Evaluation

CS4700

Kenneth Sundberg

Subjective Evaluation

- Many languages exist
- There is no convergence in sight - there isn't even agreement on what is important.
- This is a multi-valued optimization problem

Tradeoffs

- Some of the evaluation criteria are in conflict
- Each programming language represents a solution to this conflict
- These solutions reveal what was important to the language designer – which may not be what is important to you

Readability

- How easily can an algorithm be read?
- Is is easy for a novice?
- Is is easy for an expert?

Writability

- How easily can an algorithm be written?
- What high level constructs are available?
- How much control does the programmer have?

Reliability

- How easily can bugs be written?
- What safety guarantees are available?
- What development tools are available?

Cost

- How will language choice affect development costs
- How difficult is it to learn the language?
- How familiar are the developers with the language already?
- How suited is the language to the problem domain?

Performance

- How quickly will the resulting program execute?
- Not important in all domains but dominates others.

Scientific

- Complex simulations with large data
- CPU bound
- There is always a bigger problem to solve

Business

- Large data
- IO bound
- Non-technical audience - sometimes even for the code

Artificial Intelligence

- Symbolic manipulation
- Complex algorithms (Big-O)

Web Software

- Portability
- Security concerns
- Small segments

Simplicity

- How many ways are there to do one thing?
- How many things can one symbol / construct mean?

Orthogonality

- How can language features be combined?
- Are they independent or correlated?

Data Types

- What data types are supported?

Syntax Design

- Are keywords chosen well?
- Does the form / shape of code correlate well with meaning?

Abstraction Support

- Can new types be defined?
- Are they second class citizens?
- How generic can functions be?

Expressivity

- How easily can you express an idea?
- How directly can you express an idea?
- Is the syntax cumbersome or clunky?

Type Checking

- How much compile time support is available for avoiding bugs?

Exception Handling

- How does the language handle errors and exceptional cases?

Restricted Aliasing

- What access do you have to raw memory?
- What support do you have to avoid memory errors?

Cost

- How difficult is it to produce code?
- How difficult is it to maintain code?
- Is optimization available?

Portability

- How easily does code written for one system transfer to another?

von Neumann Architecture

- Most languages are designed for the von Neumann architecture
- This has profound implications

Compiling and Interpreting

- The choice of when and how to convert a language to machine code impacts the features possible in the language.
- Compiled
  - Conversion done once up front
  - Allows for more optimization
- Interpreted
  - Conversion done at run time.
  - Cost of interpretation hides cost of some features making them more acceptable.
  - Portability is much easier
- Hybrid
  - Conversion to abstract machine (bytecode) done upfront
  - Allows for many optimizations
  - Portability is still easier than a compiled language

Textbook sections covered:

- Section 01-02 (frame 9)
- Section 01-03 (frame 4)
- Section 01-04 (frame 24)
- Section 01-06 (frame 3)
- Section 01-07 (frame 25)