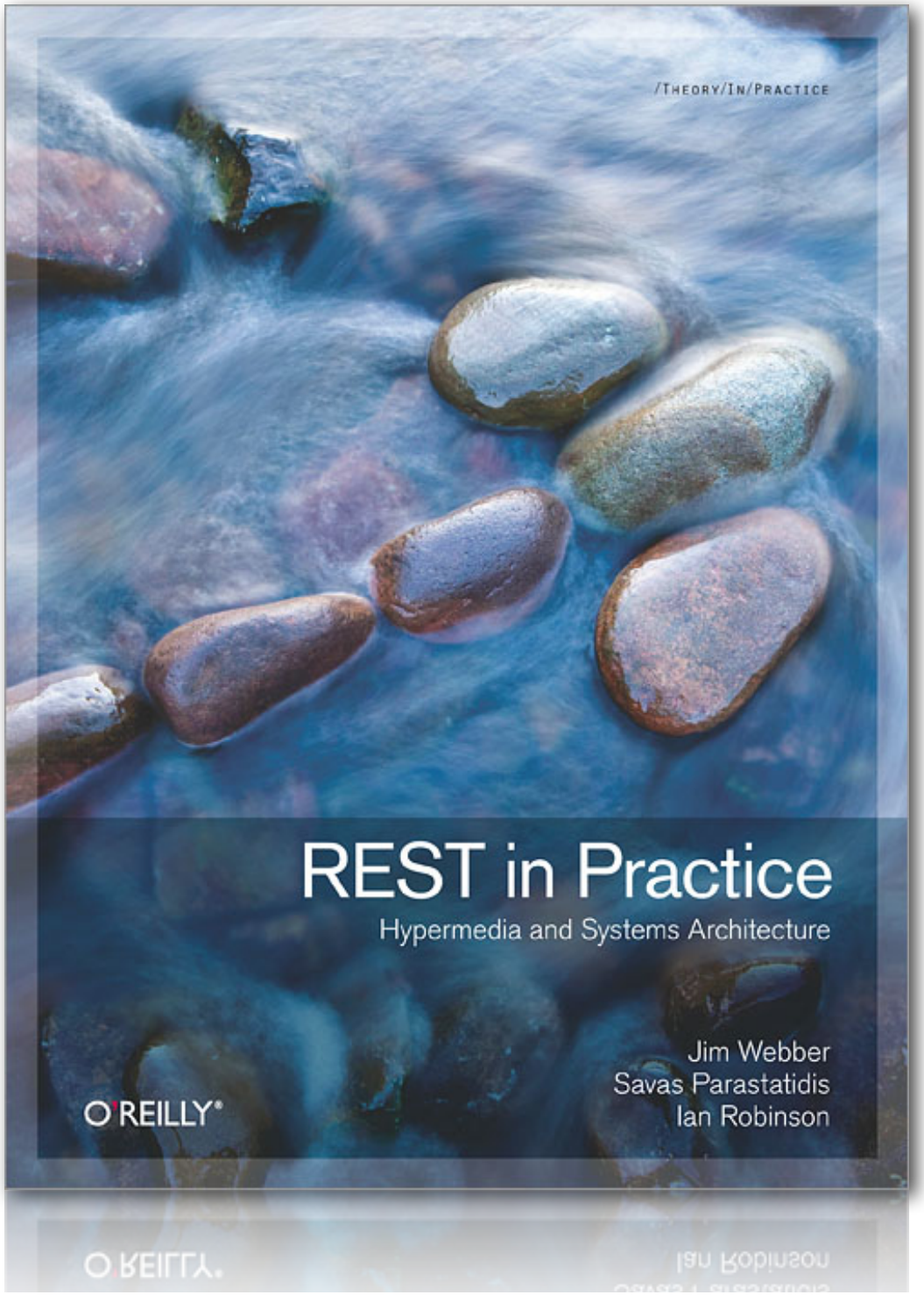
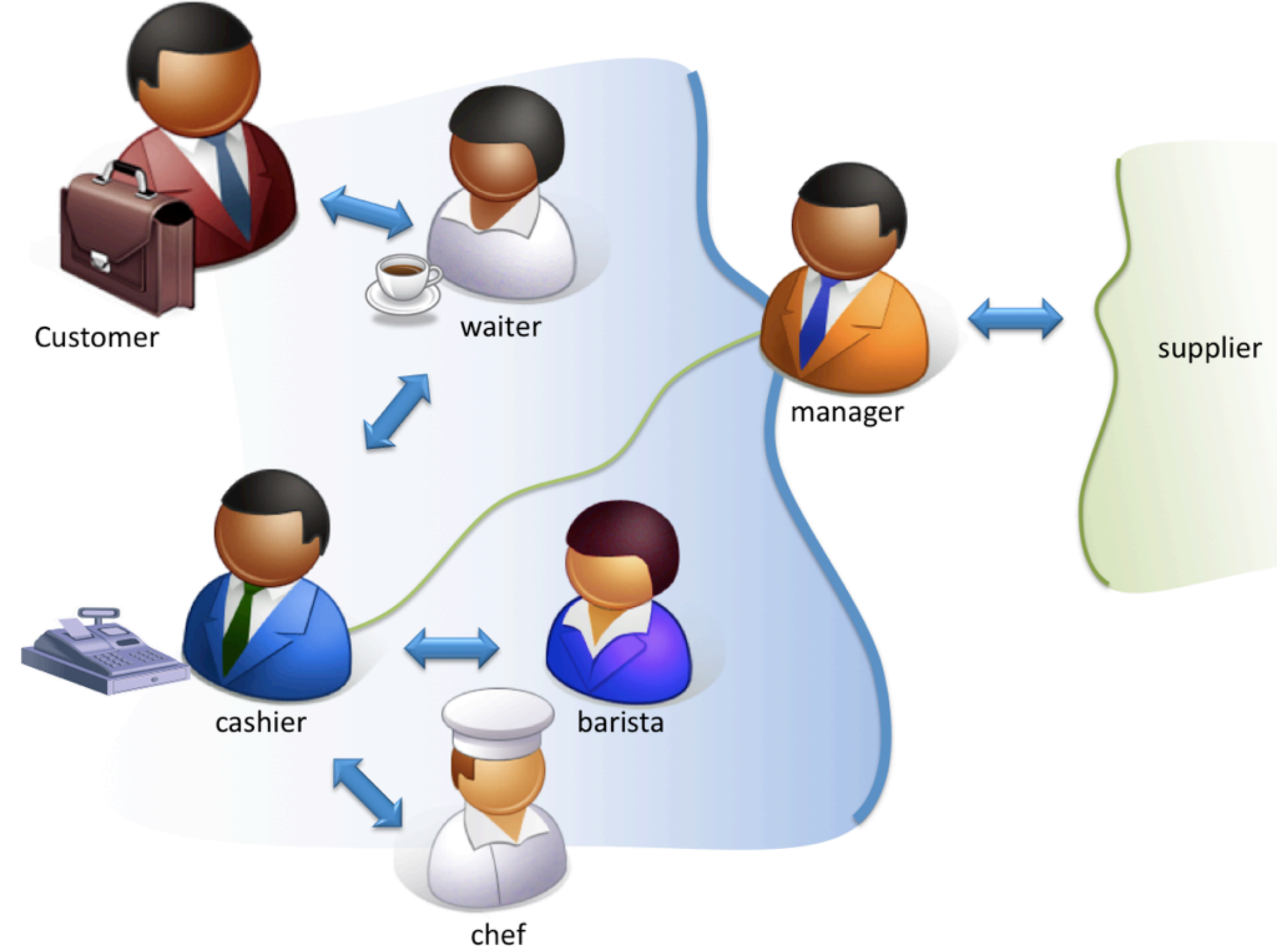
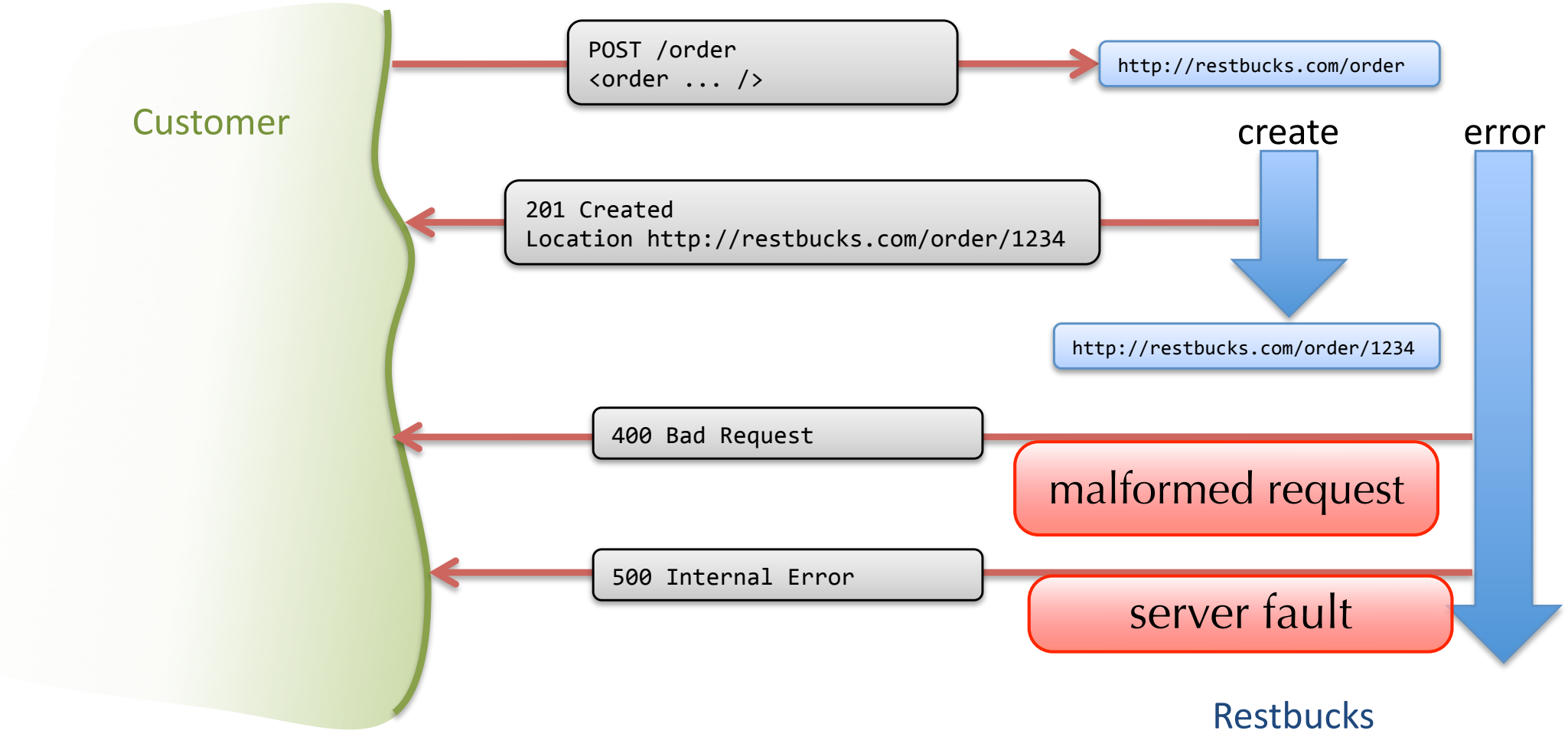
web services & messaging

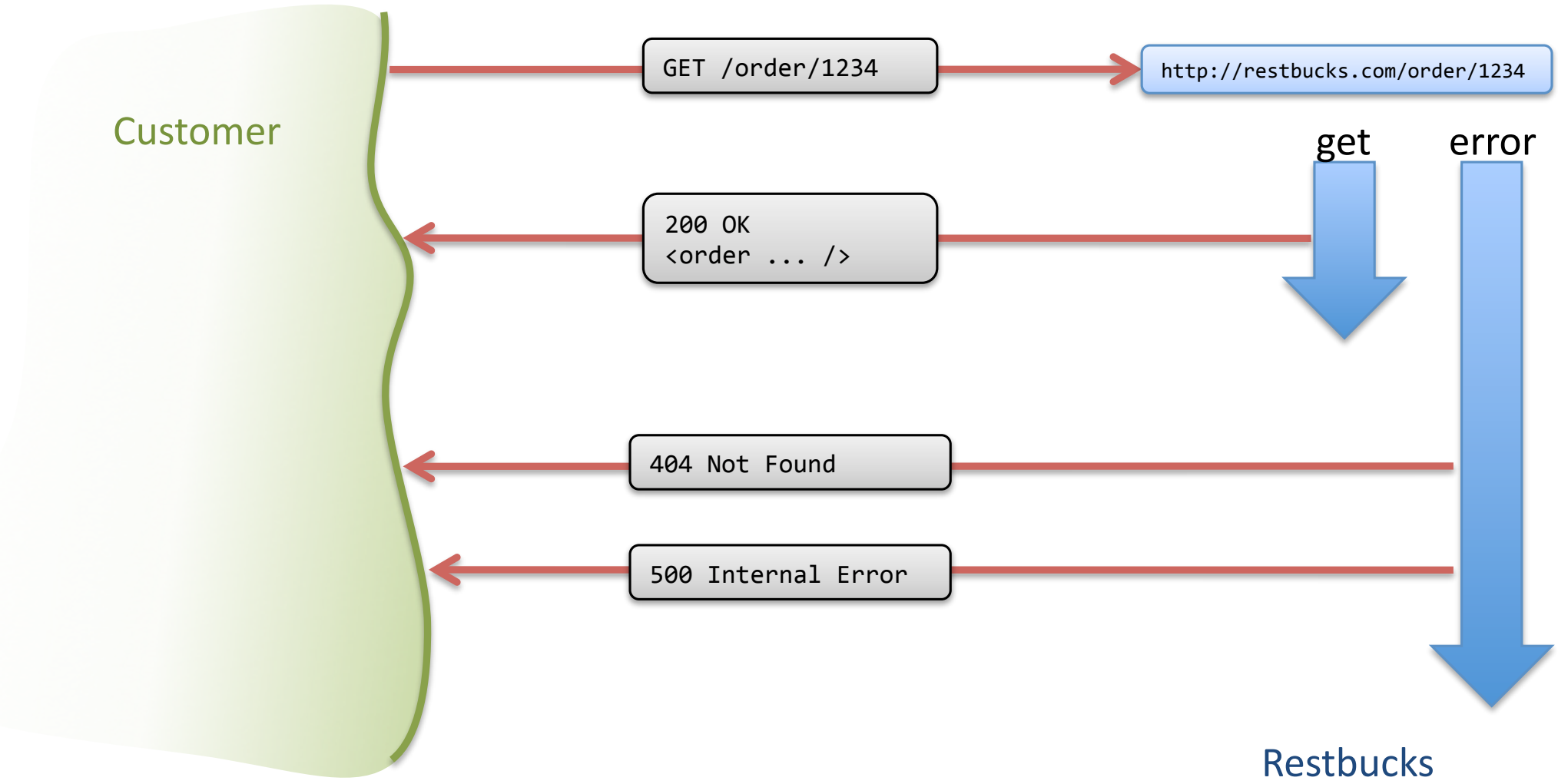

/Theory/In/Practice
REST in Practice
Hypermedia and Systems Architecture
Jim Webber
Savas Parastatidis
Ian Robinson
O'REILLY®


Customer
waiter
manager
supplier
cashier
barista
chef

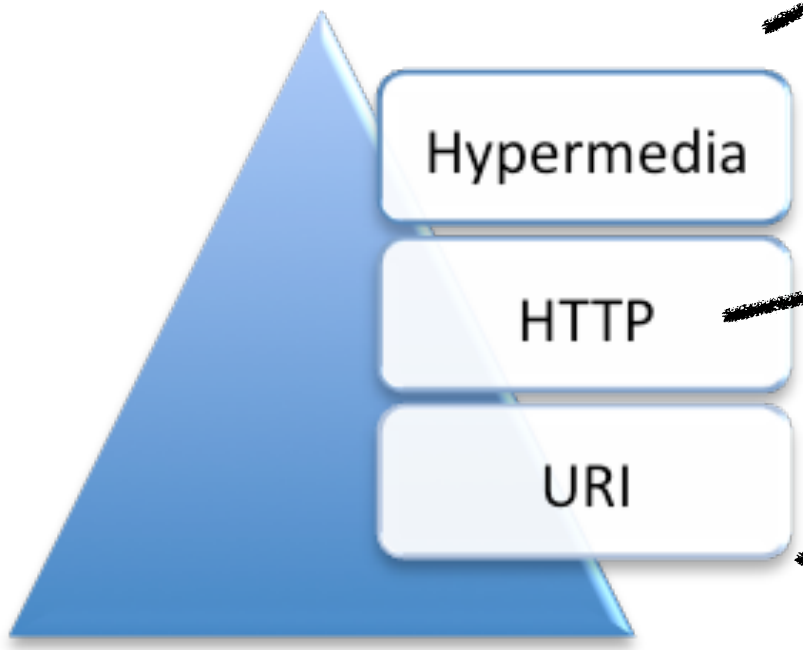
creating an order via POST



reading via GET



Richardson's restful levels of maturity



level 3:
HATEOAS

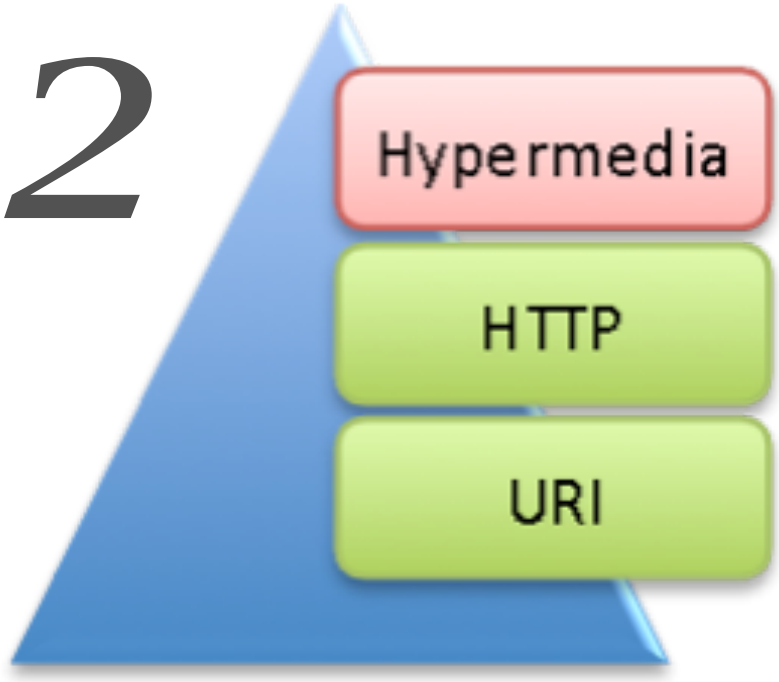
level 2:
numerous URI
addressable
resources

level 1:
many URIs
single HTTP verb

level 0:
single URI
single HTTP verb

CRUD services

2



embraces both HTTP & URIs

viable, robust, and easily implemented solution for some problem domains

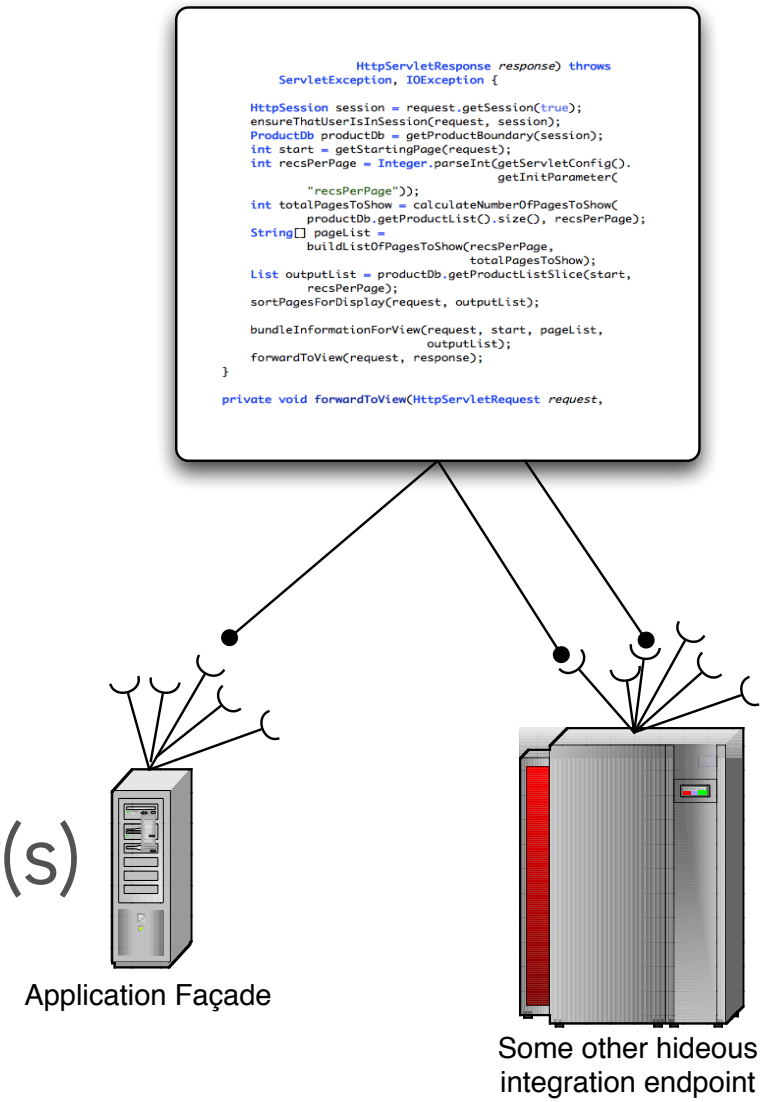
CRUD is best for...CRUD

shared, tightly coupled, understanding of resource life cycles

fundamental problem

client

server(s)

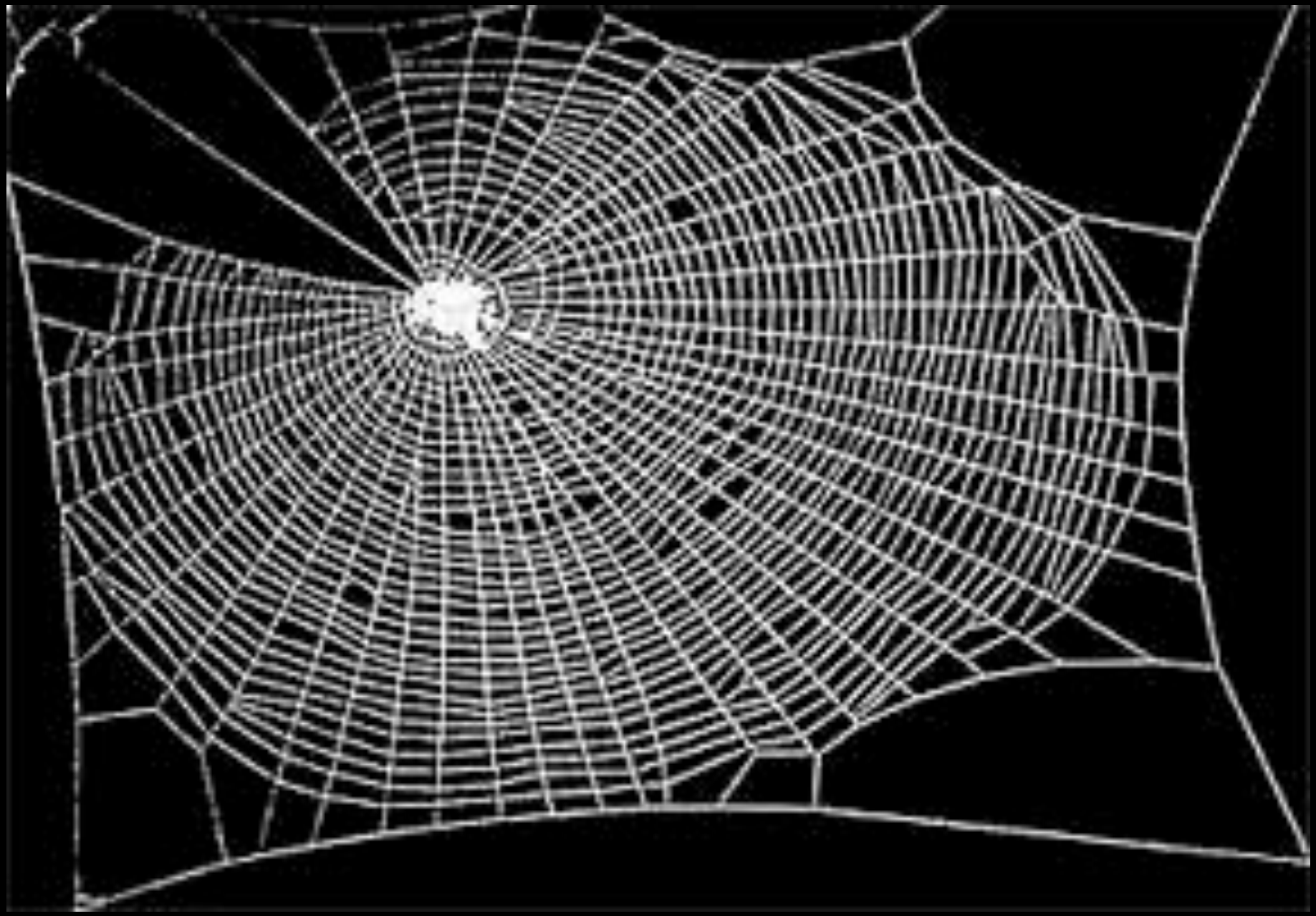


fine-grain workflow driven by client

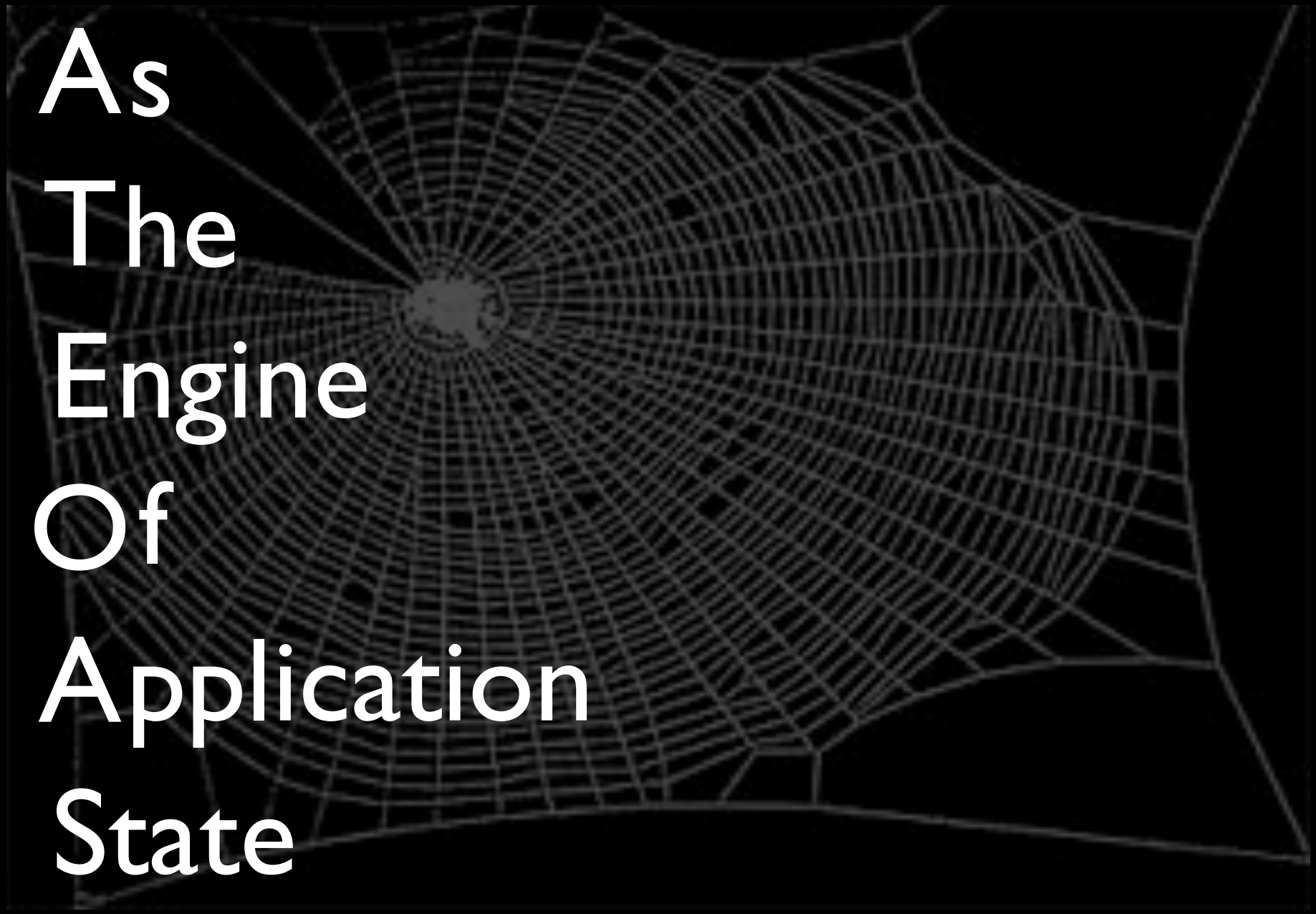
couples client & server together forever

applications own their business logic but not their workflow?!?

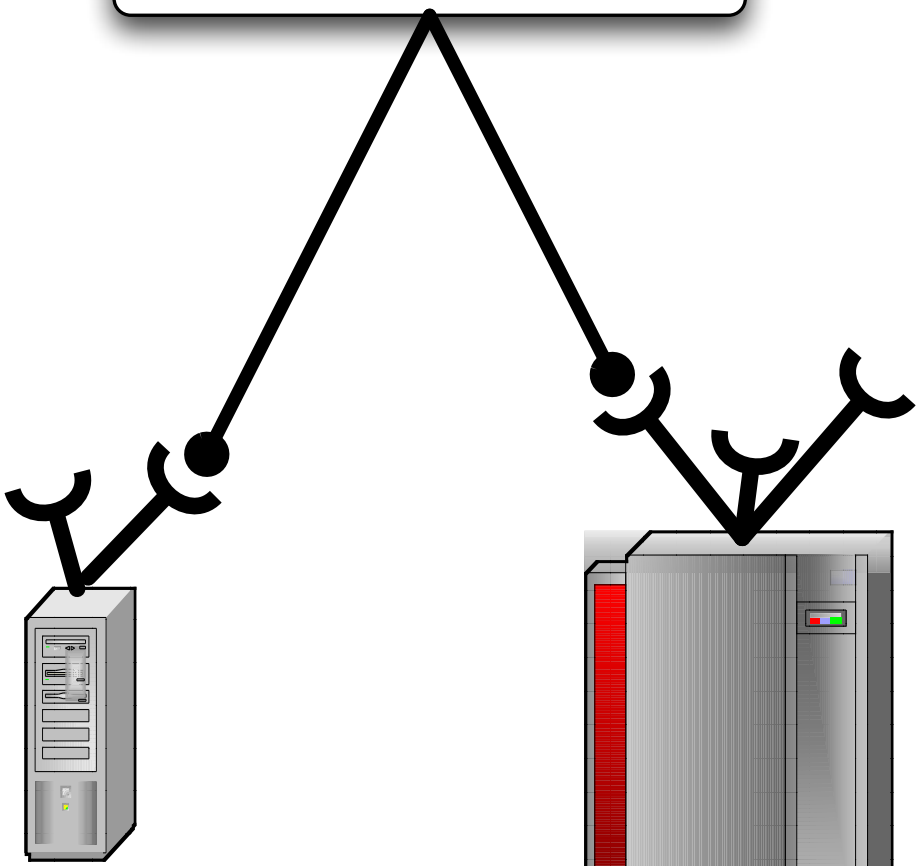
H A T E O A S



Hypermedia As The Engine Of Application State



```
                    HttpServletResponse response) throws
            ServletException, IOException {

        HttpSession session = request.getSession(true);
        ensureThatUserIsInSession(request, session);
        ProductDb productDb = getProductBoundary(session);
        int start = getStartingPage(request);
        int recsPerPage = Integer.parseInt(getServletConfig().
                                           getInitParameter(
                "recsPerPage"));
        int totalPagesToShow = calculateNumberOfPagesToShow(
                productDb.getProductList().size(), recsPerPage);
        String[] pageList =
                buildListOfPagesToShow(recsPerPage,
                                       totalPagesToShow);
        List outputList = productDb.getProductListSlice(start,
                recsPerPage);
        sortPagesForDisplay(request, outputList);

        bundleInformationForView(request, start, pageList,
                                 outputList);
        forwardToView(request, response);
    }

    private void forwardToView(HttpServletRequest request,
```



Application Façade

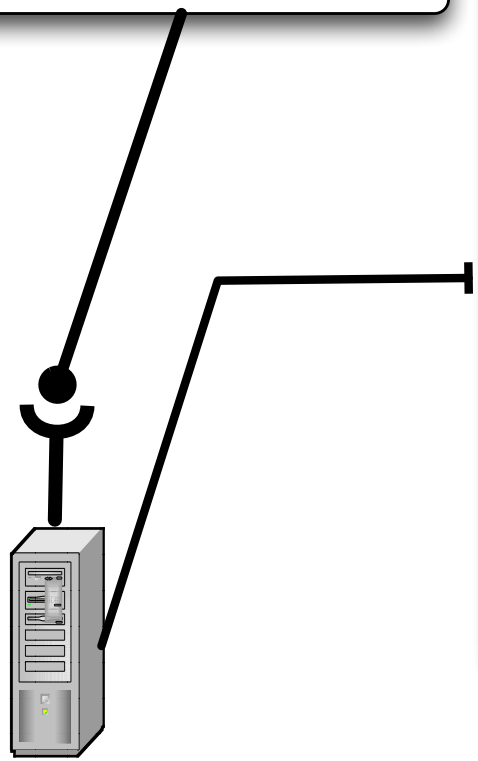
Some other hideous integration endpoint

processes, not methods

HATEOAS

```
                    HttpServletResponse response) throws
            ServletException, IOException {

        HttpSession session = request.getSession(true);
        ensureThatUserIsInSession(request, session);
        ProductDb productDb = getProductBoundary(session);
        int start = getStartingPage(request);
        int recsPerPage = Integer.parseInt(getServletConfig().
                                           getInitParameter(
                "recsPerPage"));
        int totalPagesToShow = calculateNumberOfPagesToShow(
                productDb.getProductList().size(), recsPerPage);
        String[] pageList =
                buildListOfPagesToShow(recsPerPage,
                                       totalPagesToShow);
        List outputList = productDb.getProductListSlice(start,
                recsPerPage);
        sortPagesForDisplay(request, outputList);

        bundleInformationForView(request, start, pageList,
                                 outputList);
        forwardToView(request, response);
    }

    private void forwardToView(HttpServletRequest request,
```

```
<context-param>
  <param-name>user</param-name>
  <param-value>root</param-value>
</context-param>
<servlet>
  <servlet-name>catalog</servlet-name>
  <servlet-class>com.nealford.art.memento.emotherearth.controller.Catalog</servlet-class>
  <init-param>
    <param-name>recsPerPage</param-name>
    <param-value>6</param-value>
  </init-param>
</servlet>
<servlet>
  <servlet-name>checkout</servlet-name>
  <servlet-class>com.nealford.art.memento.emotherearth.controller.Checkout</servlet-class>
</servlet>
<servlet>
  <servlet-name>showcart</servlet-name>
  <servlet-class>com.nealford.art.memento.emotherearth.controller.ShowCart</servlet-class>
</servlet>
<servlet>
  <servlet-name>welcome</servlet-name>
  <servlet-class>com.nealford.art.memento.emotherearth.controller.Welcome</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>catalog</servlet-name>
  <url-pattern>/catalog</url-pattern>
</servlet-mapping>
<servlet-mapping>
```

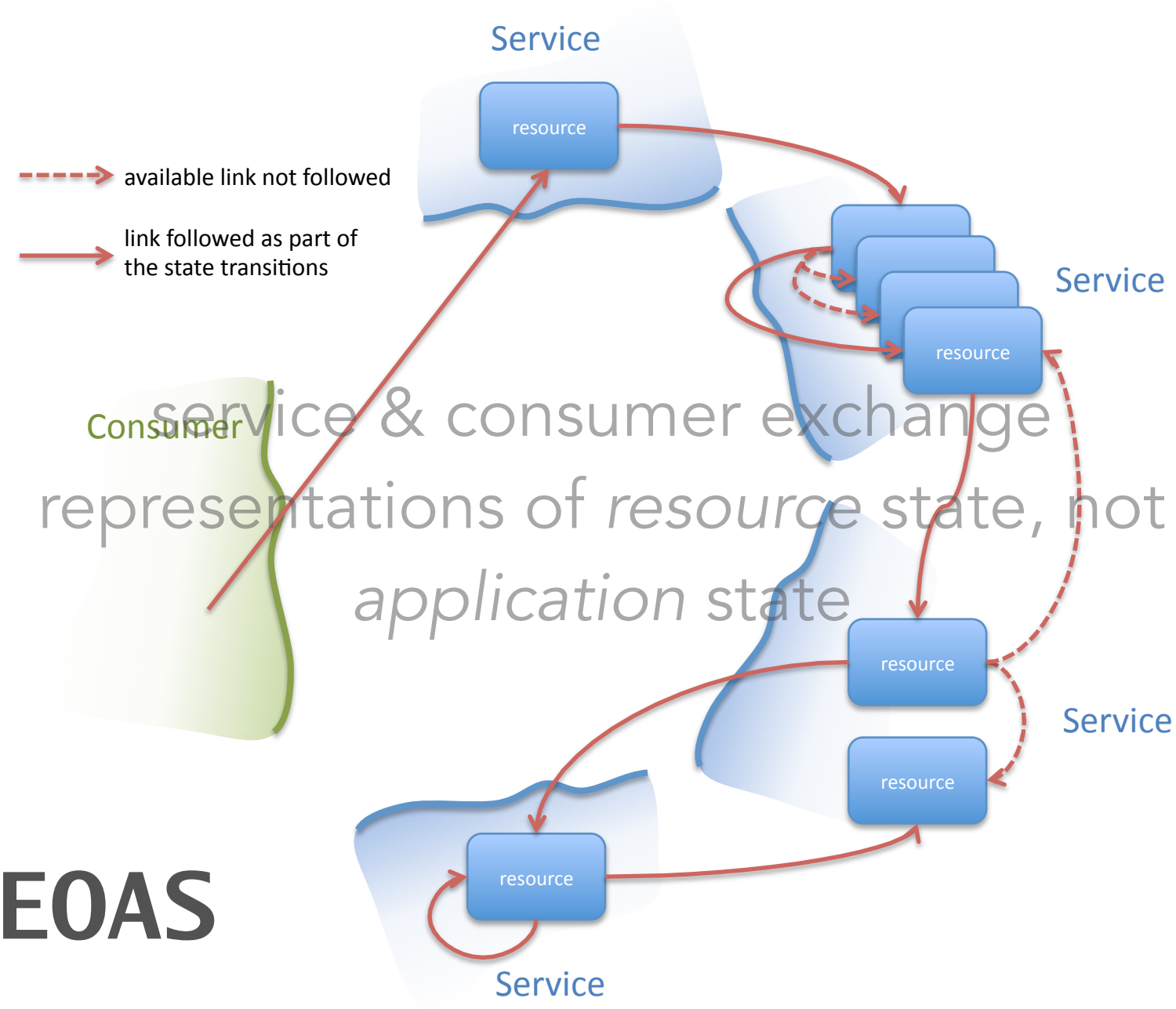


Application Façade

data

process links

the hypermedia tenet



HATEOAS

hypermedia

A service enforces a protocol by advertising legitimate interactions with relevant resources

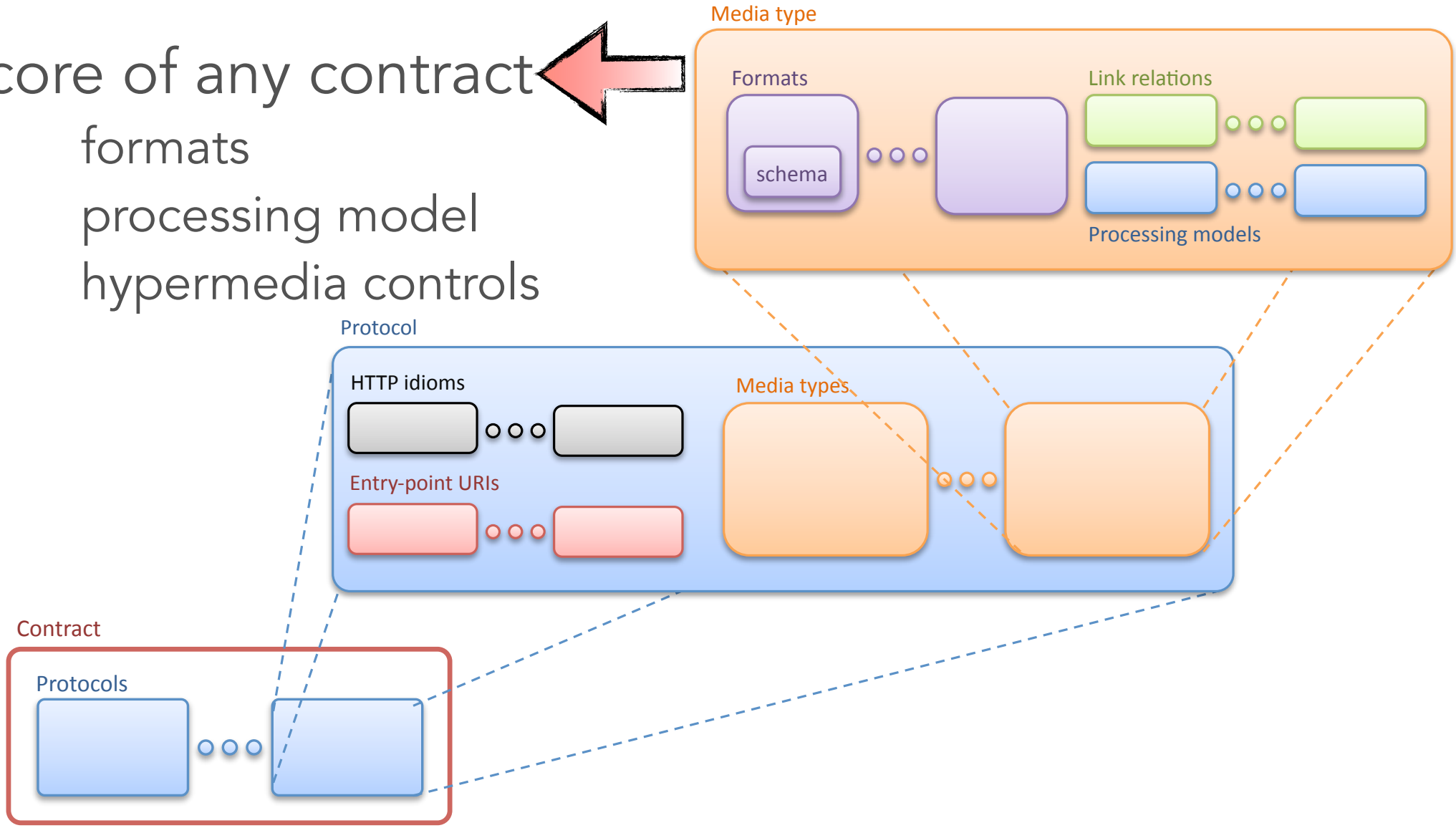
reduces coupling between services and consumers

services can evolve without worrying about consumers

contracts

core of any contract

- formats
- processing model
- hypermedia controls



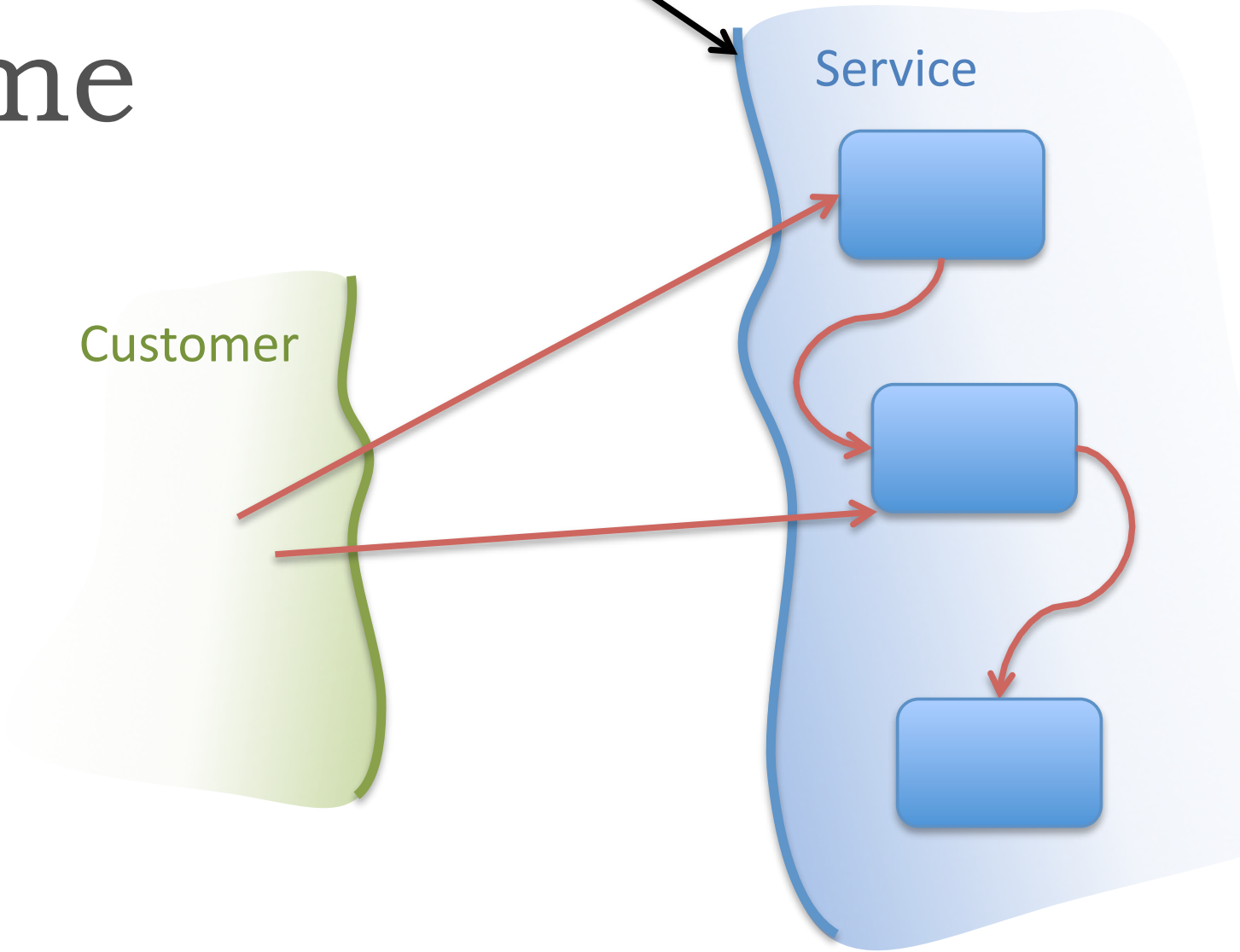
domain specific hypermedia format

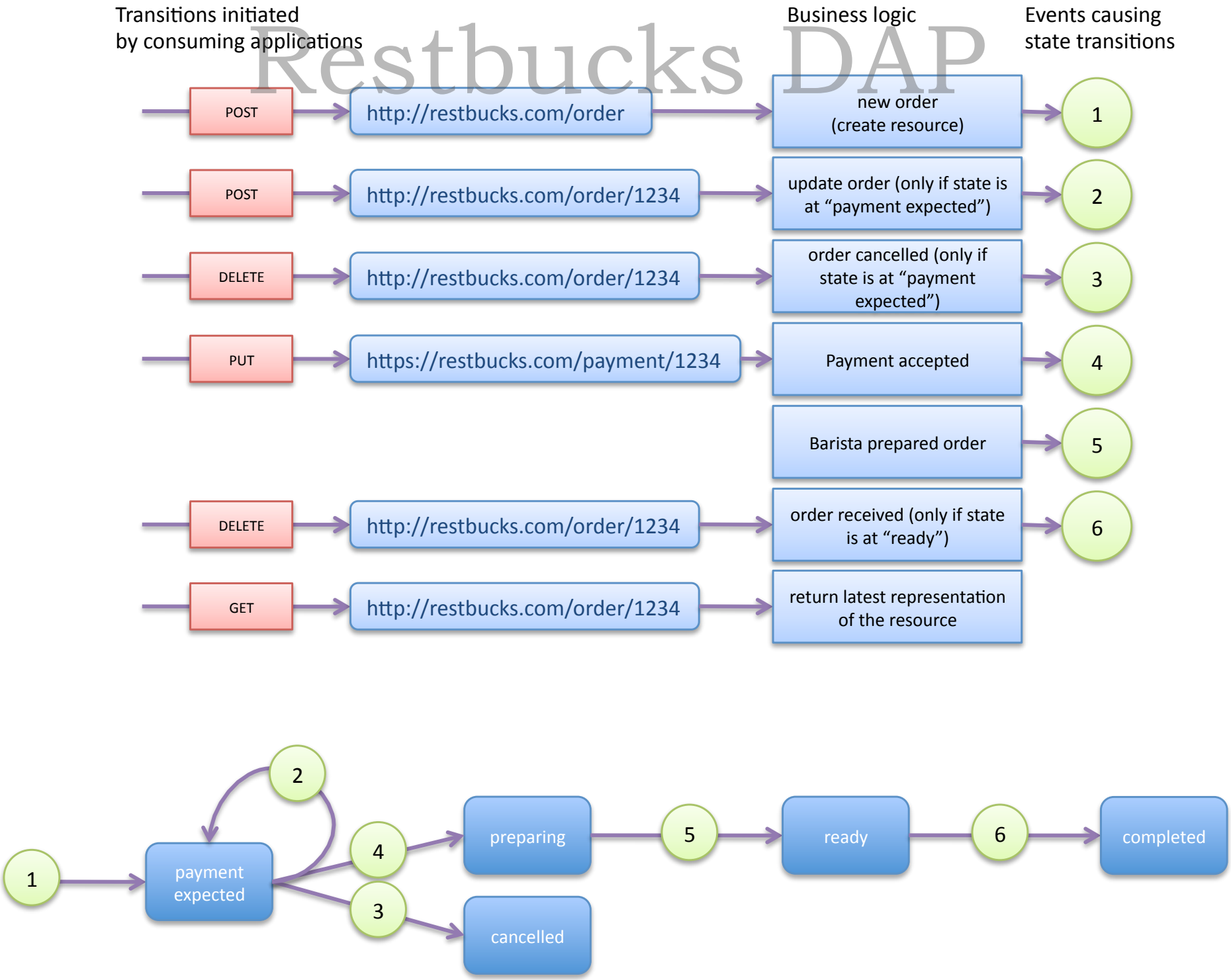
```
<order xmlns="http://schemas.restbucks.com">
  <location>takeAway</location>
  <item>
    <name>latte</name>
    <quantity>1</quantity>
    <milk>whole</milk>
    <size>small</size>
  </item>
  <cost>2.0</cost>
  <status>payment-expected</status>
  <link rel="payment" href="https://restbucks.com/payment/1234" />
</order>
```

enforcing contracts at runtime

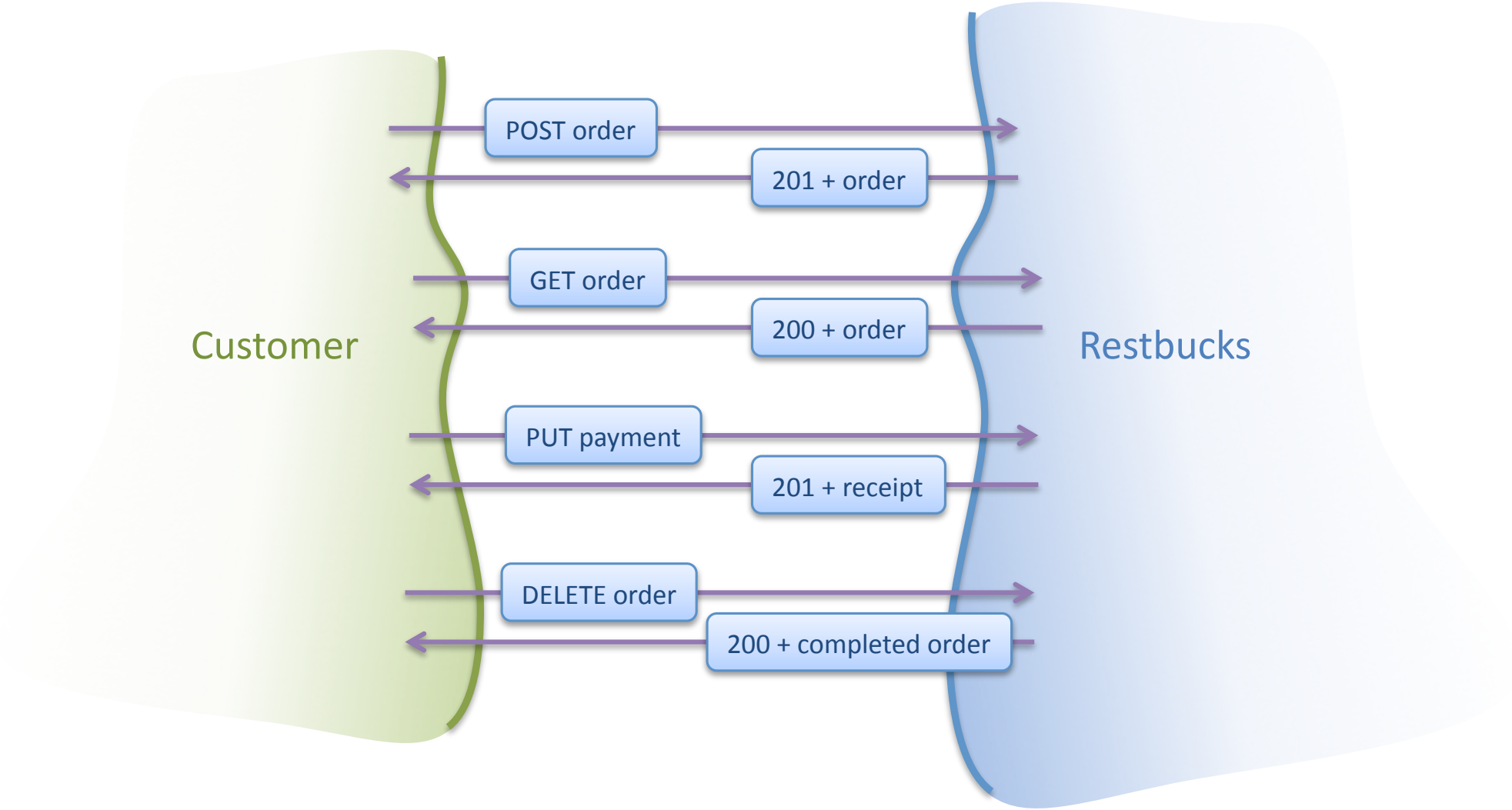
Contract:

- One or more media types
- Protocol(s)
- Any additional link relation values
- HTTP communications idioms
- Entry point URI(s)

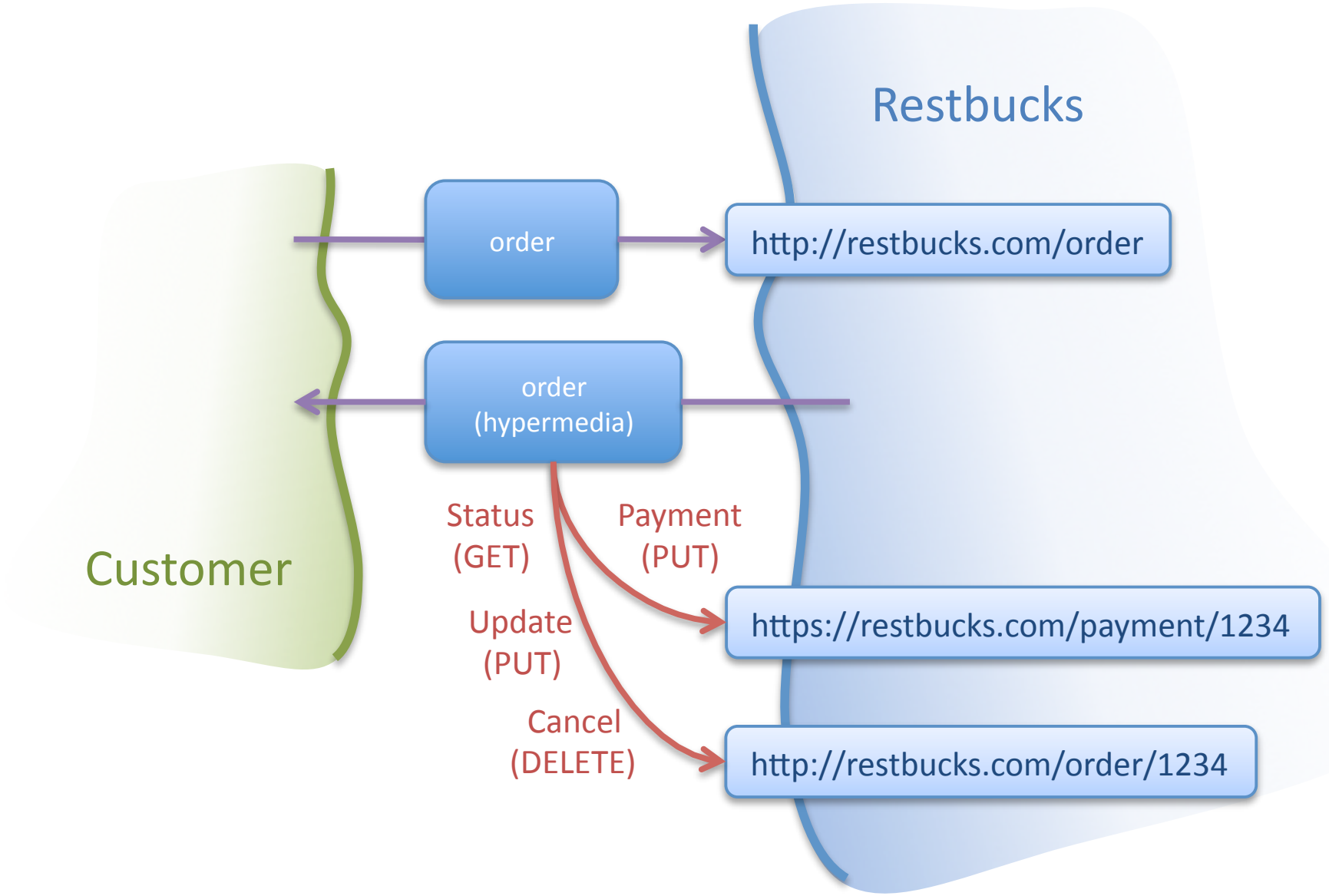




Restbucks ordering protocol



links define interactions



hypermedia order

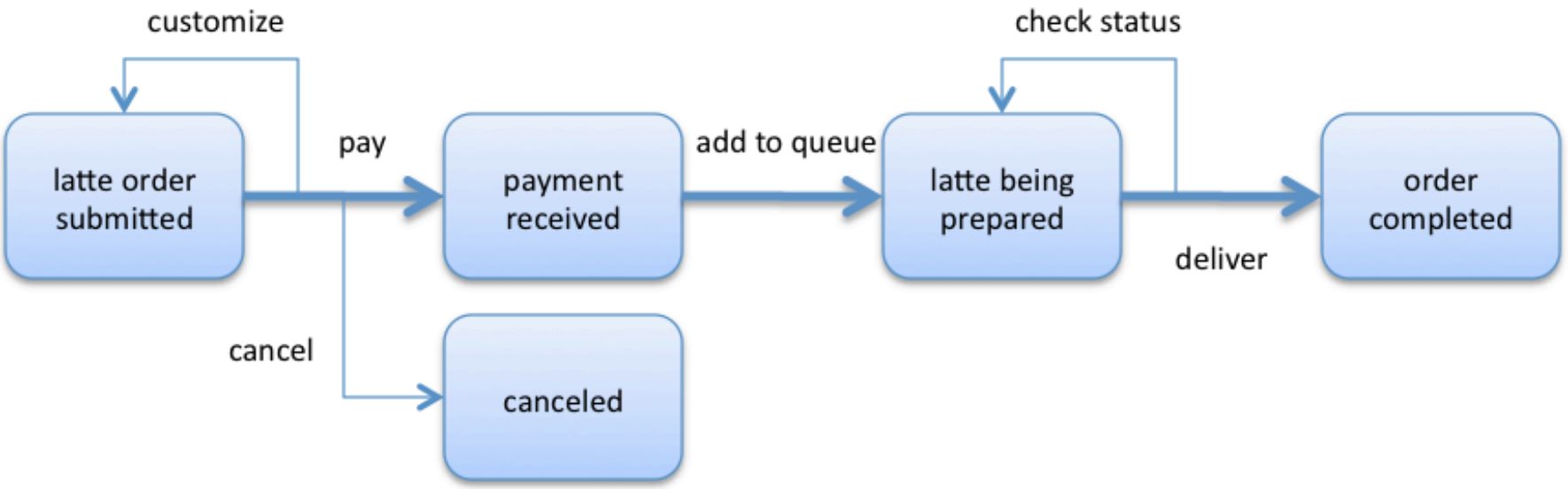
```
<order xmlns="http://schemas.restbucks.com"
  xmlns:dap="http://schemas.restbucks.com/dap">
  <location>takeAway</location>
  <item>
    <name>latte</name>
    <quantity>1</quantity>
    <milk>whole</milk>
    <size>small</size>
  </item>
  <cost>2.0</cost>
  <status>unpaid</status>

  <!-- Restbucks Domain Application Protocol -->
  <dap:link rel="payment" uri="https://restbucks.com/payment/1234"
            mediaType="application/vnd.restbucks+xml"/>
  <dap:link rel="latest"  uri="http://restbucks.com/order/1234"/>
  <dap:link rel="update"  uri="http://restbucks.com/order/1234"
            mediaType="application/vnd.restbucks+xml"/>
  <dap:link rel="cancel"  uri="http://restbucks.com/order/1234"/>
</order>
```

semantic markup

interaction resource

payload format

modeling states

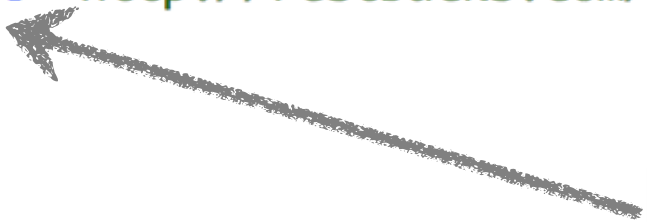


after submitting payment

```
<payment xmlns:dap="http://schemas.restbucks.com/dap"
  xmlns="http://schemas.restbucks.com">
  <amount>2.0</amount>
  <cardholderName>Michael Farraday</cardholderName>
  <cardNumber>123456789012</cardNumber>
  <expiryMonth>12</expiryMonth>
  <expiryYear>12</expiryYear>
  <dap:link uri="http://restbucks.com/order/1234" rel="latest"
            mediaType="application/vnd.restbucks+xml"/>
  <dap:link uri="http://restbucks.com/receipt/1234" rel="receipt"
            mediaType="application/vnd.restbucks+xml"/>
</payment>
```

payment received

```
<receipt xmlns:dap="http://schemas.restbucks.com/dap"
  xmlns="http://schemas.restbucks.com">
  <amount>2.0</amount>
  <paid>2009-09-15T16:49:16.234+12:00</paid>
  <dap:link uri="http://restbucks.com/order/1234" rel="order"/>
</receipt>
```



only 1 place to go

updated order

```
<order xmlns="http://schemas.restbucks.com"
  xmlns:dap="http://schemas.restbucks.com/dap">
  <item>
    <milk>semi</milk>
    <size>large</size>
    <drink>cappuccino</drink>
  </item>
  <location>takeaway</location>
  <cost>2.0</cost>
  <status>preparing</status>
  <dap:link uri="http://restbucks.com/order/1234" rel="latest"
            mediaType="application/vnd.restbucks+xml"/>
</order>
```

completed order

```
<order xmlns="http://schemas.restbucks.com"
  xmlns:dap="http://schemas.restbucks.com/dap">
  <item>
    <milk>semi</milk>
    <size>large</size>
    <drink>cappuccino</drink>
  </item>
  <location>takeaway</location>
  <cost>2.0</cost>
  <status>ready</status>
  <dap:link uri="http://restbucks.com/order/1234" rel="latest"
            mediaType="application/vnd.restbucks+xml"/>
</order>
```

after order is deleted from queue

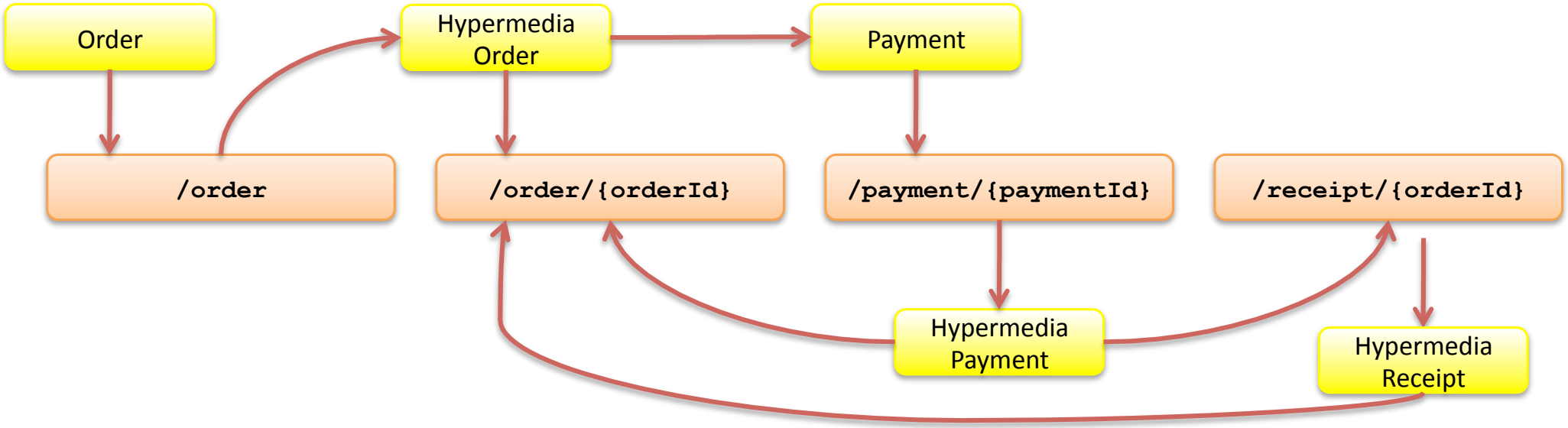
```
<order xmlns="http://schemas.restbucks.com/order"
       xmlns:dap="http://schemas.restbucks.com/dap">
  <location>takeAway</location>
  <item>
    <name>latte</name>
    <quantity>1</quantity>
    <milk>whole</milk>
    <size>small</size>
  </item>
```

no further application workflow

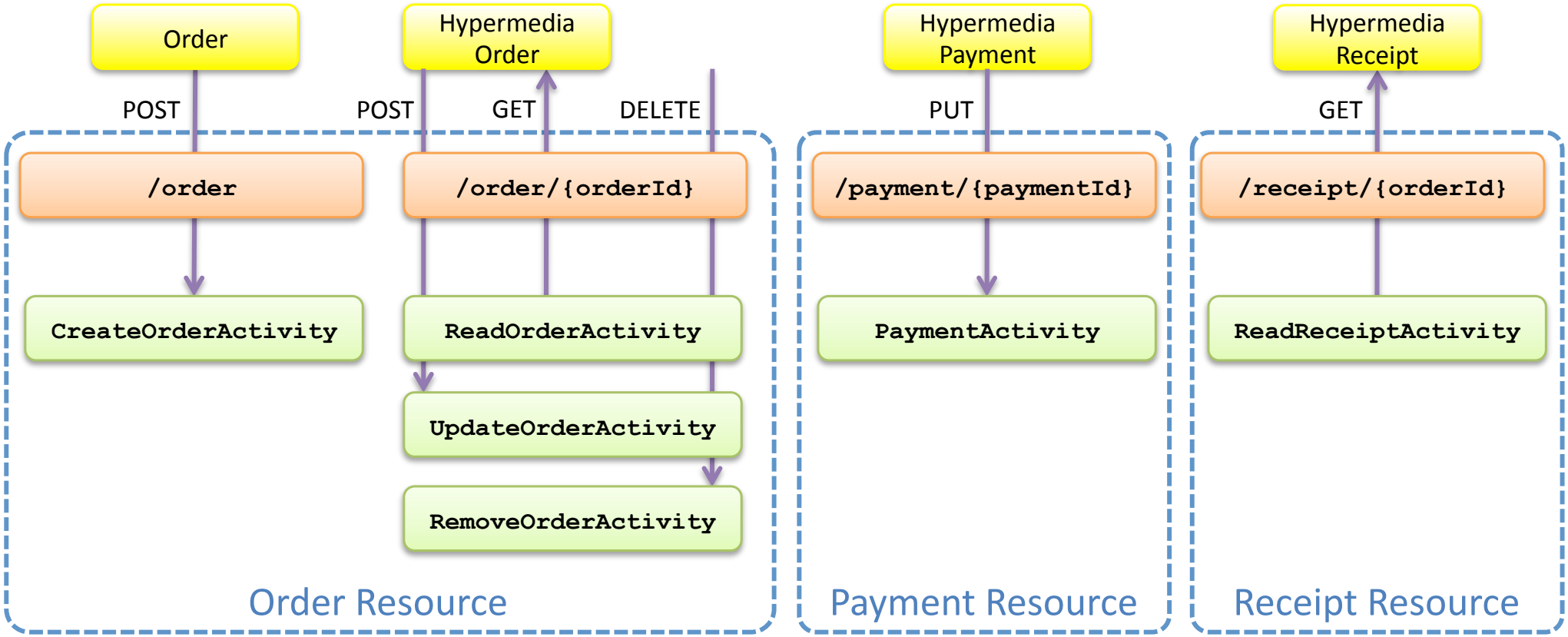
dynamically extending DAP

```
<order xmlns="http://schemas.restbucks.com "
  xmlns:dap="http://schemas.restbucks.com/dap">
  <location>takeAway</location>
  <item>
    <name>latte</name>
    <quantity>1</quantity>
    <milk>whole</milk>
    <size>small</size>
  </item>
  <cost>2.0</cost>
  <status>preparing</status>
  <dap:link rel="status" uri="http://restbucks.com/order/1234"
            mediaType="application/vnd.restbucks+xml"/>
  <dap:link rel="coffee-card" uri="http://restbucks.com/order/1234/coffeecard"
            mediaType="application/vnd.restbucks+xml"/>
</order>
```

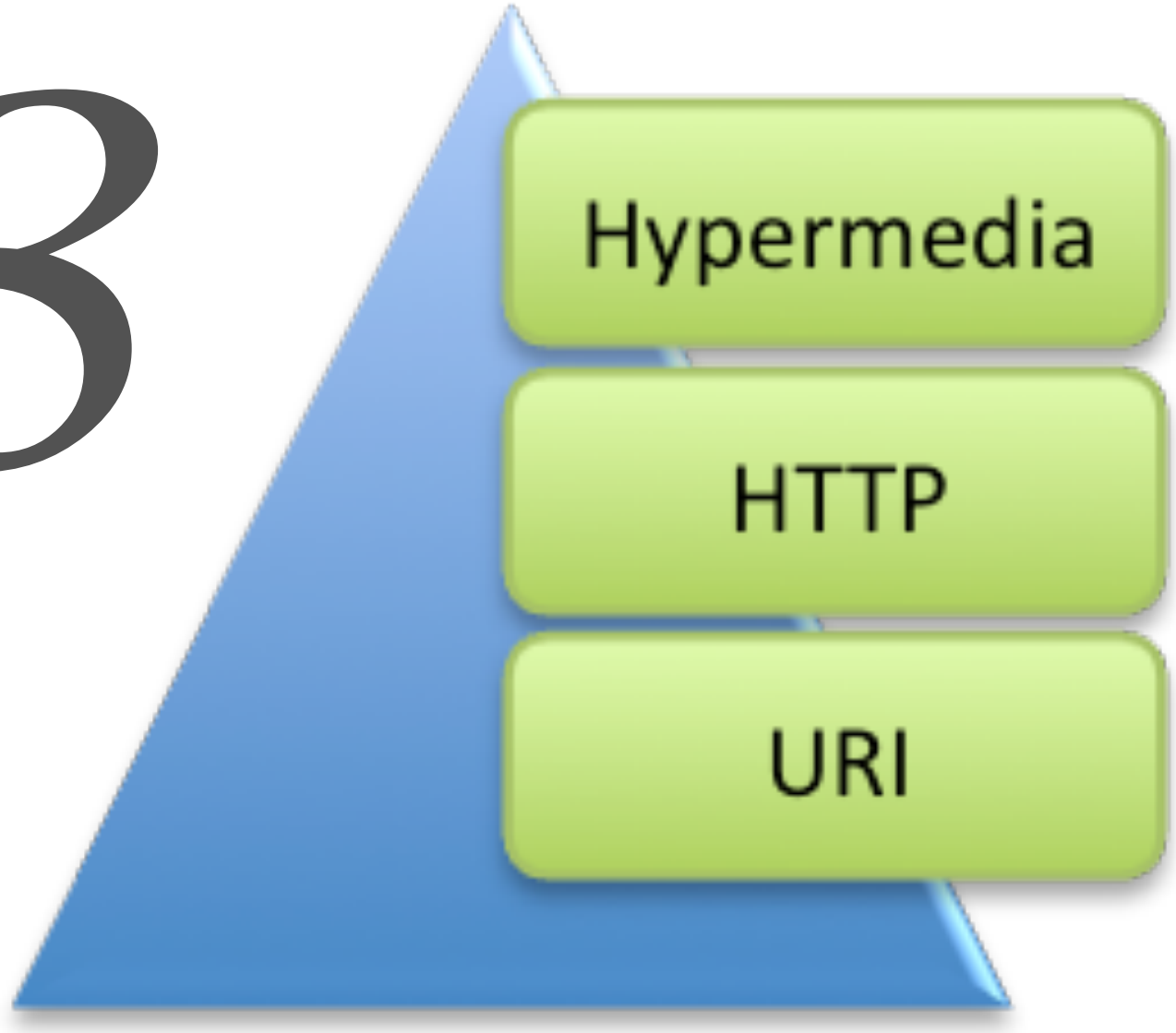
Hypermedia resources describe the ordering and payment protocol to consumers



mapping to existing application architecture

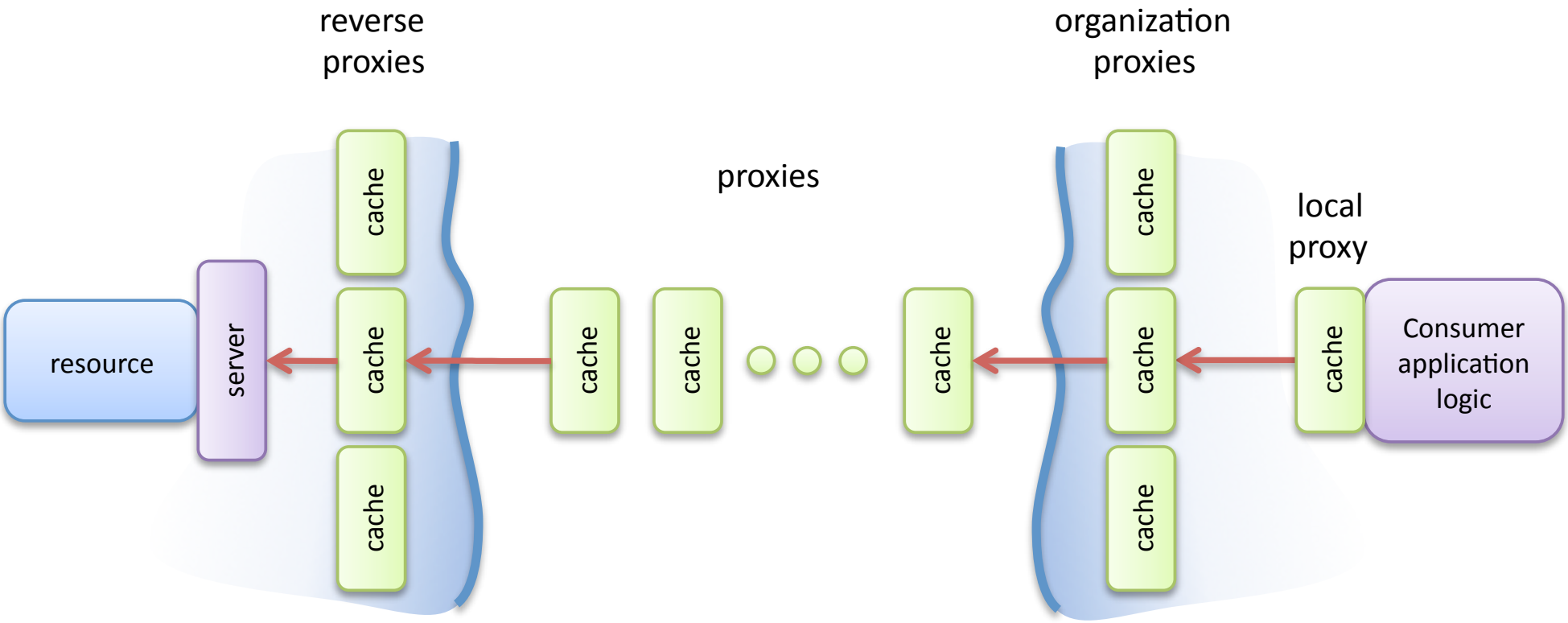


3



Scaling

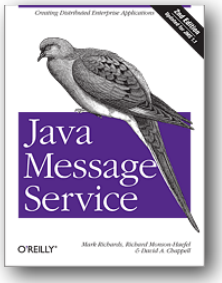
the web is built to be cached



Cache-Control & Expires

```
HTTP/1.1 200 OK
Cache-Control: public, max-age=604800
Content-Type: application/xml
Server: Mono-HTTPAPI/1.0
Date: Sun, 27 Dec 2009 01:30:51 GMT
Content-Length: ...
...
<!-- Content ommitted -->

Cache-Control: public
Expires: Sun, 10 Jan 2010 08:00:00 GMT
Content-Type: application/xml
Server: Mono-HTTPAPI/1.0
Date: Wed, 30 Dec 2009 06:06:28 GMT
...
```

messaging vs. web services

remote message delivery capability		
asynchronous requests	✓	✗
guaranteed delivery	✓	✗
monitoring capability	✓	✗
exception processing	✓	✗
performance	✓	✗
external firewall access	✗	✓

messaging vs. web services

http://hostname/approot/message/jms/queue1?type=queue

hidden post variables → message header properties

any content type → TextMessage

HTTP GET → receive() (for convenience)

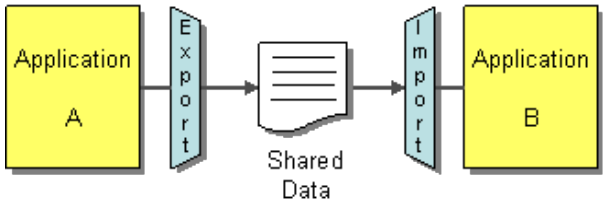
HTTP POST → send()

HTTP DELETE → receive()

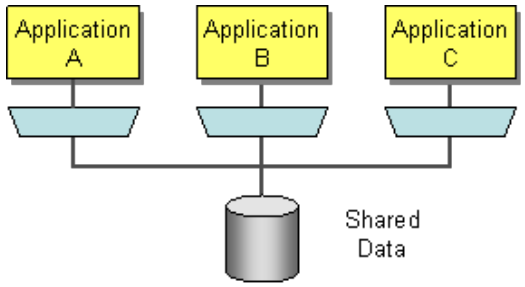
HTTP PUT → no mapping

messaging vs. web services

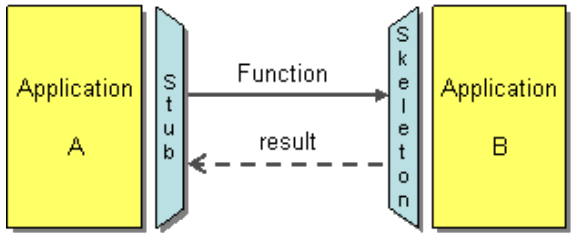
From Enterprise Integration Patterns by Hohpe and Woolf



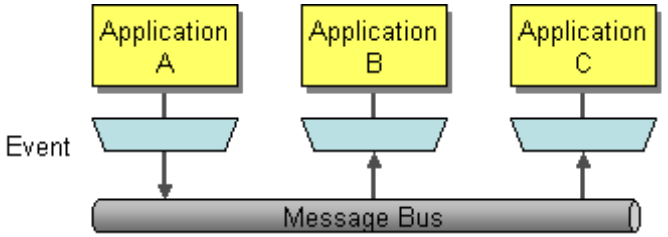
file transfer



shared database



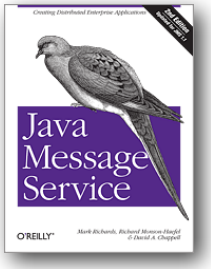
remote procedure invocation



messaging

messaging vs. web services

recommendations



internal (inside the firewall)

external (outside the firewall)

?'S

Mark Richards

Independent Consultant

Hands-on Enterprise / Integration Architect
Published Author / Conference Speaker

http://www.wmrichards.com
http://www.linkedin.com/pub/mark-richards/0/121/5b9

Published Books:

Java Message Service, 2nd Edition
97 Things Every Software Architect Should Know
Java Transaction Design Strategies

Neal Ford

Director / Software Architect /
Meme Wrangler

ThoughtWorks®

2002 Summit Blvd, Level 3, Atlanta, GA 30319, USA
T: +1 40 4242 9929 Twitter: @neal4d
E: nford@thoughtworks.com W: thoughtworks.com