The Value of Values

Rich Hickey

Datomic, Clojure
Information

Inform

‘to convey knowledge via facts’
‘give shape to (the mind)’

Information

the facts
What is a Fact?

Place where specific information is stored
There is a place for every piece of information
Facts have operations, e.g. get and set
Operations control how facts can change
To convey a fact, convey its location
Place

‘A particular portion of space’
‘An area used for a particular purpose’
Memory address, disk sector
‘Information’ Systems

In memory

mutable objects are abstractions of places

objects have methods

In durable storage

tables/documents/records are places

DBs have update
PLOP

PLace-Oriented Programming

New information replaces old

Born of limitations of early computers

small RAM and disks

Those limitations are long gone
The Efficiencies of Place

Ok, when ‘birthing’ new values

birthing == prior to perceptibility

i.e. prior to becoming a fact

But: an implementation detail

I.T., not T.T.
Memory and Records

We’ve co-opted
and believe our own mythos
Mental memory is associative and open
Real records are enduring
and accreting
not erase and overwrite
The Point

Values have many advantages

in process

across processes

in storage

We know these things

Place has no role in an information model
Value

‘Relative worth’

‘A particular magnitude, number or amount’

‘Precise meaning or significance’
Is a String a Value?

Is it immutable?

Equality, comparability are basis for logic

Who wants to go back to mutable Strings?
Programming Values

Immutable

Don’t need methods

I can send you values without code

and you are fine

Are semantically transparent

Can be abstracted
Values Can be Shared

Share freely

aliases are free

No one can mess you up

nor you them

Incremental change is cheap

PlACES

Defensive copy, clone, locks
Reproducible Results

Operations on values are stable

Testing

Debugging

reproduce failures w/o replicating state

Places

must establish matching ‘state’ first
Easy to Fabricate

 Anything can create compliant values for testing, simulation

 Places must emulate operational interface
Thwart Imperativeness

Values refuse to help you program imperatively

That’s a feature

Imperative code is inherently complex

Places

Encourage and require imperativeness
Language Independence

Pure values are language independent

the polyglot tool

Places are defined by language constructs (methods)

can be proxied, remoted, with much effort
Values are Generic

Representations in any language

Few fundamental abstractions

for aggregation (lists, maps, sets)

Places

Operational interface is specific

More code

Poor reuse
Values Are the Best Interface

For subsystems can be moved ported enqueued

Places application, language and flow coupled
Values Aggregate

Values aggregate to values

So all benefits accrue to compositions

Places

Combinations of places, what properties?

Need new operational interface for aggregate
Extended Value Propositions

Mechanism for conveyance and perception
Mechanism for memory
Reduced coordination
Location flexibility

Essential for decision making
Conveyance

In the small

Aliases of values convey value
Mutable things on queues convey nothing

In the large

Values rule on the wire
No reproducible values in PLOP DBs
Perception

In the small

Values: to reach is to perceive

Places: How to perceive a coherent value of object with multiple getters?

In the large

Values still rule on the wire

No reproducible values in PLOP DBs
Memory

In the small

Values: remembering == aliasing

Places: copy, if you can

In the large

What if there were no permalinks?

Place-oriented DBs - DIY time
Reduced Coordination

In the small

Values: No locks!

Places: Lock policies don’t aggregate

In the large

No read transactions!

PLOP: Often gotten wrong
Location Flexibility

In the small

Values: aliasing means only one copy

Places: master copy is special

In the large

Cache (e.g. HTTP caching)

CDN etc

Data-based interface is movable
Facts are Values

Not places

Don’t facts change?

No - they incorporate time

Fact - ‘an event or thing known to have happened or existed’

From: factum - ‘something done’
Facts != Recent Facts

Knowledge is derived from facts

Comparing

Combining

Especially from different time points

You cannot update a fact

any more than you can change the past
Information Systems

Are fundamentally about facts

Maintaining, manipulating

To give users leverage

Making decisions

Systems should be value-oriented

Don’t use process constructs for information
Decision Making

We know what it takes to support our own decision making (hint: information)

Compare present to past

Spot trends, rates

Aggregates

Often requires time
Programmer I.T.

Source Control
  Update in place? - No
  Timestamps - of course!

Logs
  Update in place? - No
  Timestamps - of course!
Big Data

Business to programmers:

“I like your database better than the one you gave me”

Logs have all the information and timestamps

We are reactive here

mining logs, seriously?

Not delivering leverage
The Space Age

Space

‘The unlimited expanse in which all things are located, and all events occur’

If new never fails...

you are effectively running in space

If S3 never fills up...

it is not the cloud, but space
New Facts, New Space

The end of PLOP

If you can afford this, why do anything else?
You can afford this

(there will be garbage)
Summary

We continue to use place-oriented programming languages and databases
and make new ones!

long after rationale is gone

We are missing out on the value of values
which we recognize

We need to deliver information systems
demand is clear, resources available
Facts do not cease to exist because they are ignored.

Aldous Huxley