Rules as a Control Structure

Ryan Brush

Let’s code some business logic

Delivery date must be more than 10 business days away

Must have a 27B/6

Ensure compatible parts

Process a Work Order

Additional charge for expedited orders

Approval required if total cost > X

Ensure regulatory paperwork complete

27B/6?

How do we code this?

```
(defn process-order [order]
;;TODO: write giant mess of logic
)
```

```
(defn process-order [order]
    ;; TODO: write smaller messes of logic
    ;; and tightly couple them together
    )
```

“Restricted elements depend on location and cannot be used without manager approval or with a justification form.”

"Restricted elements depend on location and cannot be used without manager approval or with a justification form."

"Restricted elements depend on location and cannot be used without manager approval or with a justification form."

"Restricted elements depend on location and cannot be used without manager approval or with a justification form."

Excessive plumbing is not a requirement

Can we get rid of it?

How our requirements look

How our code looks


MIND THE GAP

Simple things should be simple to do. -Alan Kay

So how can we close the gap?

Delivery date must be more than 10 business days away

Must have a 27B/6

Ensure compatible parts

Process a Work Order

Additional charge for expedited orders

Approval required if total cost > X

Ensure regulatory paperwork complete

Write independent rules

Let the system do the plumbing

Drools

Nools

Jess

Rule Engines

OPS5

CLIPS

Rule engines do the plumbing

when:
  item I restricted at location L
  work W order at location L
then:
  approval required

when:
  approval required
  no approval form
then:
  reject work order

But there are downsides

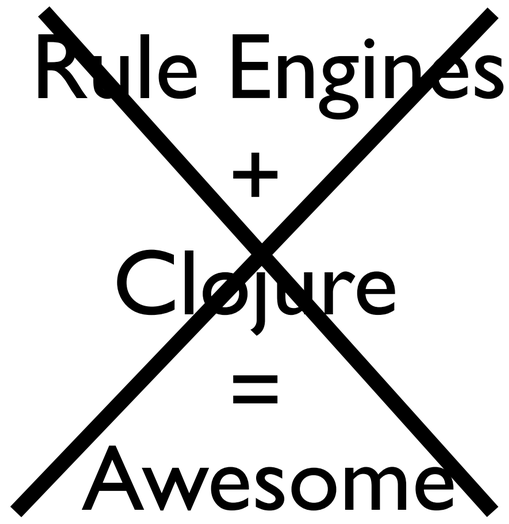
Simplicity of a DSL limits expressiveness

Obstacles to invoking arbitrary functions

Working memory is mutable

No direct rule introspection

Brush's Conjecture




```
(= “Awesome”
   (+ “Clojure”
      “Rule Engines”))
```

Emacs Time!

Questions?