evident to a reader
- Why go to all that trouble?
  - One may think one knows the literature, but often one is surprised!
  - It is **extremely nontrivial** to determine what two or more studies on the same question mean collectively (and nontrivial to interpret even a single study)

Evidence-Based Practice

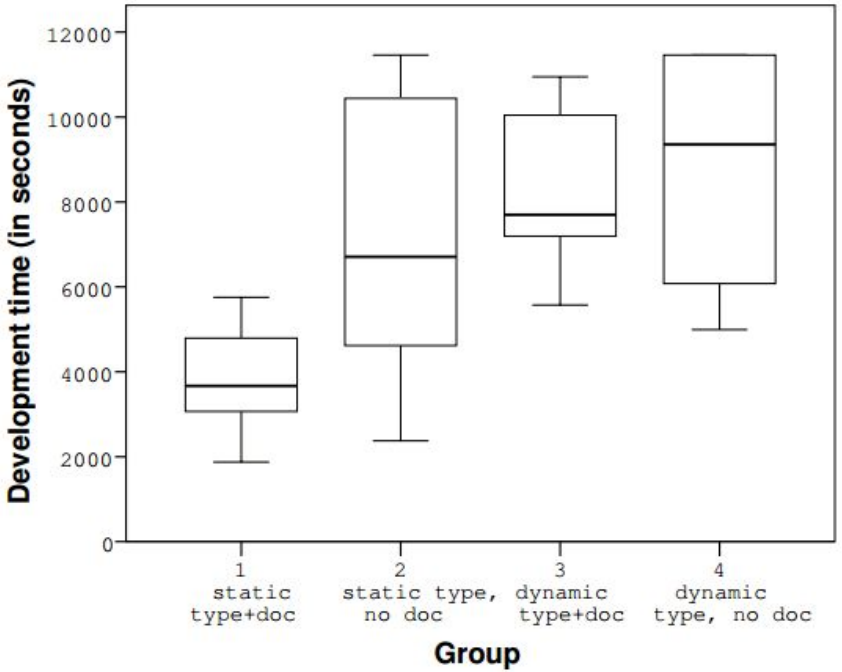
- … for when you have to make a decision
- Origins in medicine, adopted (with variants) in many other disciplines
- Five step process
  a. Formulate your problem as an answerable question
  b. Search the literature for studies that may bear on the question.
  c. Perform a quality appraisal of the studies found
  d. Apply the lessons of the studies to your decision problem
  e. Evaluate your own performance in this
- Systematic reviews are research, EBP is practice
- For detailed discussion, see [12]

What is the State of the Art?

Some basic progress

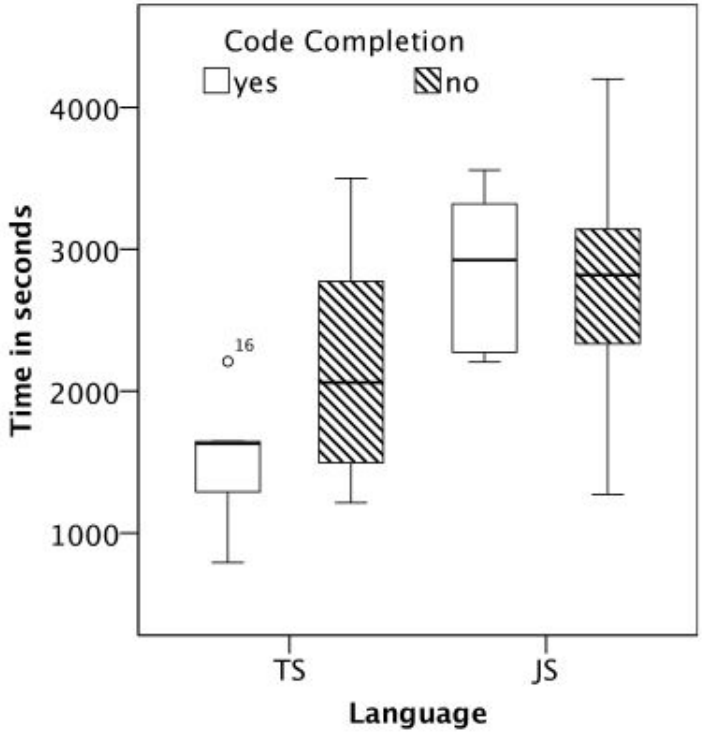
- Studies to document impacts of the syntax of a variety of programming languages:
  - *Andreas Stefik and Susanna Siebert. 2013. An Empirical Investigation into Programming Language Syntax. ACM Transactions on Computing Education 13, 4, Article 19 (November 2013), 40 pages.*
  - *Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15). ACM, New York, NY, USA, 522-527. DOI=http://dx.doi.org/10.1145/2676723.2677258*
  - *Jaime Spacco, Paul Denny, Brad Richards, David Babcock, David Hovemeyer, James Moscola, and Robert Duvall. 2015. Analyzing Student Work Patterns Using Programming Exercise Data. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15). ACM, New York, NY, USA, 18-23. DOI=http://dx.doi.org/10.1145/2676723.2677297*
- Studies on static typing vs. dynamic typing (about 12 randomized controlled trials so far)

Example Studies on Type Systems (Endrikat et al. ICSE'14)



- 2x2 controlled trial on static/dynamic typing with documentation/no documentation
- Static typing helps ($p<.05$, $\eta_p^2 = .30$)
- Minor effect of documentation ($.05<p<.1$, $\eta_p^2 = .14$)

Example Studies on Type Systems (Fischer, Hanenberg DLS'15)



- 2x2 controlled trial on static/dynamic typing with/without code completion

- Static typing helps ($p<.05$, $\eta_p^2 = .33$)
- Minor effect of code completion ($.05<p<.1$, $\eta_p^2 = .14$)

  (almost the same effect sizes as the previous study!)

But...

- .. some blog posts still seem to doubt about the meaningfulness of the results (see for example http://danluu.com/empirical-pl/)
- And there are no systematic reviews at all!

The Quorum Project

- Quorum is a programming language where the designers are trying to use the scientific method to make an easier to use alternative to the current generation of general purpose languages
- We call this an "evidence-oriented" programming language
- Quorum 4 (out this summer) is cross-compiled and supports the Java Virtual Machine/JavaScript/Apple backends, with a wide variety of standard libraries for:
  - Gaming (e.g., 2D, 3D)
  - Music generation
  - LEGO robots
  - Mobile support (iPhone)

Quorum started as a Language for Blind Children

- At its inception, it was quickly adopted across the U.S. at schools for the blind, however:
  - As we investigated the usability of programming languages and expanded the language, its popularity grew far beyond its original purpose
  - Quorum is now taught throughout the U.S. and has recently expanded to the UK, India, several African countries, Canada, and other locales
- Quorum changes on a regular cycle according to newly published evidence in the field, which we track. Examples include:
  - Changes to word choices in syntax/semantics as other scholars check alternatives
  - Changes to the type system as the evidence expands
  - Changes based on internal studies on a variety of topics (e.g., lambdas)

Token Accuracy Maps Help with Syntax/Semantics

Token Accuracy Maps are a statistical procedure, originally derived from DNA processing, that indicates "trouble spots" with syntax. Here is an example from a recent thesis on Token Accuracy Maps in Concurrent Programming:

```
class (90.91) Main (100.00)
  action (93.18) F (100.00)
    output (90.91) "hello" (75.00)
  end (90.91)

  action (95.45) G (100.00)
    output (90.91) "world" (79.55)
  end (93.18)

  action (90.91) Main (100.00)
    concurrent (90.91)
      F (97.73) ( (95.45) ) (97.73)
      G (100.00) ( (97.73) ) (95.45)
    end (90.91)
    output (88.64) "Done" (90.91)
  end (88.64)

end (90.91)
```

```
class (93.18) F (100.00) is (88.64) Thread (90.91)
  action (95.45) Run (97.73)
    output (95.45) "hello" (72.73)
  end (86.36)

end (93.18)
class (86.36) G (95.45) is (88.64) Thread (90.91)
  action (93.18) Run (93.18)
    output (95.45) "world" (72.73)
  end (86.36)

end (93.18)
class (86.36) Main (95.45)
  action (93.18) Main (93.18)
    F (93.18) t1 (90.91)
    G (90.91) t2 (90.91)
    t1 (90.91) : (88.64) Run (93.18) ( (93.18) ) (93.18)
    t2 (95.45) : (88.64) Run (95.45) ( (90.91) ) (90.91)
    check (77.27)
      t1 (81.82) : (79.55) Join (84.09) ( (84.09) ) (84.09)
      t2 (86.36) : (77.27) Join (84.09) ( (86.36) ) (86.36)
    detect (77.27) e (79.55)
    end (79.55)
    output (95.45) "Done" (100.00)
  end (95.45)

end (95.45)
```

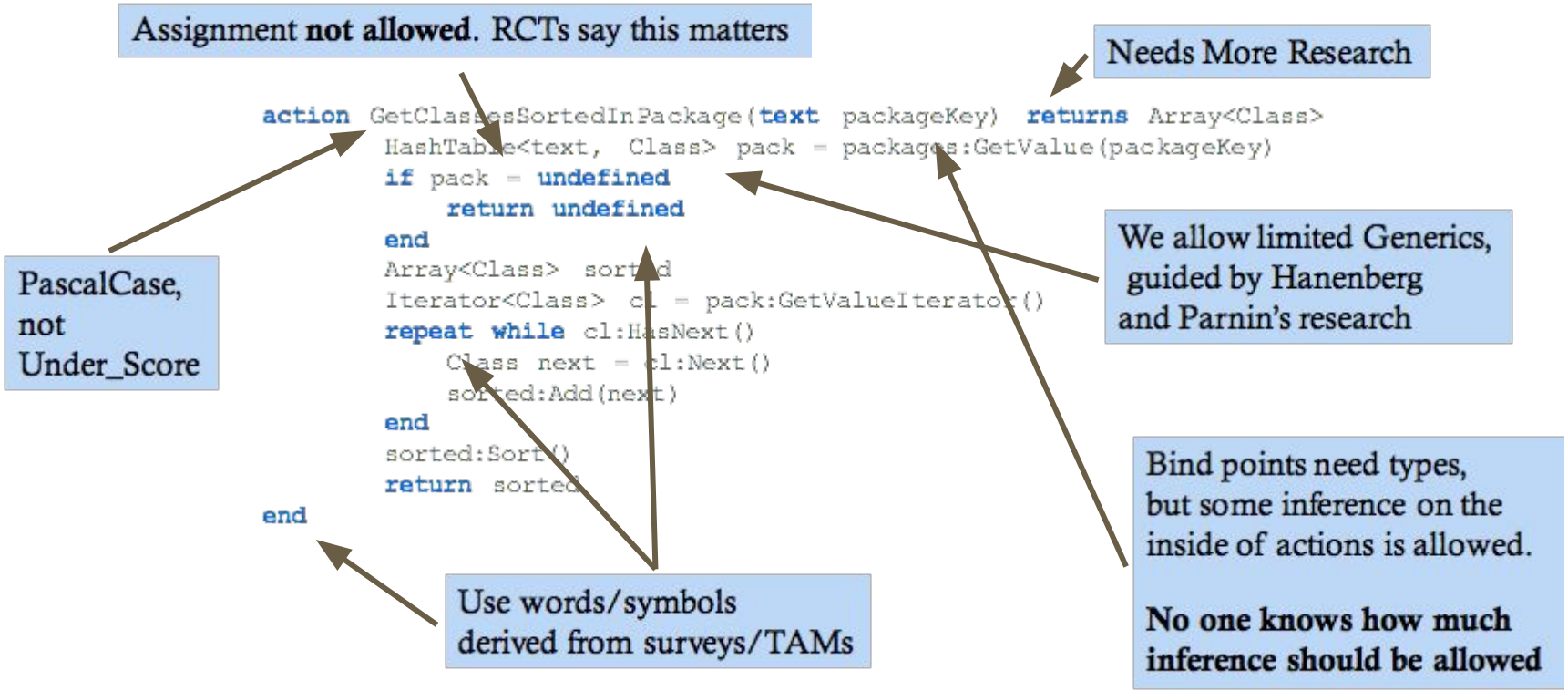
Data from these studies seems to match well with newer data from the literature, even for ***different methodologies***:

Amjad Altadmri and Neil C.C. Brown. 2015. 37 Million Compilations: Investigating Novice Programming Mistakes in Large-Scale Student Data. In Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15). ACM, New York, NY, USA, 522-527. DOI=http://dx.doi.org/10.1145/2676723.2677258

Paul Denny, Andrew Luxton-Reilly, and Ewan Tempero. 2012. All syntax errors are not equal. In Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education (ITiCSE '12). ACM, New York, NY, USA, 75-80. DOI=http://dx.doi.org/10.1145/2325296.2325318

David Weintrop and Uri Wilensky. 2015. Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs. In Proceedings of the eleventh annual International Conference on International Computing Education Research (ICER '15). ACM, New York, NY, USA, 101-110. DOI=http://dx.doi.org/10.1145/2787622.2787721

Let's look at an "Action" in Quorum



Why Now?

- Some researchers are trying to establish a body of knowledge to help future language designers:
  - Industry is using thousands of languages, none of which are carefully vetted for human factors. This may be expensive for organizations
    - C++ 14 is now ratified, with ***no human factors testing***, but it will be deployed world-wide, impacting millions of students and developers: https://en.wikipedia.org/wiki/C%2B%2B14
    - ECMA 6: The feature list of the latest version of JavaScript is extensive, yet the impact of any of them on people is unclear: http://es6-features.org/
  - As languages evolve, large institutions must adapt to them
  - Programming languages are increasingly being pushed in K-12 schools, yet are sometimes ***excruciatingly difficult to use***
- Existence proof that data can be gathered.
- A similar SIG on API Usability at CHI'2009 was very successful
  - See www.apiusability.org

Goals

- Development of community standards for evaluating programming languages
  - Alpha-levels, sample sizes, testing techniques, experiment layouts, etc.
  - Common experimental tasks and replication packets that can be used acr
    research groups, cultures, and language paradigms
  - **Example**: Briana B. Morrison, Lauren E. Margulieux, Barbara Ericson, and
    Guzdial. 2016. Subgoals Help Students Solve Parsons Problems. In Proceed
    the 47th ACM Technical Symposium on Computing Science Education (SIG
    ACM, New York, NY, USA, 42-47. DOI=http://dx.doi.org/10.1145/2839509.2844617
- Develop a body of exemplary studies, to serve two purposes:
  - Language designers can draw upon the results to inform their designs (res
    those studies)
  - Researchers can use them as a model for future studies (methods of those studies).

Goals - Part II

- Collect information about:
  - References for successful studies
  - What is known that can be used by future designs (results of those studies)
  - What is known about methods that work (methods used in those studies)
- Hope to create a **resource for the community**
- Reserved the URL: **www.programminglanguageusability.org**
- Help us get started by filling in this GoogleDoc: **http://tinyurl.com/ProgLangUsabilitySig**

Goals for the SIG

1. What do you know about? What else is known?
2. What methods have been used successfully (and unsuccessfully!)
3. What would you like to know about?
4. How carry this forward? What happens next?
    a. Mailing list? Web page?
    b. Dagstuhl? What else?
        i. Scope, length?

Agenda

- About 12 minutes each:

- Discussion of the Topics in the GoogleDoc:
  - **Articles**, etc. written by the organizers and audience and others
    - What do you know about?
  - **What is Known** about the Usability of Programming Languages
  - **Methods** that can help with Programming Language Usability
  - What **Needs** to be Studied
- Next steps?
  - Future conferences, meetings, workshops?

Articles, etc. written by the organizers and audience and others

- Please just add them to the GoogleDoc: http://tinyurl.com/ProgLangUsabilitySig
- What do you know about?

What is Known about the Usability of Programming Languages

- Event-based is a natural way for non-programmers to express behaviors [Pane]
- Logical operations like “And”, “or”, “not” are not natural [Pane]
- The “You” construct of CoScripter: used in a variety of ways to intertwine user action/computation with system computation [Bogart 2008]
- …

Methods that can help with Programming Language Usability

- Natural Programming Elicitation method
  - Understand the “natural” way developers think about a design issue
  - Natural Programming Plus method [Bogart 2012] adds scaffolding for language designers who are not HCI experts.
  - Contextual Inquiry Field Studies: Understand developers' real needs and barriers
- Randomized Controlled Trials
  - Methodological conventions that are considered the "gold standard" in most scientific fields.
  - Studies must follow conventions like: be replicable by other scientists, have control groups, assign participants randomly to groups, test hypotheses using data
- “Expert” evaluations
  - Heuristic Analysis, Cognitive Dimensions, Cognitive Walkthroughs
  - *(Controversial among the organizers!)*

What Needs to be Studied

- There are many open questions, given the lack of evidence in the field:
  - **Error Handlers**: research shows that error return values and try-catch blocks are both used poorly
  - **Syntax**: Only a handful of studies exist, yet there are millions of combinations that "could" be good designs. How do we narrow the field?
  - **Type systems**: Most studies have tested static vs. dynamic typing, but there are many other open questions (e.g., what should types be named, the design of generics/enums)
  - **Lambda Expressions**: Programming language designers have been adding them (e.g., C++ 11, JDK 8). What is their human factors impact?
  - **Modality**: Visual blocks are increasingly being used as an alternative to text in intro programming activities, but there are few studies comparing blocks vs. text.
  - **Different Kinds of People:** Many kinds of people use programming languages (e.g., scientists, game programmers, children in kindergarten, blind programmers). How does **each** of the design decisions impact **each** of these groups?

Next Steps?

- How carry this forward? What happens next?
- Mailing list? Web page?
- Goals for the future?
- A more formal workshop?
  - Dagstuhl conference?
  - Scope? Goals?
  - How long?
    - ½ day
    - 1 day
    - Multiple days?
  - When?
  - With CHI? ICSE? Something else?
  - Summary article?
  - For where?
- How to influence language designers?