# Maneuverable Web Architecture

Michael T. Nygard - Cognitect

# Thesis

# Thesis

- Agile dev works at micro scale

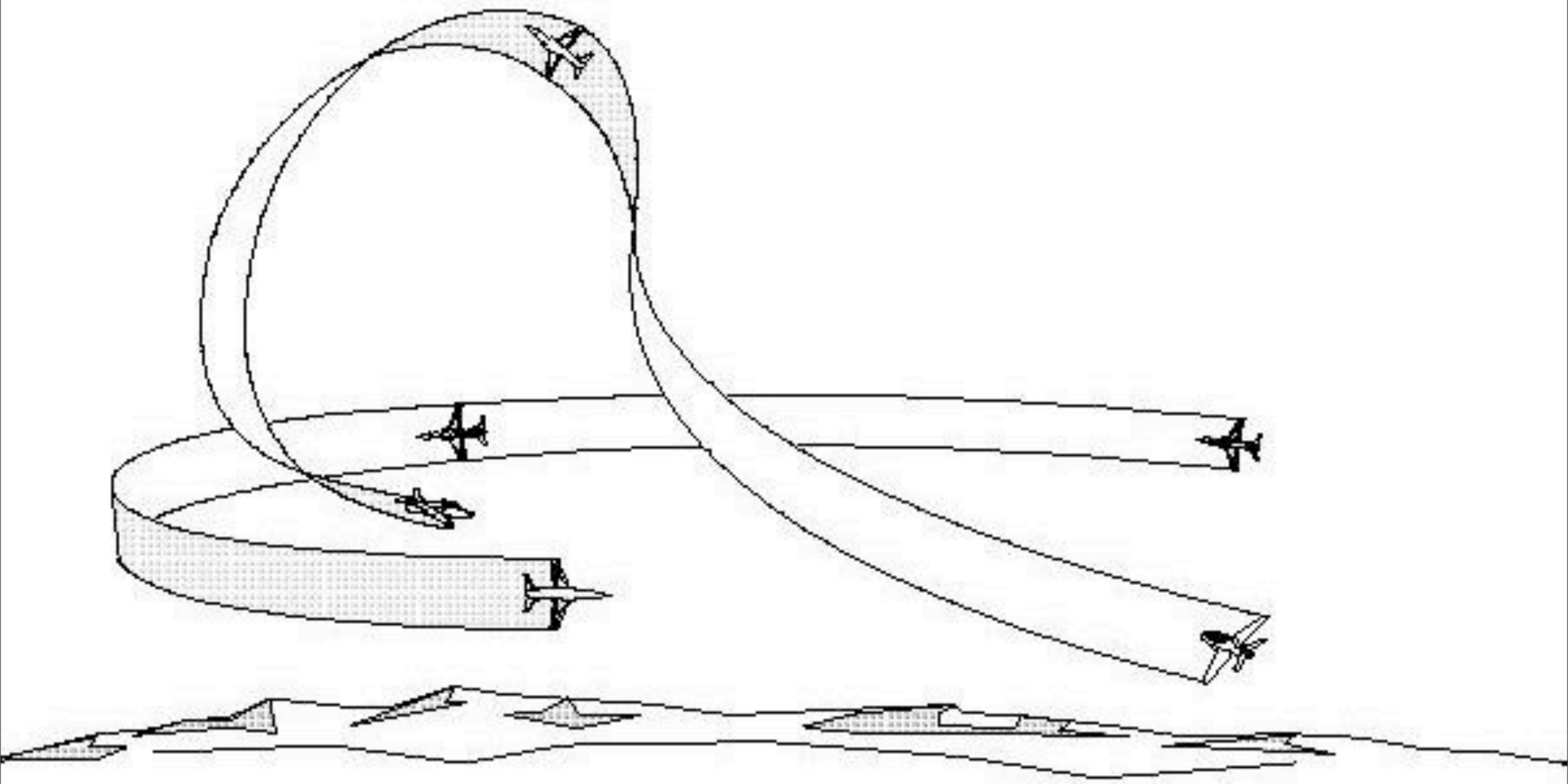
# Thesis

- Agile dev works at micro scale
- Won't create macro scale agility

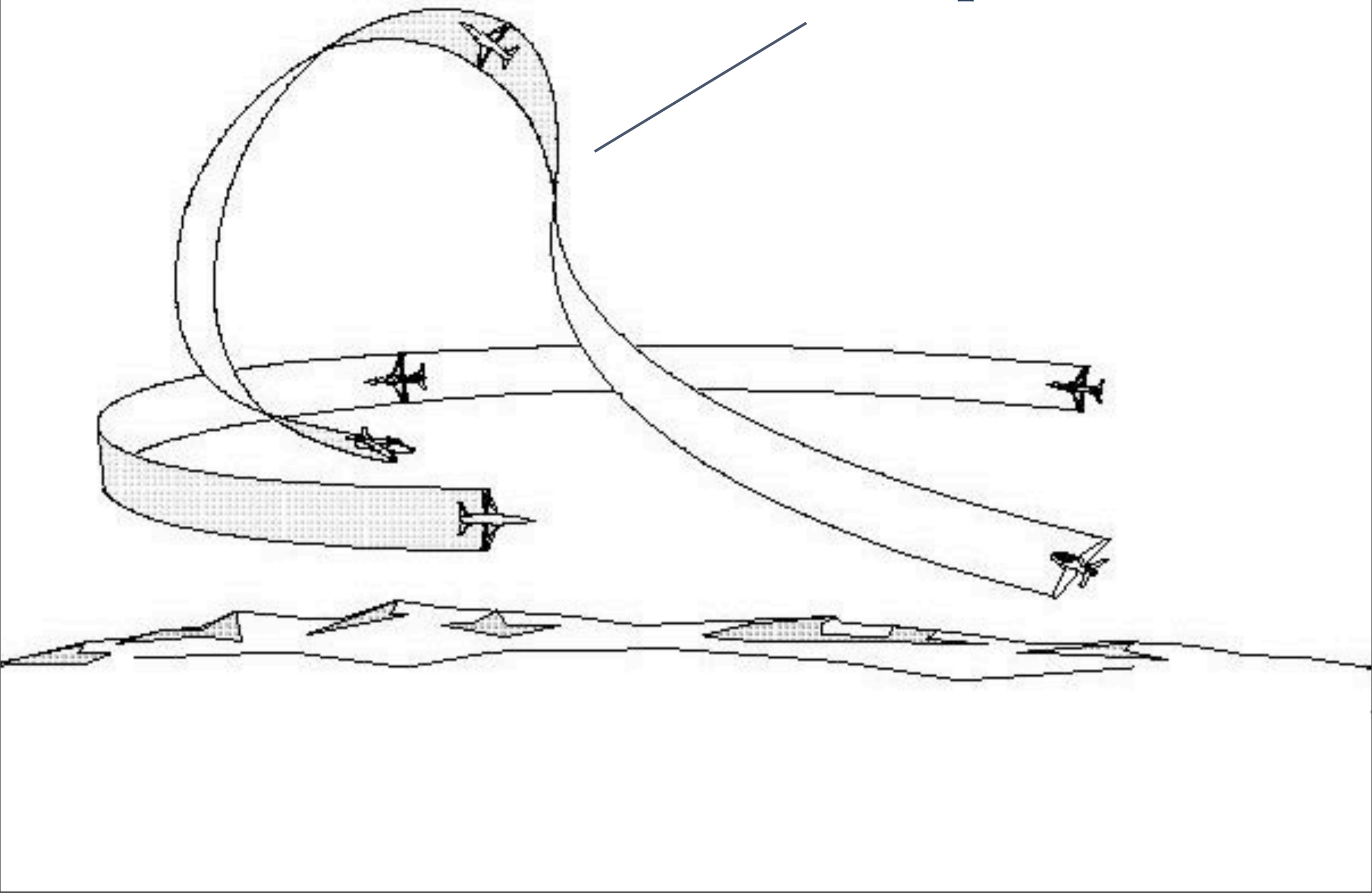
# Maneuverability

# John Boyd

- Fighter pilot
- Air combat instructor
- Military theorist

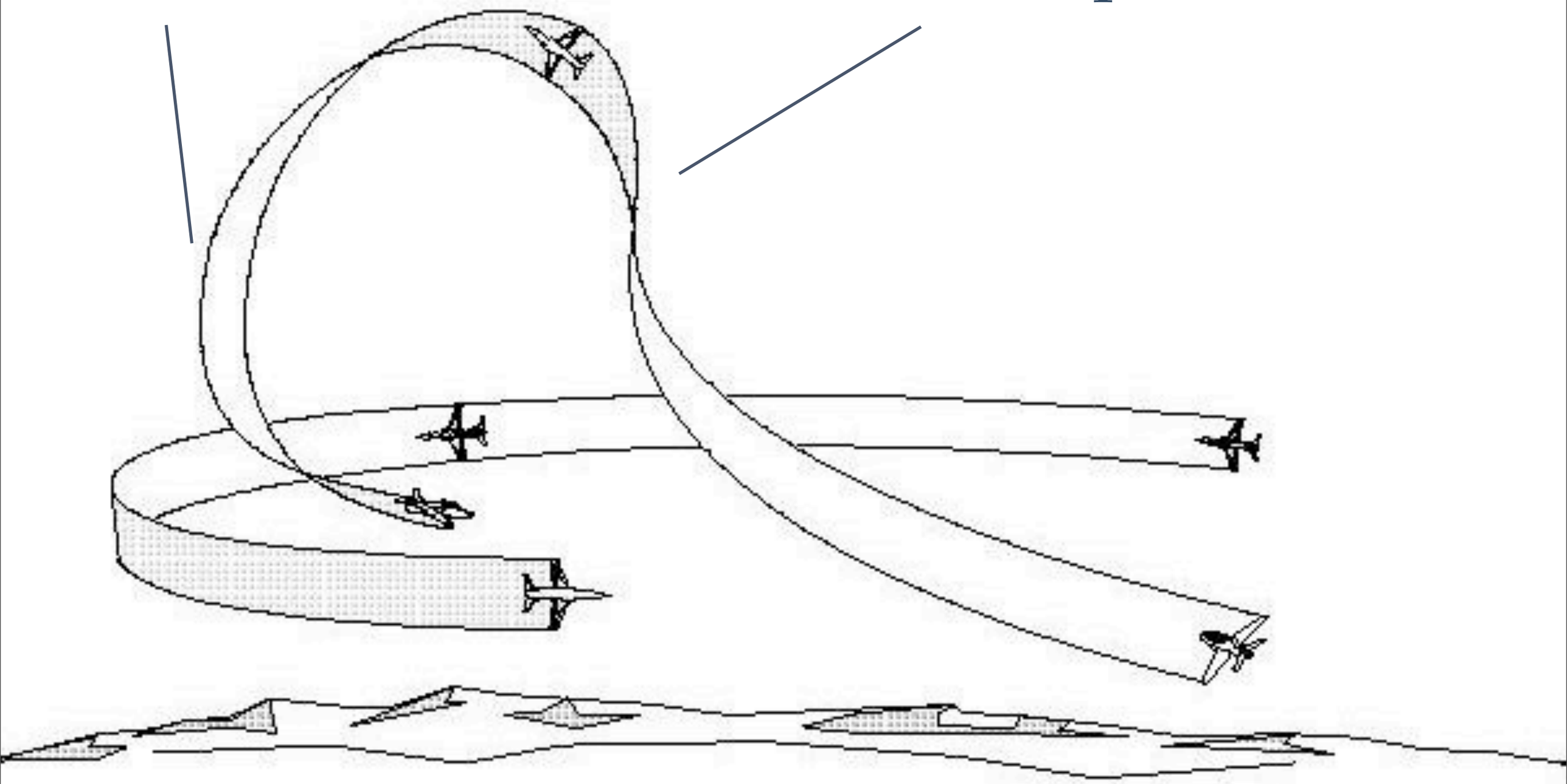


# Kinetic to potential



Potential to kinetic

Kinetic to potential



# Energy-Manueverability (EM)

# Energy-Manueverability (EM)

## Kinetic ⟷ Potential

# Energy-Manueverability (EM)

## Kinetic ⟷ Potential

## Gain and Shed Momentum

# Energy-Manueverability (EM)

Kinetic ⟷ Potential

Gain and Shed Momentum

Rapidly Change Maneuvers

# Maneuver Warfare

# Maneuver Warfare

## Control tempo of engagement

# Maneuver Warfare

## Control tempo of engagement

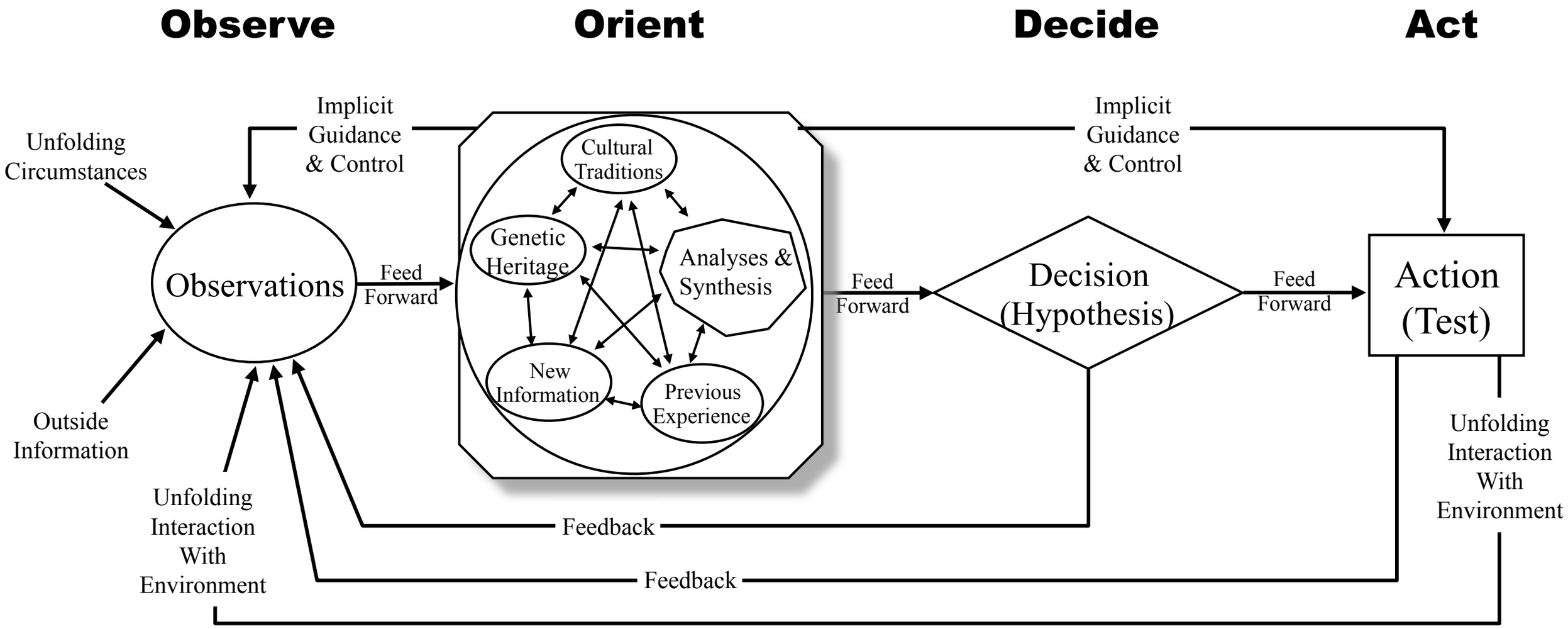
## Take initiative

# Maneuver Warfare

## Control tempo of engagement

## Take initiative

## Send ambiguous signals


Observe
Orient
Decide
Act
Unfolding Circumstances
Implicit Guidance & Control
Observations
Feed Forward
Cultural Traditions
Genetic Heritage
Analyses & Synthesis
New Information
Previous Experience
Implicit Guidance & Control
Feed Forward
Decision (Hypothesis)
Feed Forward
Action (Test)
Outside Information
Unfolding Interaction With Environment
Feedback
Unfolding Interaction With Environment
Feedback

# Maneuverable Web Architecture

Control tempo of engagement

Take initiative

Send ambiguous signals

“From now on, I control the tempo.”

# Cross-cutting Themes

- Plurality
- Break monoliths
- Use URIs with abandon
- c.f. “Architecture Without an End State”
  - Augment upstream
  - Contextualize downstream

# Techniques, not patterns… yet

# Sufficiently Abstract UI

Counterexample:

100+ countries.

6 services per country.

UI knows which services to call

UI has forms specific to each one.

# Sufficiently Abstract UI 2

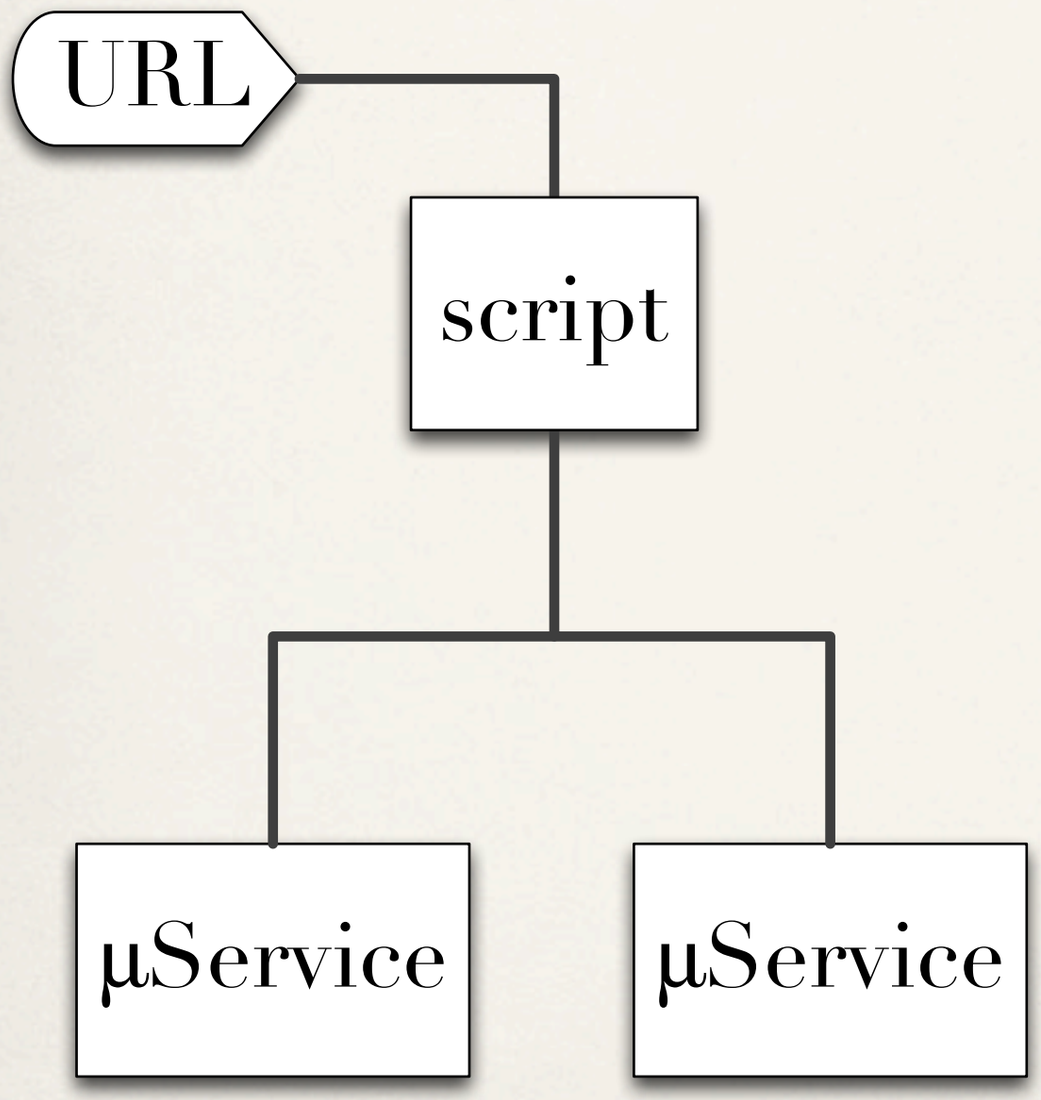
Better:

Generic UI +
Semantic HTML +
Unobtrusive JavaScript

Best:

Above, plus either SSR or CSR, with CMS & toolkit for adaptable components.

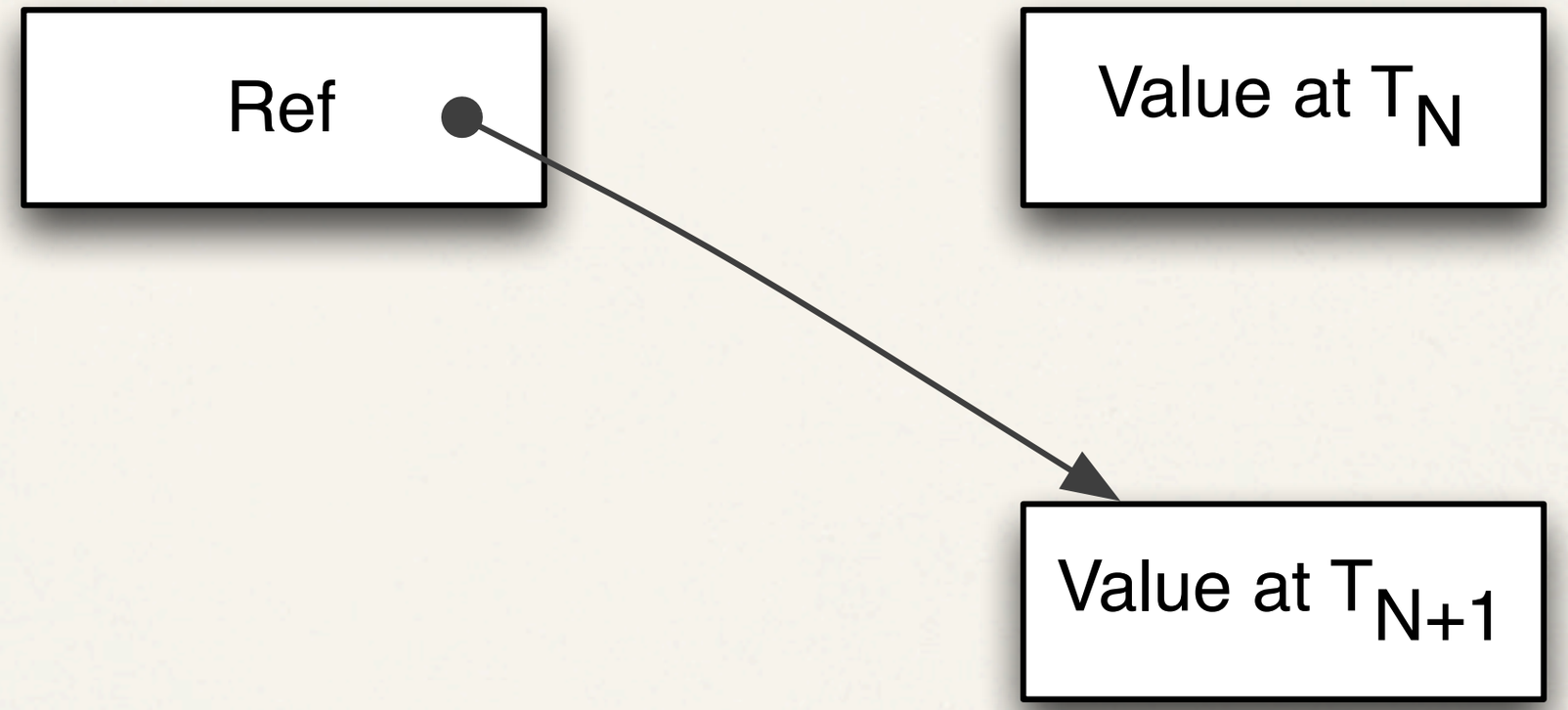
# Services & scripting




- Dynamic deployment of script
- Script addressable by URL

# Immutable values, monotonic succession

# Immutable values, monotonic succession



| Value Semantics | Ref Semantics |
| --- | --- |

| Value Semantics | Ref Semantics |
| --- | --- |
| Values do not change | Change is atomic<br>No observable intermediate states |

| Value Semantics | Ref Semantics |
| --- | --- |
| Values do not change | Change is atomic<br>No observable intermediate states |
| Equality is identity | Equality based on referent |

# Example: Perpetual string

- **Perpetual string**: store strings forever.
- URL is SHA-256 hash of string.
- Use for scripts, legal text.
- Edit the script, get a new URL

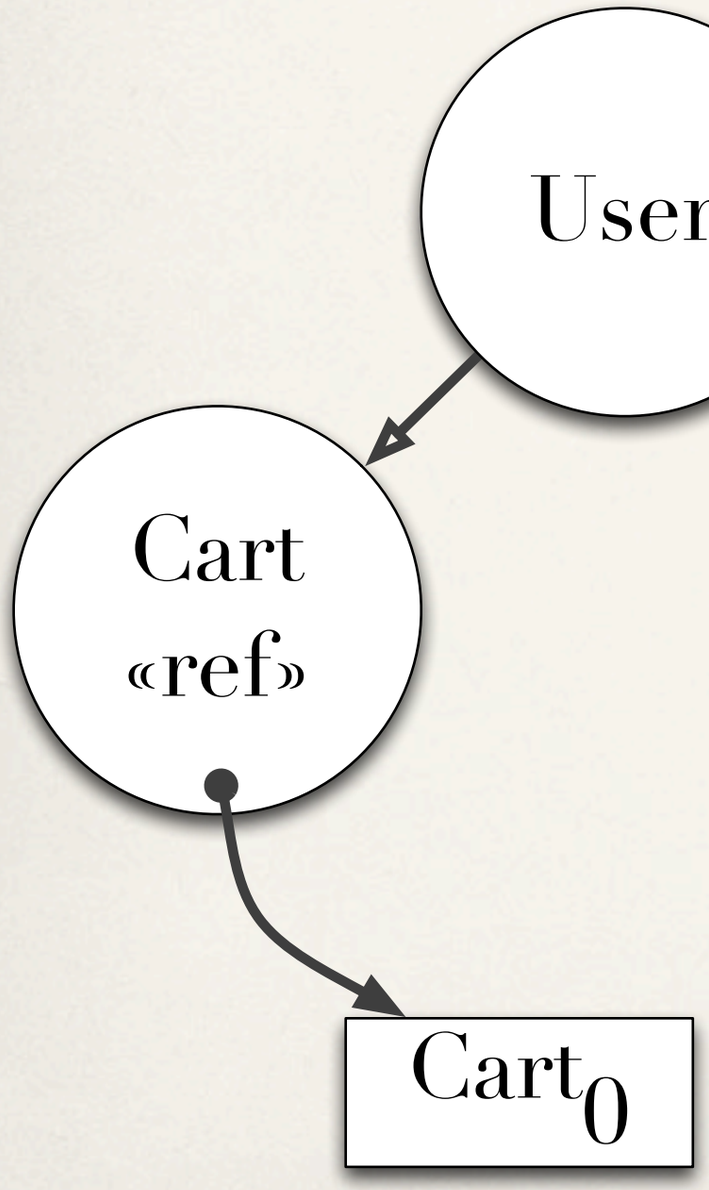
# Example: Shopping Cart as Value

Cart is a number

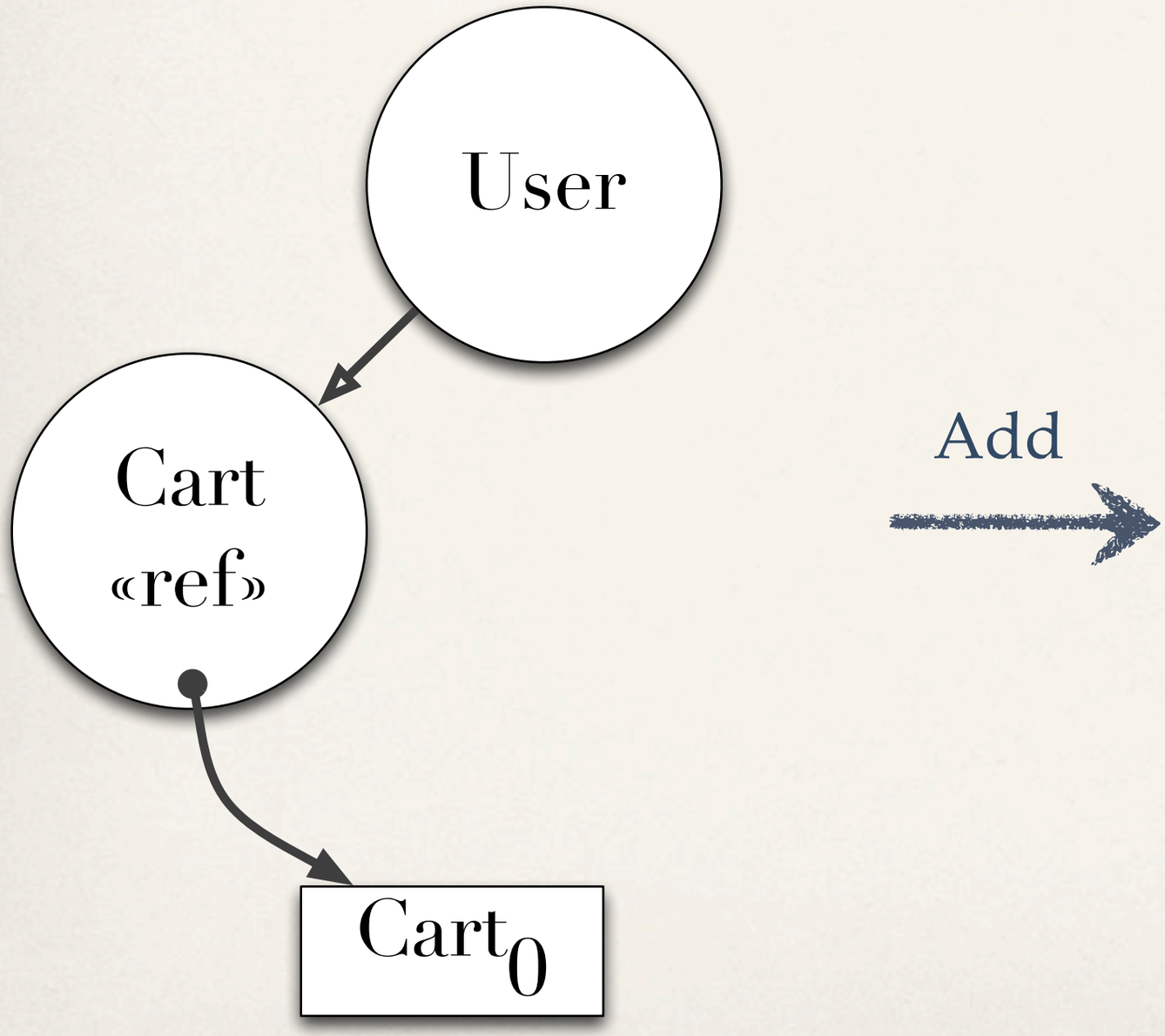
Add: function from Cart, Item, Qty to Cart

Remove: function from Cart, Item to Cart

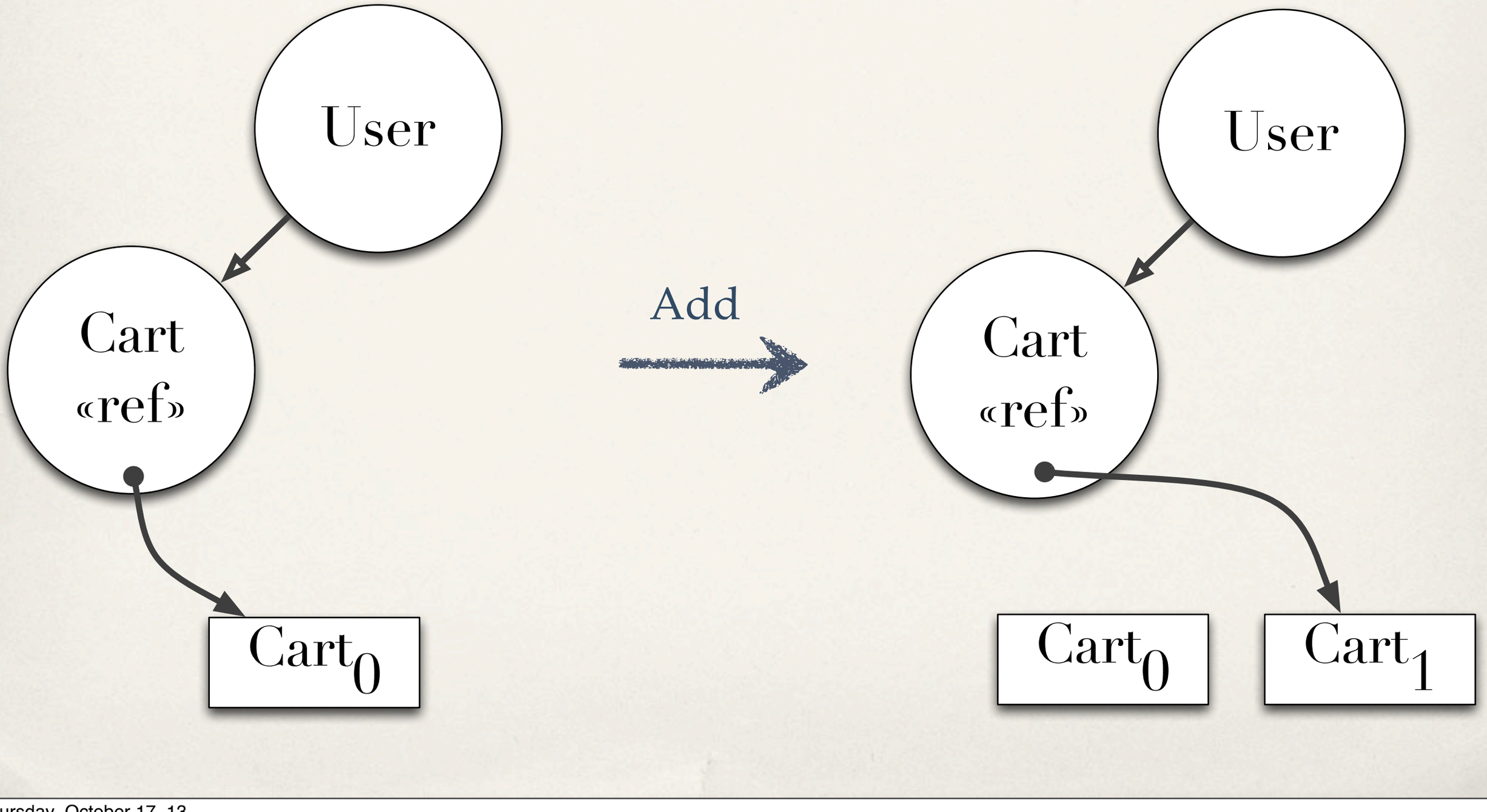
# Example: Shopping Cart as Value



# Example: Shopping Cart as Value



# Example: Shopping Cart as Value



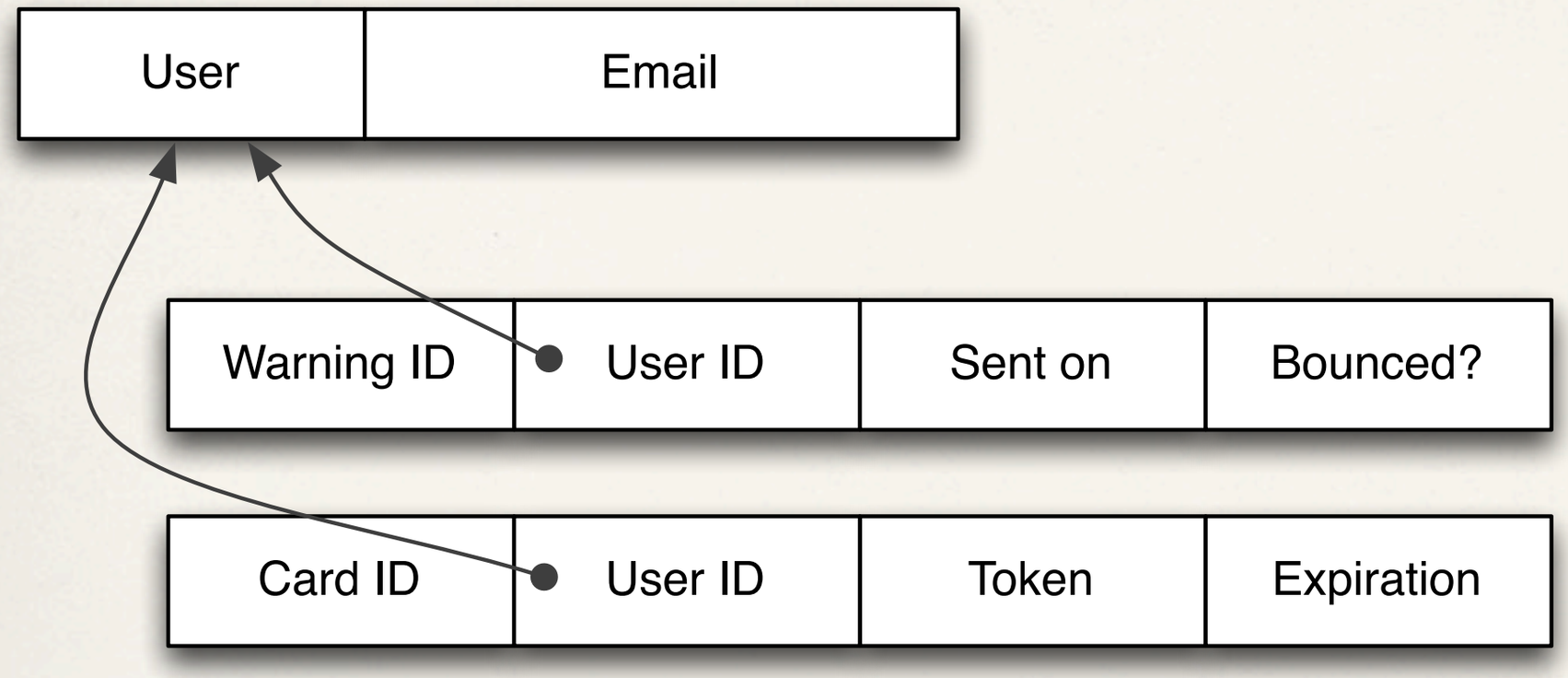
# Generalized minimalism

Feature: Send email to customer ahead of credit card expiration.

Required: Record email sent, bounced.

Complexity: Number of reminders and advance notice vary by tenant.

# Complected solution




Daily job wakes up, scans cards, creates warnings, sends email, checks bounces.

# Minimal, generalized solution

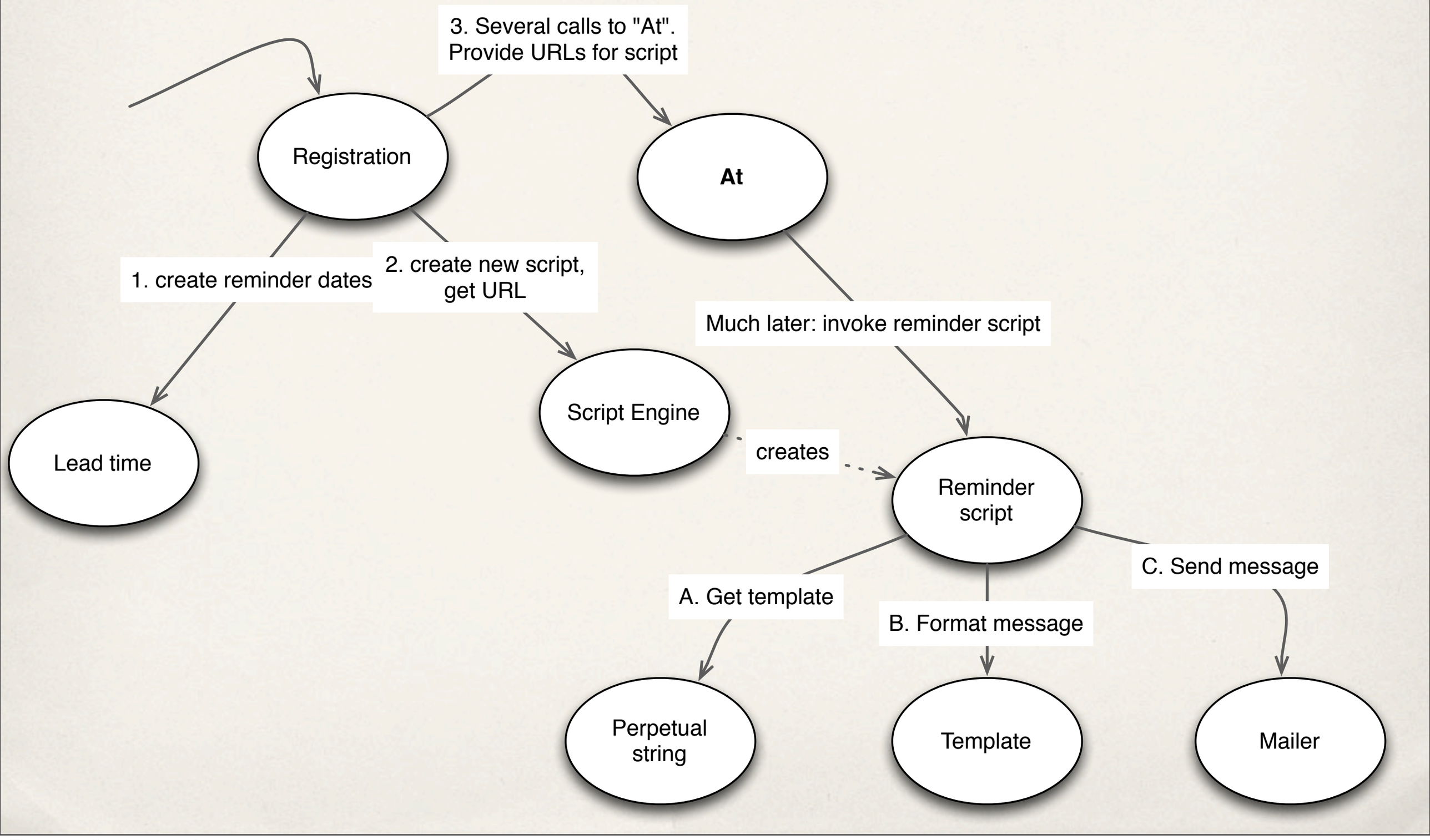
**At**: at datetime, call this URL.

**Template**: accept body + params to format text

**Lead time**: generate series of datetimes

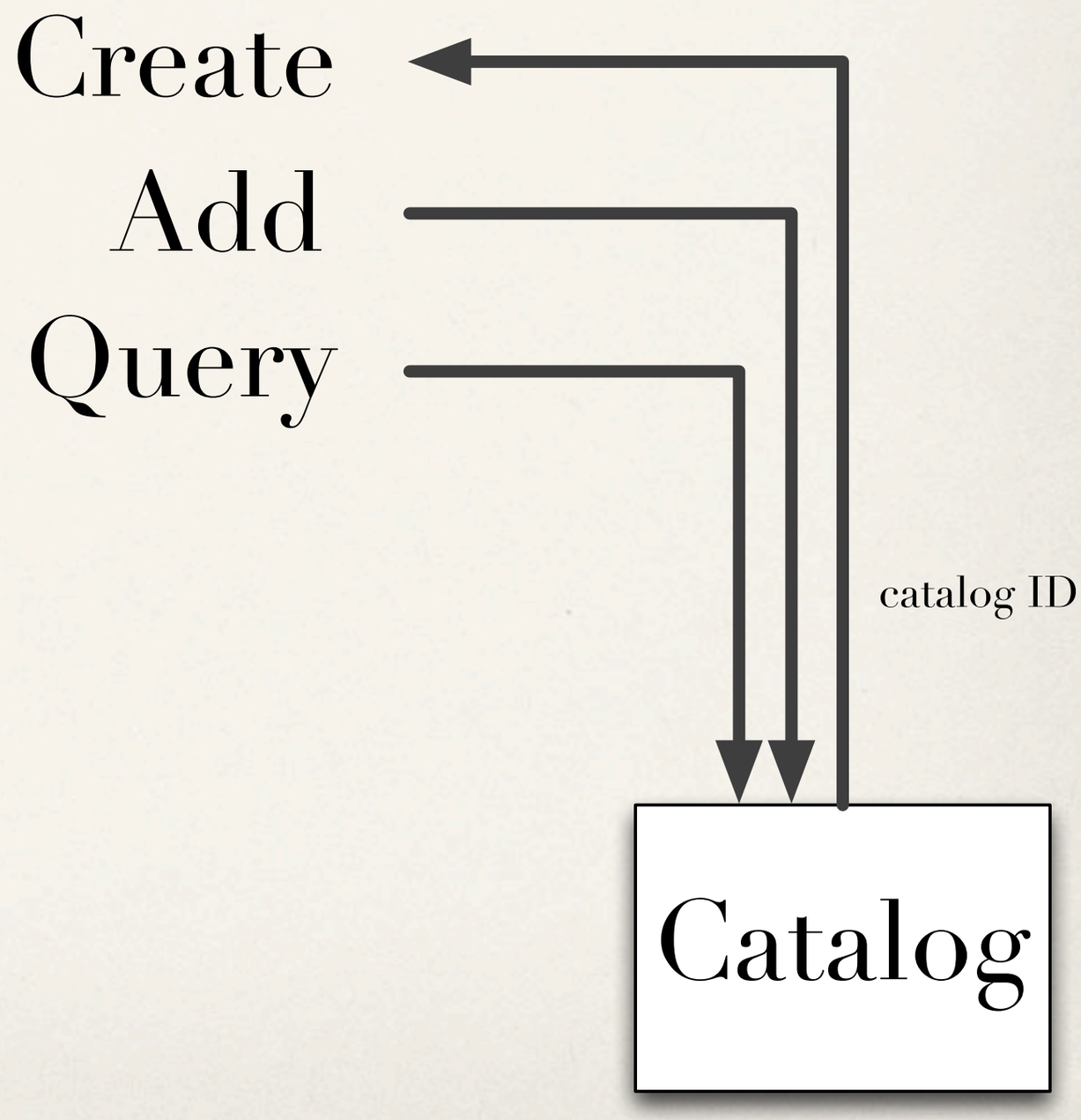
**Mailer**: send email to addr, track bounces

# Minimal, generalized solution

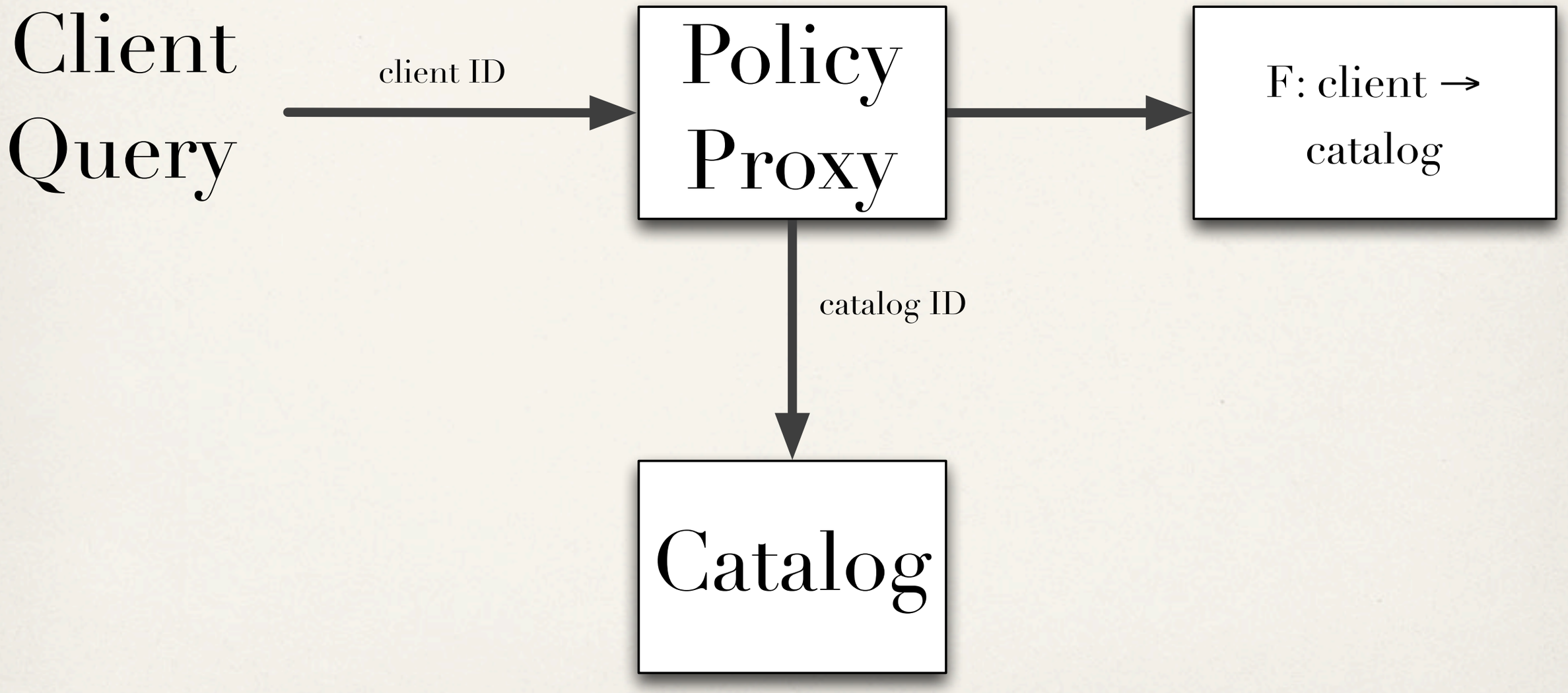


# Issue identifiers

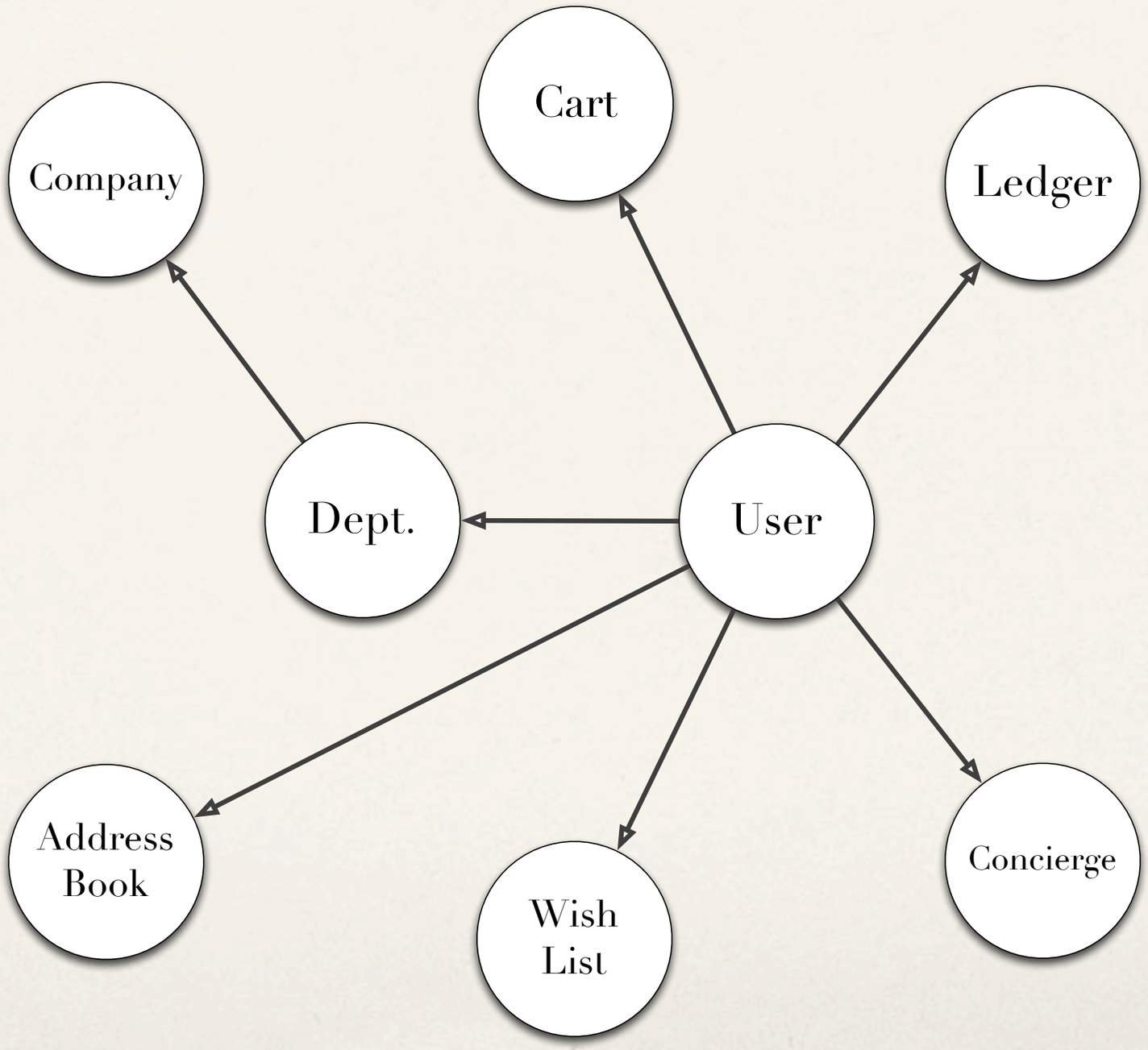
- Every service issues identifiers
- No restrictions on use



# Policy Proxy

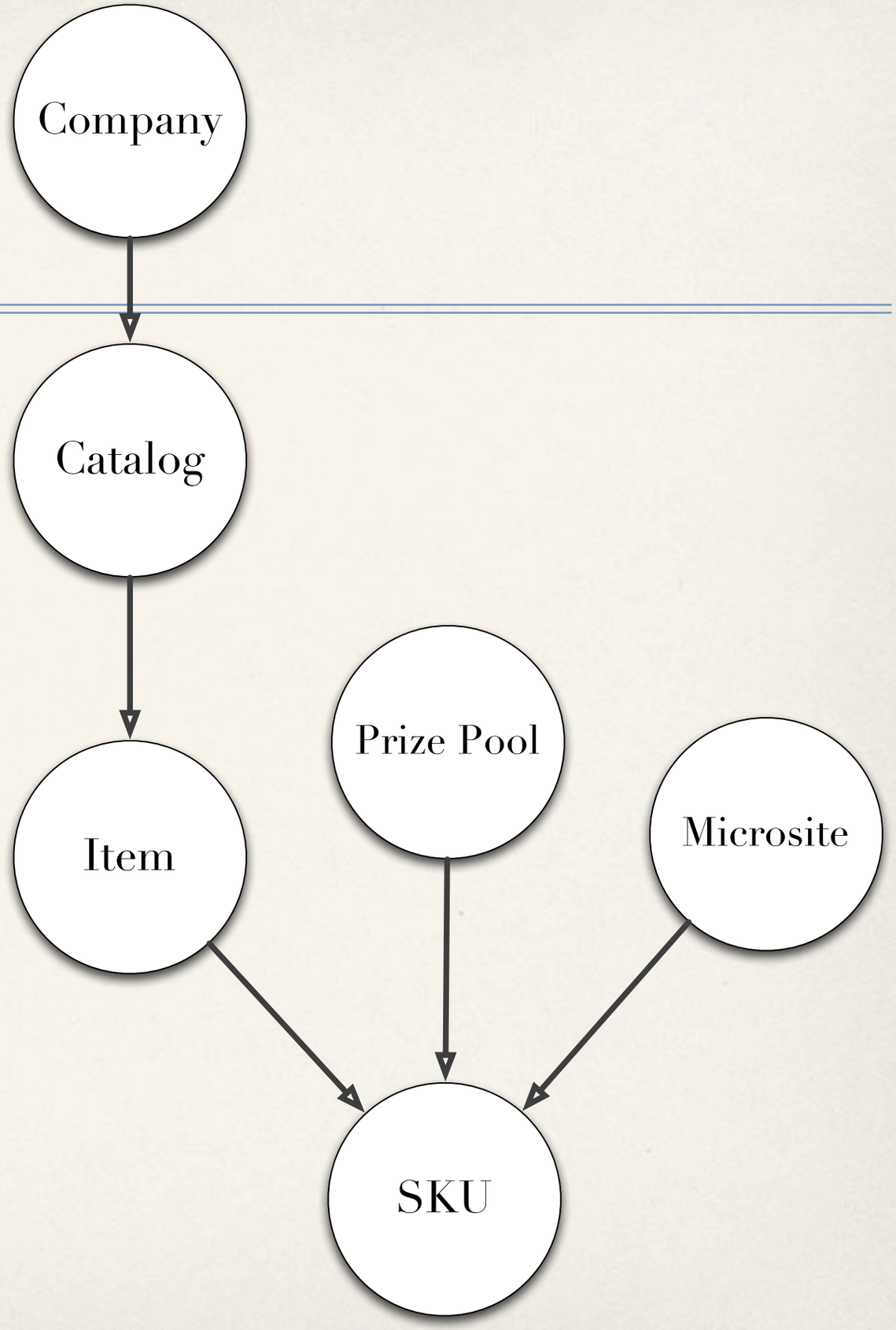


# Faceted identities



# Faceted identities

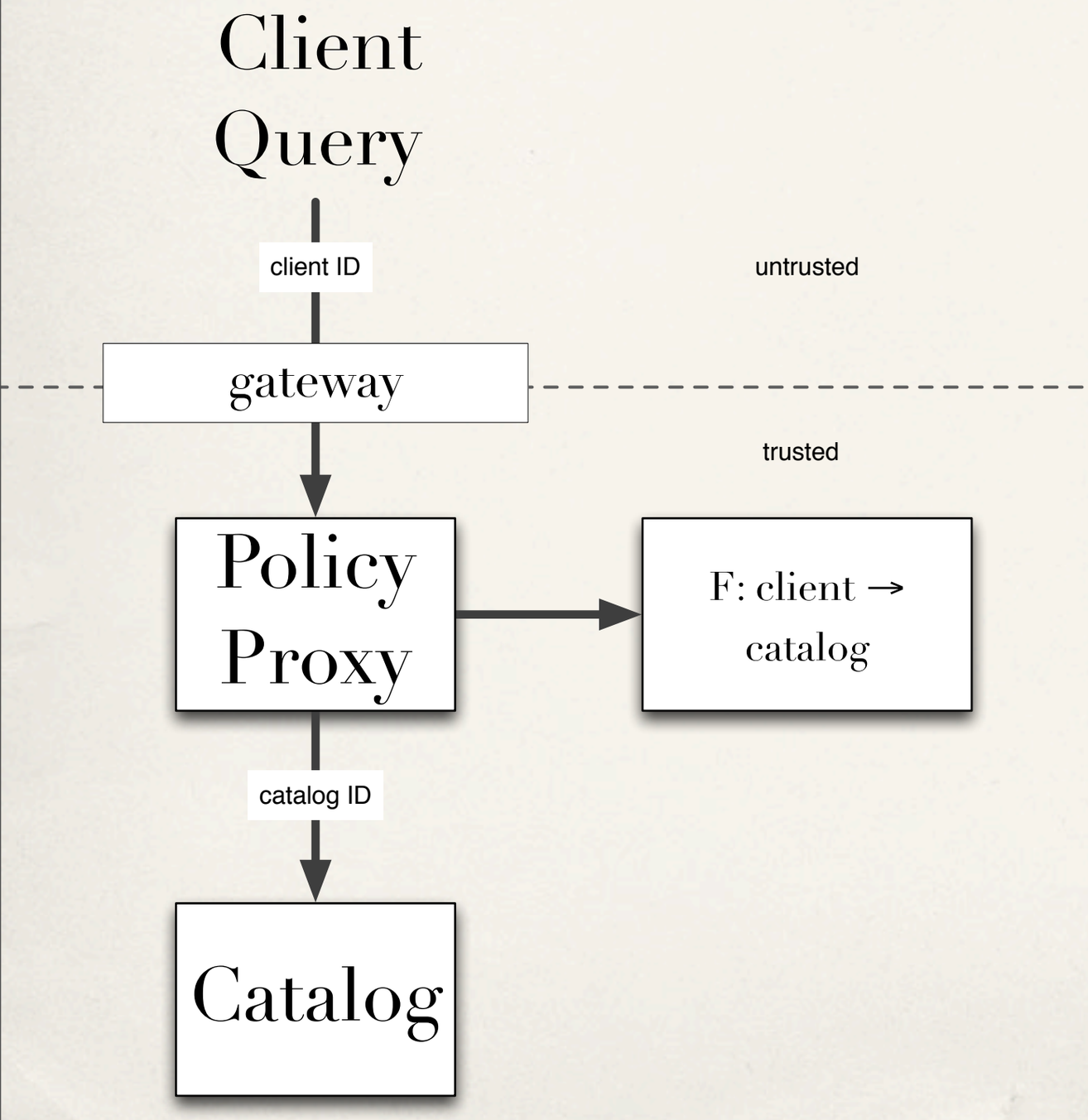
- IDs all issued by services
- Relationships externalized



# Explicit context

| Implicit | Explicit |
| --- | --- |
| Bare identifiers | URLs |
| State names | State machines |
| Assumed channel | Reply-to queue |

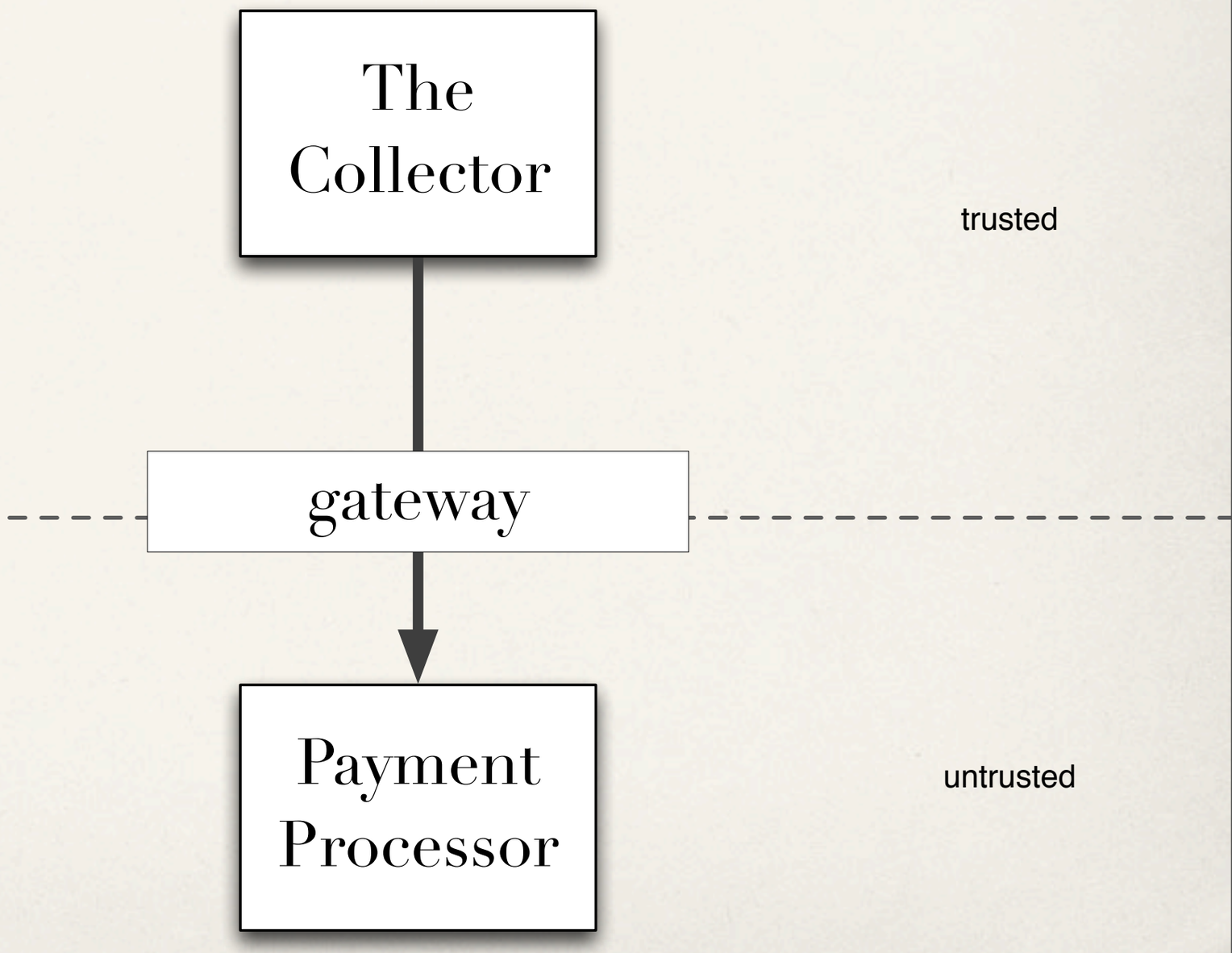
# Self-defense




- Required for use without permission
- Protect from overload
- Allow cut off of maluser
- Important for operational safety and authorization

# Self-defense 2

- Also applies to outcalls



# Half-duplex testing

**Wrong**

1. Set up mock

2. Set up call

3. Make call

4. Assertions about result

5. Verify mock was called

# Half-duplex testing

**Wrong**

1. Set up mock
2. Set up call
3. Make call
4. Assertions about result
5. Verify mock was called

**Right - call side**

1. Set up mock
2. Set up call
3. Make call
4. Verify mock was called correctly.

# Half-duplex testing

**Right - response side**

1. Skip mock & call

2. Inject fake results

3. Verify results handled correctly.

**Right - call side**

1. Set up mock

2. Set up call

3. Make call

4. Verify mock was called correctly.

# Other techniques I'm not 100% sure about

- Separate query from action
- Never deploy together
- Tell, don't ask
- Unbundle parameters

# Maneuverability and tempo

# Thanks!

Michael T. Nygard
Cognitect

mtnygard@cognitect.com
@mtnygard