Language Design: Back to the Future?

Laurence Tratt
http://tratt.net/laurie/

Bournemouth University

2008/07/08

Overview

1. Where are we at today?
2. Why are we where we are?
3. A glance backwards and sideways.
4. A gaze forward.

Part I: Where are we at today?

Where are we at today?

We've come a long way

- Always remember: software today is pretty good.
- Many programming languages to choose from.

Facing reality

Lisp sucks

Facing reality

Smalltalk sucks

Facing reality

Python sucks

Facing reality

Ruby sucks

Converge sucks

Facing reality

It sucks too!

The situation

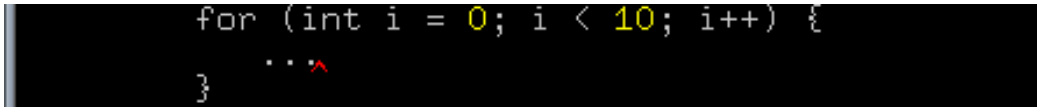
- Every programming language has flaws.

The situation

- Every programming language has flaws.
- Programming languages vary little.

The situation

- Every programming language has flaws.
- Programming languages vary little.
- In C:



The situation

- Every programming language has flaws.
- Programming languages vary little.
- In C:

```
for (int i = 0; i < 10; i++) {
    ...
}
```

- In Java:

```
for (int i = 0; i < 10; i++) {
    ...
}
```

The situation

- Every programming language has flaws.
- Programming languages vary little.
- In C:

```
for (int i = 0; i < 10; i++) {
    ..^
}
```

- In Java:

```
for (int i = 0; i < 10; i++) {
    ..^
}
```

- In D:

```
for (int i = 0; i < 10; i++) {
    ..^
}
```

The situation

- Every programming language has flaws.
- Programming languages vary little.
- In C:

```
for (int i = 0; i < 10; i++) {
    ..^
}
```

- In Java:

```
for (int i = 0; i < 10; i++) {
    ..^
}
```

- In D:

```
for (int i = 0; i < 10; i++) {
    ..^
}
```

- In Cyclone:

```
for (int i = 0; i < 10; i++) {
    ..^
}
```

- Is this a problem?

The situation

- Every programming language has flaws.
- Programming languages vary little.
- In C:

```
for (int i = 0; i < 10; i++) {
    ...
}
```

- In Java:

```
for (int i = 0; i < 10; i++) {
    ...
}
```

- In D:

```
for (int i = 0; i < 10; i++) {
    ...
}
```

- In Cyclone:

```
for (int i = 0; i < 10; i++) {
    ...
}
```

- Is this a problem?
- If language A isn't good for your problem, language B probably isn't either...

Part II: Why are we where we are?

History is written by the victors.

- Winston Churchill (1874 - 1965)

The gene pool

Source: Wikipedia

Homogeneity

- Most languages draw influences from the same small pool.
- A cliché (but true): syntax is often the main differentiator.
- Differences are perceived as much larger than they really are.

Homogeneity

- Most languages draw influences from the same small pool.
- A cliché (but true): syntax is often the main differentiator.
- Differences are perceived as much larger than they really are.
- Why do languages vary so little?

Language communities

- Prefer languages ‘to look familiar’.

Language communities

- Prefer languages ‘to look familiar’.
- Demand backwards compatibility.

Language communities

- Prefer languages ‘to look familiar’.
- Demand backwards compatibility.
- Language communities are insular.

Language communities

- Prefer languages ‘to look familiar’.
- Demand backwards compatibility.
- Language communities are *tribal?*

Language communities

- Prefer languages ‘to look familiar’.
- Demand backwards compatibility.
- Language communities are *tribal?*
- Informed comparisons are rare.

Language communities

- Prefer languages ‘to look familiar’.
- Demand backwards compatibility.
- Language communities are *tribal?*
- Informed comparisons are rare.
- Language communities beget language designers.

Language designers

- The obvious culprit?
- Problem #1: *really* learning a language is hard.
- Tend to have one dominant influence.

Language designers

- The obvious culprit?
- Problem #1: *really* learning a language is hard.
- Tend to have one dominant influence. Sometimes *only* one influence.

Language designers

- The obvious culprit?
- Problem #1: *really* learning a language is hard.
- Tend to have one dominant influence. Sometimes *only* one influence.
- Problem #2: designer vs. implementer.
- Implementation considered hard and expensive but vital for feedback.
- Problem #3: fear of failure.

Examples of a narrow perspective

- Scoping.
- Statements vs. expressions.

Examples of a narrow perspective

- Scoping.
- Statements vs. expressions.
- Python: confusion of class meta-levels.
- Ruby: blocks aren't first-class.
- Converge: brain-dead class hierarchy.

The risk of innovation

- New features are risky. Will they work?
- Most languages either:
  1. Have no new features.
  2. Have one or two new features.

The risk of innovation

- New features are risky. Will they work?
- Most languages either:
  1. Have no new features.
  2. Have one or two new features.
  3. Didn't mean to have new features but bad design introduced them.
- Little risk of 'failure' if there are no new features.

An example

- Java checked exceptions.
- Possibly Java 1.0's only novel feature.
- `public void f() throws X;` means callers of `f` *have* to catch `X`.
- Common user solution?

An example

- Java checked exceptions.
- Possibly Java 1.0's only novel feature.
- `public void f() throws X;` means callers of `f` *have* to catch `X`.
- Common user solution?

```
try {
  f();
}
catch (X) {
  // Empty catch statement. Ouch.
}
```

- Checked exceptions: a bad idea.
- The fate of most novel language features:

An example

- Java checked exceptions.
- Possibly Java 1.0's only novel feature.
- `public void f() throws X;` means callers of `f` *have* to catch `X`.
- Common user solution?

```
try {
  f();
}
catch (X) {
  // Empty catch statement. Ouch.
}
```

- Checked exceptions: a bad idea.
- The fate of most novel language features: ridicule.

Language paper writers

- People who write papers: designers, extenders, pedants.
- Nearly always framed in terms of one language...
- ...its syntax, semantics,

Language paper writers

- People who write papers: designers, extenders, pedants.
- Nearly always framed in terms of one language...
- ...its syntax, semantics, *and culture*.
- Extracting widely applicable ideas is extremely difficult.

Thesis

- Language communities are tribal and ignorant.

Thesis

- Language communities are tribal and ignorant.
- Language designers are timid and ignorant.

Thesis

- Language communities are tribal and ignorant.
- Language designers are timid and ignorant.
- Paper writers are obfuscators.

Thesis

- Language communities are tribal and ignorant.
- Language designers are timid and ignorant.
- Paper writers are obfuscators. And ignorant.

Part III: A glance backward ands sideways.

Icon

- The (indirect) successor to SNOBOL4.
- Dynamically typed PASCAL-ish language. But with unique expression evaluation system.
- Particularly intended for string processing.
- Expressions *succeed* (and produce a value) or *fail* and don't.
- ```
  if x := f():
    g(x)
  else:
    // x has no value
  ```
  ```

Icon

- Generators:

```
procedure upto(x)
  i := 0
  while i < x do {
    suspend i
    i := i + 1
  }
end

procedure main()
  every x := upto(10) do write(x)
end
```

- Conjunction:

```
every x := upto(10) & x % 2 == 0 do write(x)
```

Icon

- Print all words (from the Icon book):

```
text ? {
  while tab(upto(&letters)) do
    write(tab(many(&letters)))
  }
```

Icon

- Pretty cool stuff (ignoring minor, rectifiable, design flaws).
- Integrated pretty much wholesale into Converge.

Icon

- Pretty cool stuff (ignoring minor, rectifiable, design flaws).
- Integrated pretty much wholesale into Converge.
- Problem #1: `text.split(" ")`.
- Problem #2: regular expressions.

Icon

- Pretty cool stuff (ignoring minor, rectifiable, design flaws).
- Integrated pretty much wholesale into Converge.
- Problem #1: `text.split(" ")`.
- Problem #2: regular expressions.
- Conclusion: much innovation, but only generators and failure in `if` useful.

Compile-time meta-programming

- A.K.A. macros.
- They came from Lisp.

Compile-time meta-programming

- A.K.A. macros.
- They came from Lisp.
- ...and they ended with Lisp.
- Why?

Compile-time meta-programming

- A.K.A. macros.
- They came from Lisp.
- ...and they ended with Lisp.
- Why?
- Until: MetaML (and Template Haskell).
- Simple inversion of Lisp: ‘macros’ are normal functions but ‘macro calls’ are special.
- `$<f>` is a macro call.
- Code isn’t lists; `[| 2 + 3 |]` evaluates to an AST `plus(int(2), int(3))`.

An example

```
func expand_power(n, x):
  if n == 0:
    return [| 1 |]
  else:
    return [| $c{x} * $c{expand_power(n - 1, x)} |]

func mk_power(n):
  return [|
    func (x):
      return $c{expand_power(n, [| x |])}
  |]

power3 := $<mk_power(3)>
```

means that `power3` looks like:

```
power3 := func (x):
  return x * x * x * 1
```

by the time it is compiled to bytecode.

The macros dark ages

- Oh the irony.

The macros dark ages

- Oh the irony.
- An example of insularity?
- Sometimes other communities see things our own can't.

XOM

- Nowadays a language needs good libraries.
- Same principles.
- Converge needed an XML library. XML is easy, right?

XOM

- Nowadays a language needs good libraries.
- Same principles.
- Converge needed an XML library. XML is easy, right? *No*.
- XML is simple if you don't care about being correct.
- Standard answer: roll your own.
- Think outside the box: steal from XOM.

XOM

- Nowadays a language needs good libraries.
- Same principles.
- Converge needed an XML library. XML is easy, right? *No*.
- XML is simple if you don't care about being correct.
- Standard answer: roll your own.
- Think outside the box: steal from XOM.
- Thought: libraries effect users almost as much as languages.

Part IV: A gaze forward.

History will be kind to me, for I intend to write it.

- Winston Churchill (1874 - 1965)

Conclusions

- Language communities need to look outside their own box more.
  - Orthodoxies aren't always right.

Conclusions

- Language communities need to look outside their own box more.
  - Orthodoxies aren't always right.
- Language designers need to experiment more.
  - Look back as well as sideways.

Conclusions

- Language communities need to look outside their own box more.
  - Orthodoxies aren't always right.
- Language designers need to experiment more.
  - Look back as well as sideways.
- Paper writers should focus less on an individual language and more on generic issues.

Success is not final, failure is not fatal: it is the courage to continue that counts.

- Winston Churchill (1874 - 1965)