Maneuverable Web Architecture
Michael T. Nygard - Cognitect
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 $\leftrightarrow$ 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
Note how orientation shapes observation, shapes decision, shapes action, and in turn is shaped by the feedback and other phenomena coming into our sensing or observing window. Also note how the entire loop (not just orientation) is an ongoing many-sided implicit cross-referencing process of projection, empathy, correlation, and rejection.

From The Essence of Winning and Losing, John R. Boyd, January 1996.
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

Ref \rightarrow Value at T_N
Immutable values, monotonic succession

Ref

Value at $T_N$

Value at $T_{N+1}$
<table>
<thead>
<tr>
<th>Value Semantics</th>
<th>Ref Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values do not change</td>
<td>Change is atomic</td>
</tr>
<tr>
<td>No observable intermediate states</td>
<td>Equality is identity</td>
</tr>
<tr>
<td>Equality based on referent</td>
<td></td>
</tr>
<tr>
<td>Value Semantics</td>
<td>Ref Semantics</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Values do not change</td>
<td>Change is atomic</td>
</tr>
<tr>
<td></td>
<td>No observable intermediate states</td>
</tr>
<tr>
<td>Value Semantics</td>
<td>Ref Semantics</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Values do not change</td>
<td>Change is atomic</td>
</tr>
<tr>
<td>No observable intermediate states</td>
<td></td>
</tr>
<tr>
<td>Equality is identity</td>
<td>Equality based on referent</td>
</tr>
</tbody>
</table>
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.
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

1. create reminder dates
2. create new script, get URL
3. Several calls to "At". Provide URLs for script

Registration

At

Script Engine

Much later: invoke reminder script

Reminder script

A. Get template
B. Format message
C. Send message

Perpetual string
Template
Mailer

Lead time

Thursday, October 17, 13
Issue identifiers

- Every service issues identifiers
- No restrictions on use
Policy Proxy

Client Query → Policy Proxy → Catalog

F: client → catalog

client ID

catalog ID

Catalog

Thursday, October 17, 13
Faceted identities

- Company
- Cart
- Ledger
- Dept.
- User
- Address Book
- Wish List
- Concierge
Faceted identities

- IDs all issued by services
- Relationships externalized
## Explicit context

<table>
<thead>
<tr>
<th>Implicit</th>
<th>Explicit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bare identifiers</td>
<td>URLs</td>
</tr>
<tr>
<td>State names</td>
<td>State machines</td>
</tr>
<tr>
<td>Assumed channel</td>
<td>Reply-to queue</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Right - response side</th>
<th>Right - call side</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Skip mock &amp; call</td>
<td>1. Set up mock</td>
</tr>
<tr>
<td>2. Inject fake results</td>
<td>2. Set up call</td>
</tr>
<tr>
<td>3. Verify results handled correctly.</td>
<td>3. Make call</td>
</tr>
<tr>
<td></td>
<td>4. Verify mock was called correctly.</td>
</tr>
</tbody>
</table>
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