<u>Drift</u>

an imperative programming environment for the cloud

#1
Inspirations

Overview

- Rich Hickey
  - Value of values
  - Language of the system
- Cuneiform
- Unix / Bash
- Bret Victor
- Datomic
- Git

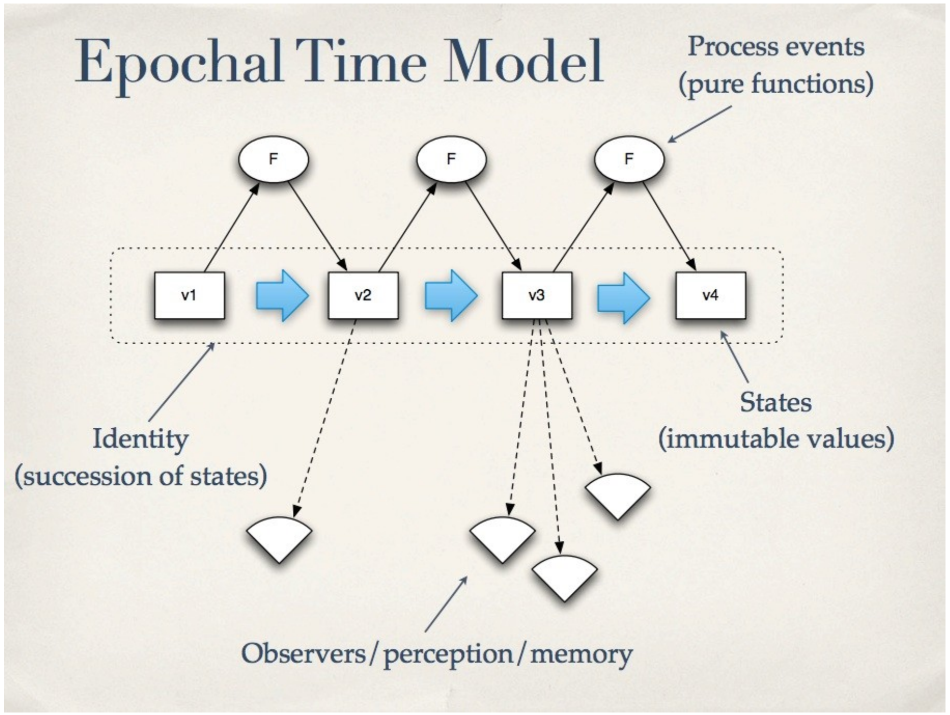
Rich Hickey

Rich Hickey

- started out as a music producer
- Java/C++ programmer for over a decade
- 'saw the light' with Lisp
- creator of *Clojure*
- creator of *Datomic*
- now CTO of *Cognitect*

Value of values

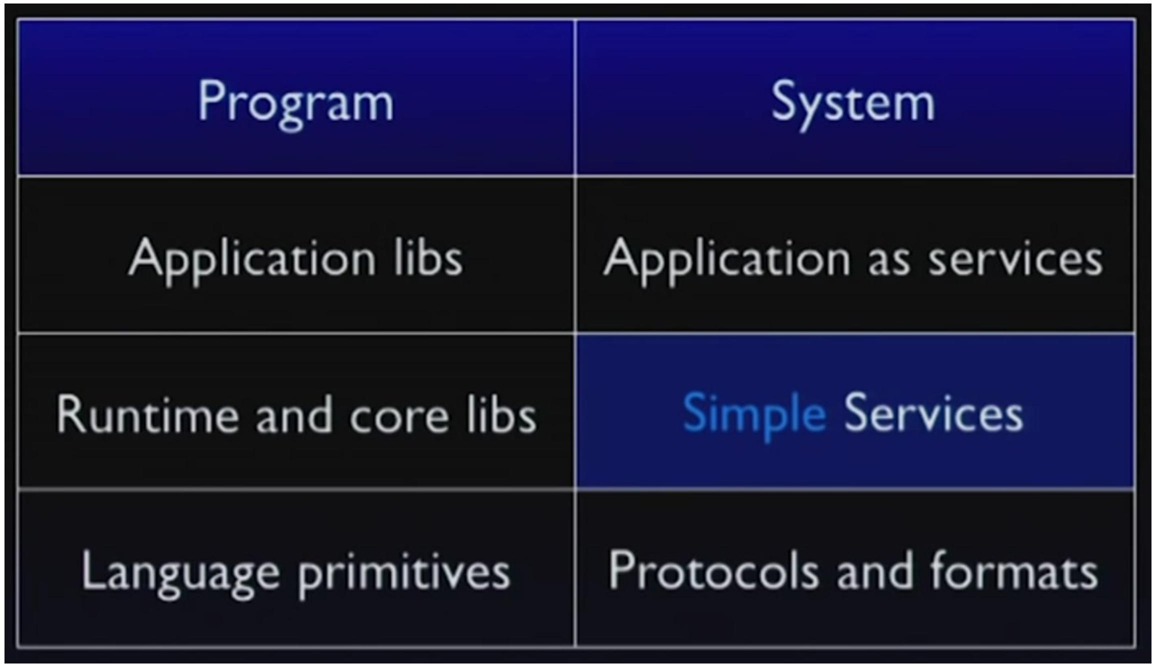
- introduces his concepts of *values* and *places*
- place:
  - mutable
  - writes overwrite current state
  - no history
- value:
  - more like a fact
  - immutable
  - can create history of facts


Epochal Time Model
Process events (pure functions)
F
F
F
v1
v2
v3
v4
Identity (succession of states)
States (immutable values)
Observers/perception/memory

Language of the System

- develop systems rather than programs
- use tools designed to write programs

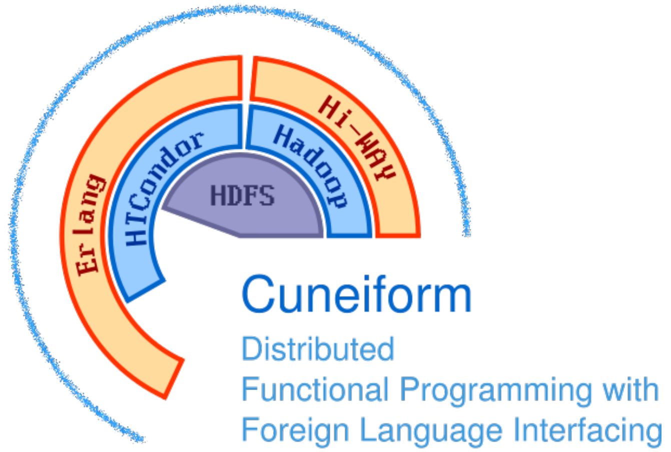
- 'systems language':
  - concepts?
  - invariants?
  - properties?
  - protocols?



Program	System
Application libs	Application as services
Runtime and core libs	Simple Services
Language primitives	Protocols and formats

Cuneiform

- workflow specification language
- functional
  - immutable
  - lazy
  - second order functions
- forreign function interface
- distributed (parallel?)



© Jörgen Brandt, HU-B.WBI

Cuneiform

```
deftask untar (<list(File)> : tar(File)) in bash *{
  tar xf $tar
  list=`tar tf $tar`
}*

txt = untar(tar: 'corpus.tar');
csv = wc(txt: txt);
result = groupby(csv: csv);
result;
```

Cuneiform

- program implicitly forms a tree
- like in Lips or λ-calculus
- every 'task invocation' must be bound to a name
- query operator finally triggers tree reduction and 'execution'

Unix

“If I had more time,

I would have written a shorter letter.”

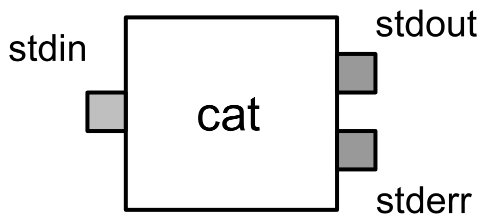
Unix

- advantage: full stack
  - *shell:* oblivious coordination
  - *C:* oblivious calculation
  - *kernel:* API, fs, memory, sockets, ...
- my linux is a 'system':
  - running processes
  - data they consume/produce
  - means to spawn and orchestrate them
  - why isn't 'Bash' considered a coord/workflow language?

Bash

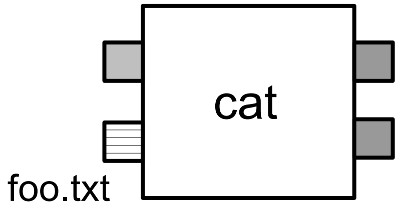
```
$ cat
```

- default conf for basic unit: stdin, stdout, stderr



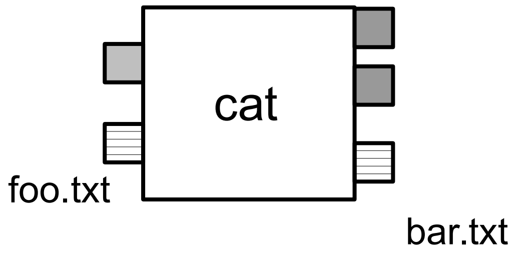
```
$ cat foo.txt
```

- overwrite default conf with string (file?)
- who knows it's a file?



```
$ cat foo.txt > bar.txt
```

- write output to file? or bind output to a new name?



What about `` `foo bar` ``, `$` and `A | B` ?

Bash

Bash	Functional
&	- (lazy)
|	function composition
>, >>	name binding
<	-
$	eval

Bret Victor

- “Human Interface Inventor” at Apple
- Talks:
  - “Inventing on Principle”
  - “Media for thinking the unthinkable”
  - “The Future of Programming”
- Most notable idea: 'immediate feedback'

Bash

- can keep the language (or most of it)
- keep the 'interactiveness' of the shell
  and visualize it

- Problem: FS is shared mutable state
  → need immutable FS ...
  (the file system as a value...)

Datomic

The Database as a Value

Rich Hickey

- first functional DB
- ACID but no CRUD
- append-only log of 'facts'
- can be implemented by any stateful DB/store

Entity	Attribute	Value	Transaction
421	:student/email	foo@bar.com	12345
421	:student/email	new@bar.com	12346

Git

- used as: version control system
- users change files *in-place*
- system keeps *full* history of all 'states'
- internal structure looks like an immutable FS based on hash trees (Merkle trees)

- “Problem”: destructive ops → merge conflicts...

Drift

- interactive system shell & language like bash
  → start and orchestrate services (task + data view)

- stateful virt. FS that is actually immutable
  → extensive cashing between sessions

- FS which is 'streaming by default'
  → emulated using 2 diff. message queues

- interactive graph visualization of the system
  → using Petri Net syntax