Staking Claims

A (Preliminary) History of Programming Language Design Claims and Evidence

Shane Markstrum
Bucknell University
PLATEAU 2010

state of the language nation

- thousands of languages
  - just check out Wikipedia
- each language must differentiate itself
  - must be fixing some issue with other languages
- there are always issues with other languages
  - the next 700 thousand languages

design claims

- three categories
  - novel features
  - incremental improvement on existing features
  - desirable language properties

the science behind such claims

- claims need evidential support
  - novelty: it exists
  - incremental improvement: show how it improves
  - desirable properties: ?
- in reality, many claims are unevidenced
  - primarily the desirable properties
  - common wisdom, but when did it start
- quick note: not judging these claims myself

how properties are measured

- objective metrics for language properties
  - case studies
  - lines of code
  - corpus analysis
  - user studies

classic papers/languages

- what did early papers claim?
- look back at higher level languages
  - Fortran [Backus ’54] (speedcoding system)
  - ALGOL [Perlis & Samuelson ’58, Backus ’59]
  - Lisp [McCarthy ’60]
- why these papers?

Fortran

- claims focus on why a higher level language is better than assembly
  - desirable properties are claims of novelty
  - claims include:
    - extremely convenient
    - extensive
    - reduced coding and testing time
    - fairly fast
    - reduce[d] number of instructions

Fortran

- evidence
  - largely anecdotal
    - IBM group's experience
    - customer experience (unintended user study)
- problems
  - numbers do not receive statistical backing
    - "often enables one to reduce the number of instructions in a loop by a factor of ½"
    - "many problems which might require two weeks or more to program … can be programmed … in a few hours"

ALGOL 1

- more descriptive, fewer claims
  - novel syntax structure
  - desirable properties
    - natural
    - simple
    - readable*
    - mutual intelligibility*

* properties as goals of development

ALGOL 1

- evidence
  - none
  - consensus of a diverse group of designers
- problems
  - correlation between readability and naturalness?
  - is consensus enough?

ALGOL 2

- a single new claim
  - "convenient and concise"
- evidence
  - syntax presented in paper
- problems
  - what do these terms mean?

Lisp

- few and modest claims
  - novel form of functional programming
  - three desirable properties
    - readable
    - writeable
    - “advantage[ou]s”

Lisp

- evidence
  - novelty proved through peer evaluation
  - “case study” for proving ?
- problems
  - readable/writeable not really parallel
  - “case study” includes future work
- hedged claims
  - “We believe this formalism has advantages…”

modern papers/languages

- a taste of modern languages
  - C++ press release [Griggs ’84]
  - Java [Gosling and McGilton ’96]
  - Scala [Odersky et al. ’06]
  - Ur [Chlipala ’10]
- why these papers?
  - where’s my favorite language?

C++

- short, but full of claims
  - novelty
    - efficient compilation of class-based code
    - C-style memory manipulation mixed w/ OO
  - incremental
    - data abstraction as classes vs. C structs/union types

C++

- short, but full of claims
  - desirable properties
    - easy to understand
    - easy to modify
    - faster development
    - fewer errors

C++

- evidence
  - acknowledgement that others are using C++
  - none otherwise
- problems
  - not surprising for a press release
  - unclear what other use of C++ indicates about properties

Java

- significantly longer than the other papers
- lots of claims
  - novelty is played down
    - primitives for threading
    - easy module distribution via networking/web
  - incremental improvement is played up
    - every good thing about C/C++ and more

Java

- lots of claims
  - desirable properties (from the first two sections only)
    - simple
    - faster
    - portable
    - robust
    - adaptable
    - secure
    - small
    - familiar
    - clean
    - efficient

Java

- evidence
  - not peer reviewed (white paper)
  - sighting sources of inspiration indicates incremental
  - no direct, objective measures of other properties

Java

- what's good?
  - context for desirable properties is occasionally provided:
    - "simple (from removing features) [of C/C++]"
    - "familiar (because it looks like C and C++)"
  - use of objective studies to motivate features
    - readability gained by removing the surprises associated with macros/preprocessing
    - introduction of labeled break/continue statements to eliminate gotos
    - remove pointers because they are a vector for bugs

Scala

- claims
  - introduces a powerful new type system
  - desirable properties
    - scalable
    - “work[s] well with Java and C#”
    - uniform
    - flexible
    - natural
    - smooth, incremental development

Scala

- evidence
  - type system peer-reviewed in many papers
  - desirable properties
    - primarily anecdotal
    - many small examples
    - scalability will be proved when it is used more
- hedged claims
  - anecdotal evidence primarily used to justify removing features from the language

Ur

- claims
  - introduces dependent types to web programming
  - desirable properties
    - effective
    - practical
    - highly competitive
    - “[higher] productivity, security, and performance”

Ur

- evidence
  - author case study
    - including lines of code
- problems
  - lines of code presented in vacuum
  - author's case study as proof for general user
  - case study claimed as proof of higher productivity

discussion

- not many surprises
- novelty/incremental improvement is obvious
- desirable properties reflect authors' opinion

authorial superiority

- egoistic notion of desirable properties
  - who's language is not natural
  - designers' case studies reflect general populace
  - own limited viewpoint cannot be acknowledged

playing to the crowd

- no measurement = looking for sympathy
  - submit to venues with a bias
  - fewer challenges
- can this lead to general progress?

objective measurement

- lines of code
  - what does this actually mean
    - practicality, scalability, familiarity, readability, simplicity?
- user studies
  - difficult to organize
  - not always feasible
  - results may be inconclusive

objective measurement

- corpus analysis
  - can only be done if there is a corpus to analyze
- other measurement
  - inspiration from linguistics?
    - simplicity
    - readability
  - cognitive models

my thoughts

- if a claim is to be made, it must be given context and either
  - hedged, or
  - supported
- user studies are not always helpful, but they don't hurt
- we should look outside for new measurement techniques
- we as a community must expect a higher burden of proof

conclusions

- language designers are
  - good at inventing new languages
  - bad at objective assessment
- this has always been the case
- objective measurement is hard
  - but we should push for it