Università della Svizzera italiana

Faculty of Informatics

Jopera
Process Support for Web Services

RESTful Service Composition with JOpera

Cesare Pautasso
Faculty of Informatics, USI Lugano, Switzerland
c.pautasso@ieee.org
http://www.pautasso.info
http://twitter.com/pautasso

21.5.2010

Abstract

Next generation Web services technologies challenge the assumptions made by current standards for process-based service composition. For example, most existing RESTful Web service APIs cannot natively be composed using the WS-BPEL standard. In this talk we apply the notion of composition to RESTful services and discuss the conceptual relationship between business processes and stateful resources. Our goal is to enable lightweight access to service compositions published with a RESTful API.

We show that the uniform interface and the hyper-linking capabilities of RESTful services provide an excellent abstraction for publishing proceses as a resource and exposing in a controlled way the execution state of a service composition.

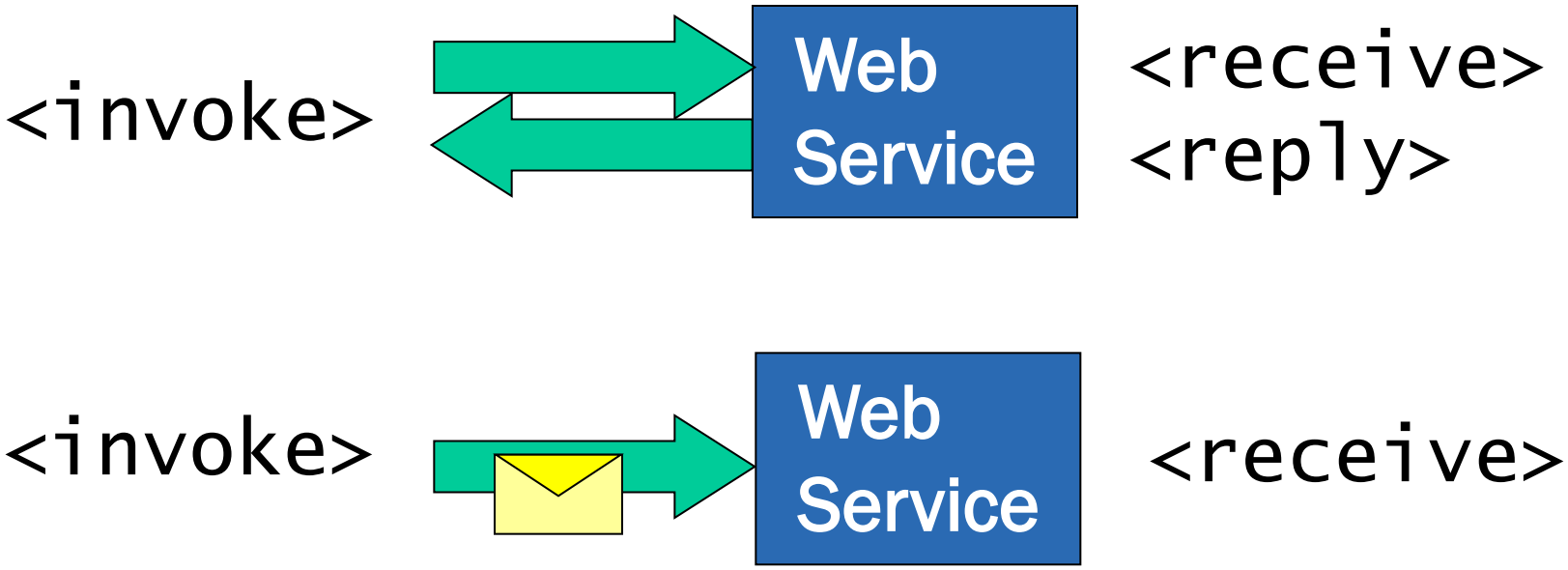
To do so, we present how to build a composite application (DoodleMap) out of some well-known, public and currently existing RESTful service APIs.

From REST-*

> We believe there is huge potential to marrying REST with workflow and BPM.
>
> [...]
>
> Combined with the architecture of the Web, a workflow service can provide both a truly **simple, portable, and flexible** way to build workflow driven integrations and applications.

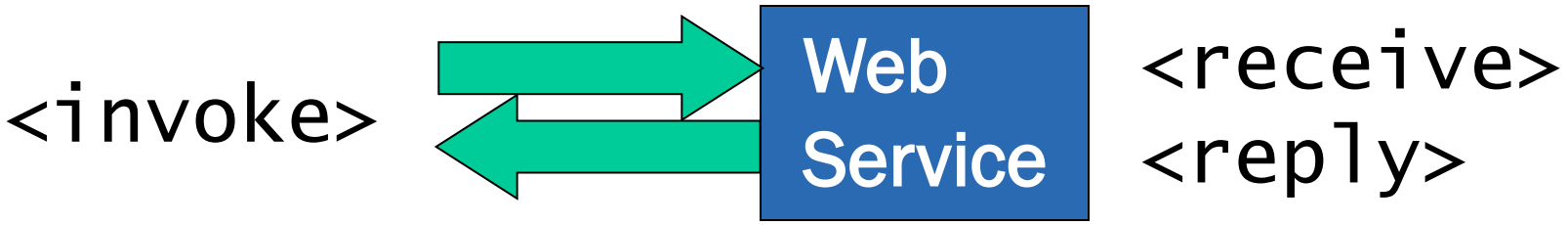
http://www.jboss.org/reststar/specifications/workflow.html

WS-BPEL Primitives



- The workflow language natively supports the RPC or message-based connectors

WS-BPEL Primitives

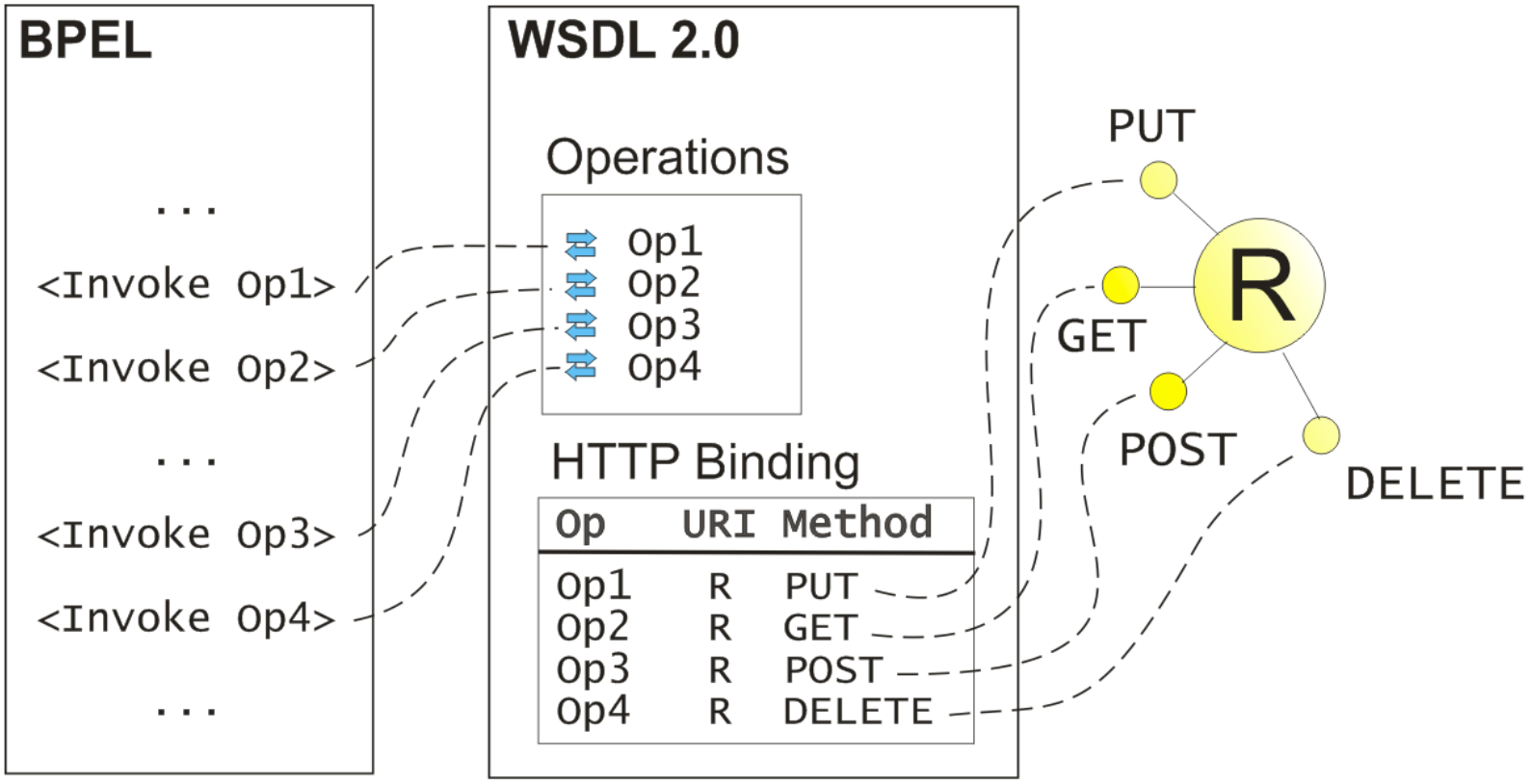


- Easy to map this to HTTP!

BPEL and WSDL 2.0

WSDL 2.0 HTTP Binding can wrap RESTful Web Services

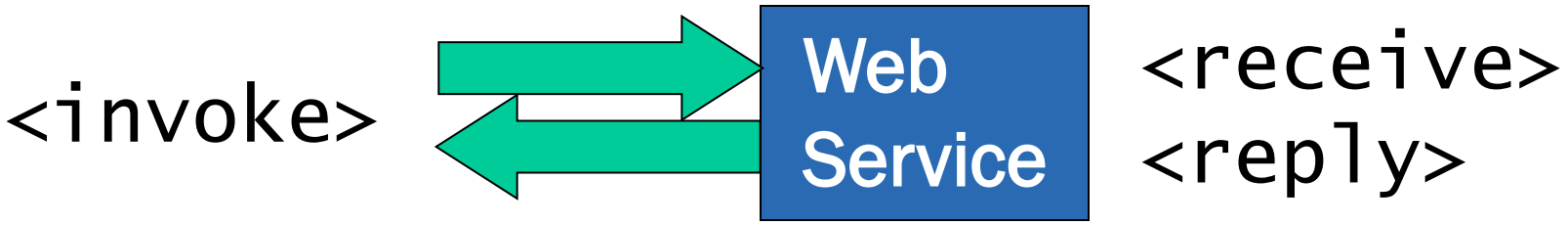
(WS-BPEL 2.0 does not support WSDL 2.0)



RESTful APIs...

...do not use WSDL

WS-BPEL Primitives



- Easy to map this to HTTP?
- We need something else

From REST-*

> “We believe there is huge potential to marrying REST with workflow and BPM.
>
> - The HATEOAS (hypermedia and linking) principal of REST is logically a dynamic state machine and **fits very well** with how workflow and BPM systems are designed.
> - Combined with the architecture of the Web, a workflow service can provide both a truly simple, portable, and flexible way to build workflow driven integrations and applications.”

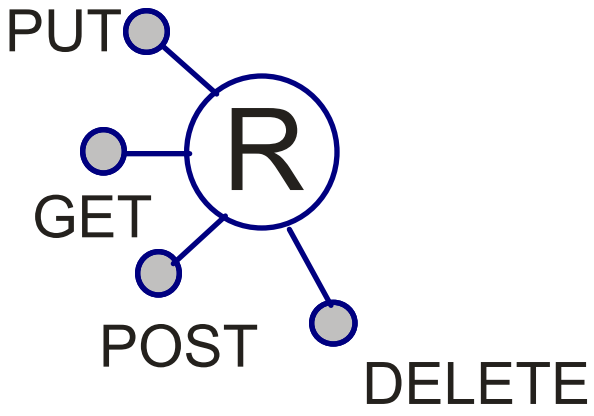
http://www.jboss.org/reststar/specifications/workflow.html

Outline

1. Introduction to RESTful Web Services
2. Defining RESTful Service Composition
3. JOpera Demo
4. More than Mashups?

REST in one slide

- Web Services expose their data and functionality trough **resources** identified by **URI**
- **Uniform Interface** Principle: Clients interact with resources through a fix set of verbs.
  Example HTTP:
  GET (read), POST (create), PUT (update), DELETE



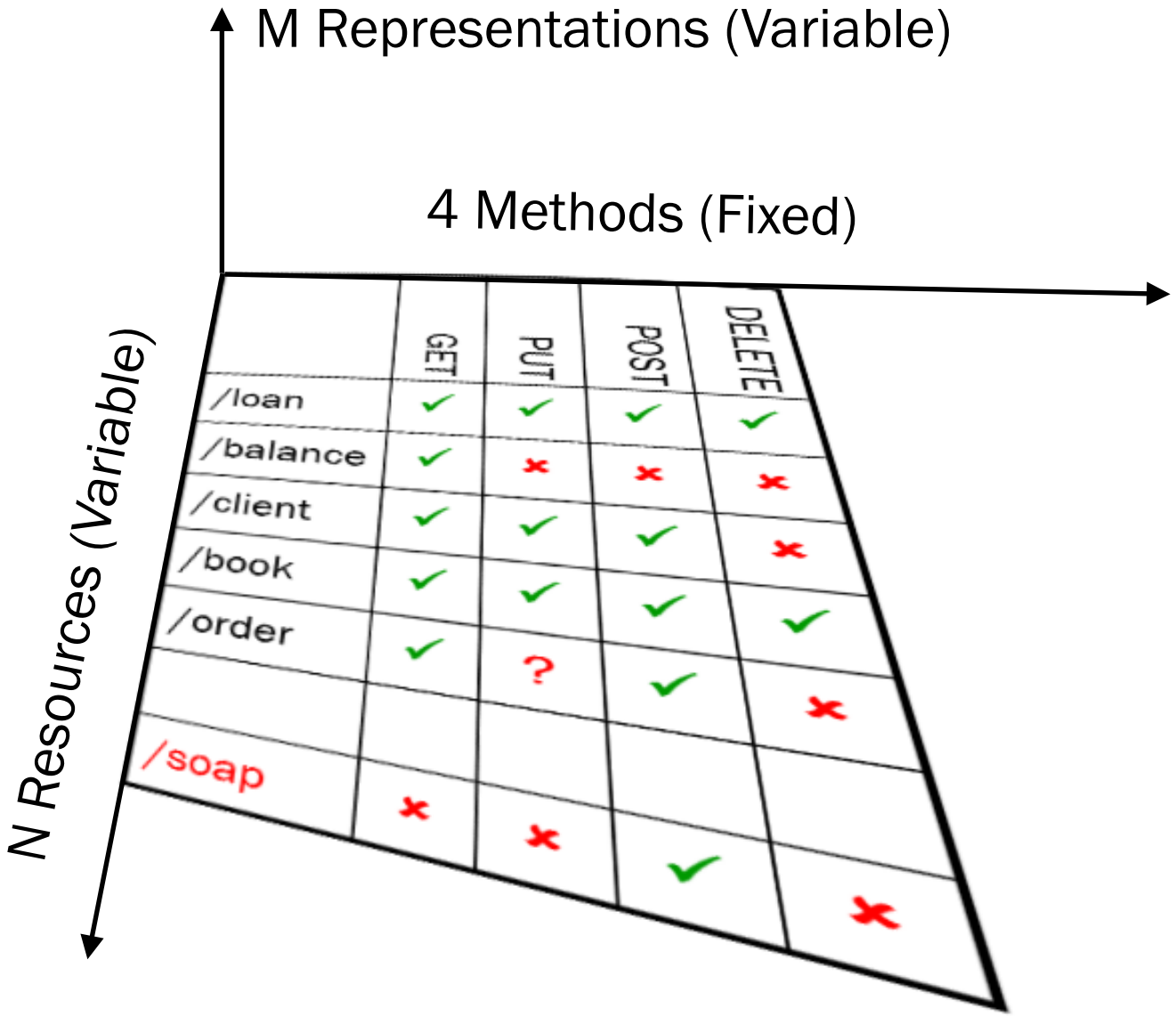
- **Multiple representations** for the same resource
- **Hyperlinks** model resource relationships and valid state transitions for dynamic protocol description and discovery

Design Methodology

1. Identify resources to be exposed as services (e.g., yearly risk report, book catalog, purchase order, open bugs, polls and votes)
2. Model relationships (e.g., containment, reference, state transitions) between resources with hyperlinks that can be followed to get more details (or perform state transitions)
3. Define “nice” URIs to address the resources
4. Understand what it means to do a GET, POST, PUT, DELETE for each resource (and whether it is allowed or not)
5. Design and document resource representations
6. Implement and deploy on Web server
7. Test with a Web browser

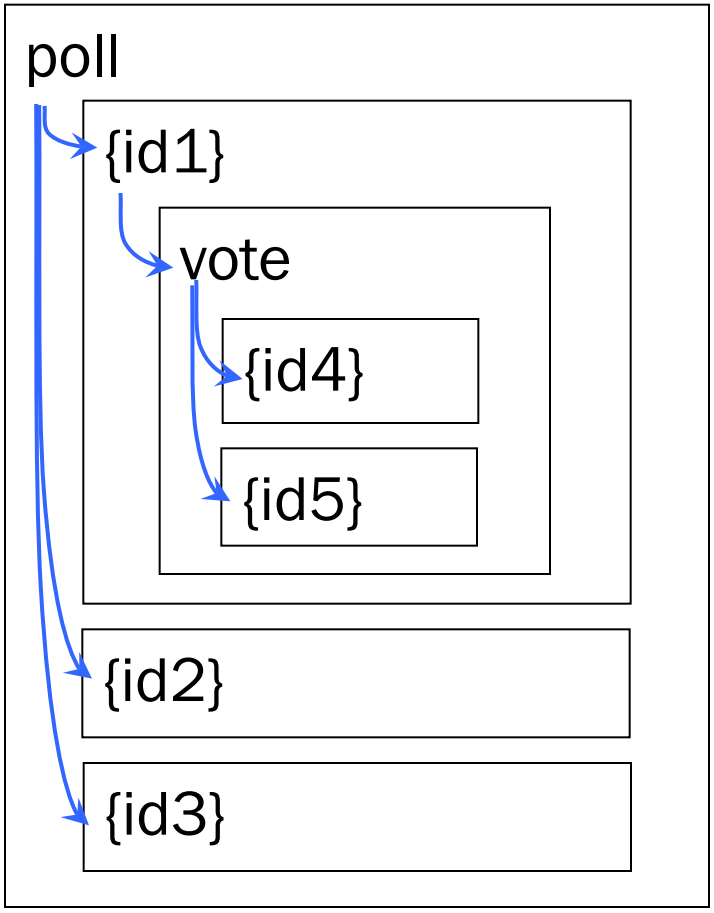
	GET	PUT	POST	DELETE
/loan	✓	✓	✓	✓
/balance	✓	✘	✘	✘
/client	✓	✓	✓	✘
/book	✓	✓	✓	✓
/order	✓	?	✓	✘
/soap	✘	✘	✓	✘

Design Space



Simple Doodle API Example

1. Resources:
   polls and votes
2. Containment Relationship:

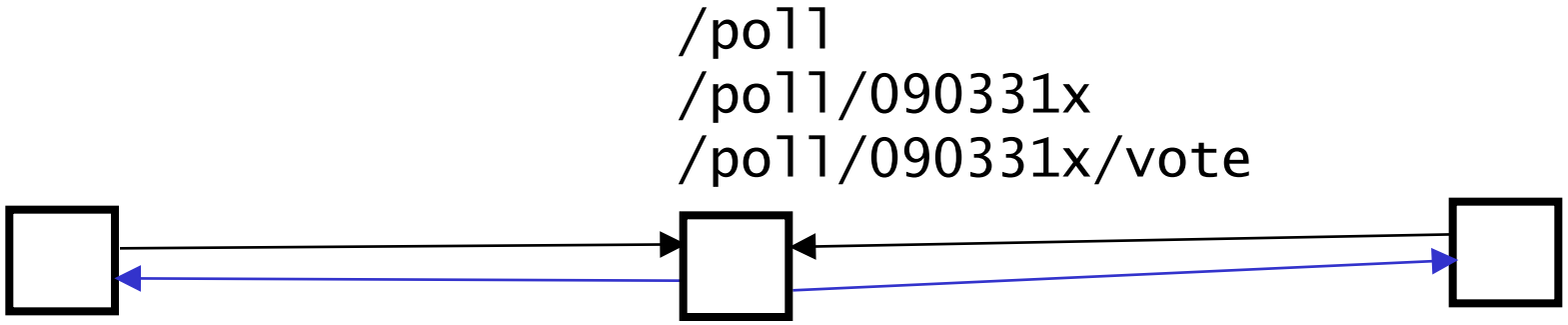


	GET	PUT	POST	DELETE
/poll	✓	✗	✓	✗
/poll/*{id}*	✓	✓	✗	✓
/poll/*{id}*/vote	✓	✗	✓	✗
/poll/*{id}*/vote/*{id}*	✓	✓	✗	?

3. URIs embed IDs of “child” instance resources
4. POST on the container is used to create child resources
5. PUT/DELETE for updating and removing child resources

Simple Doodle API Example

1. Creating a poll
(transfer the state of a new poll on the Doodle service)



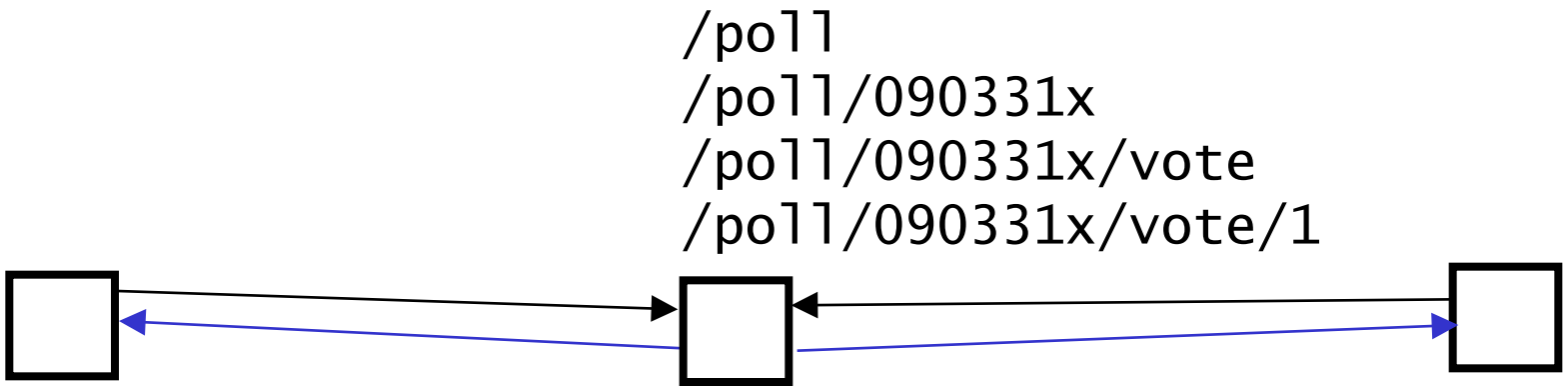

```
POST /poll
<options>A,B,C</options>

201 Created
Location: /poll/090331x
```

```
GET /poll/090331x

200 OK
<options>A,B,C</options>
<votes href="/vote"/>
```

2. Reading a poll
(transfer the state of the poll from the Doodle service)

Simple Doodle API Example

- Participating in a poll by creating a new vote sub-resource

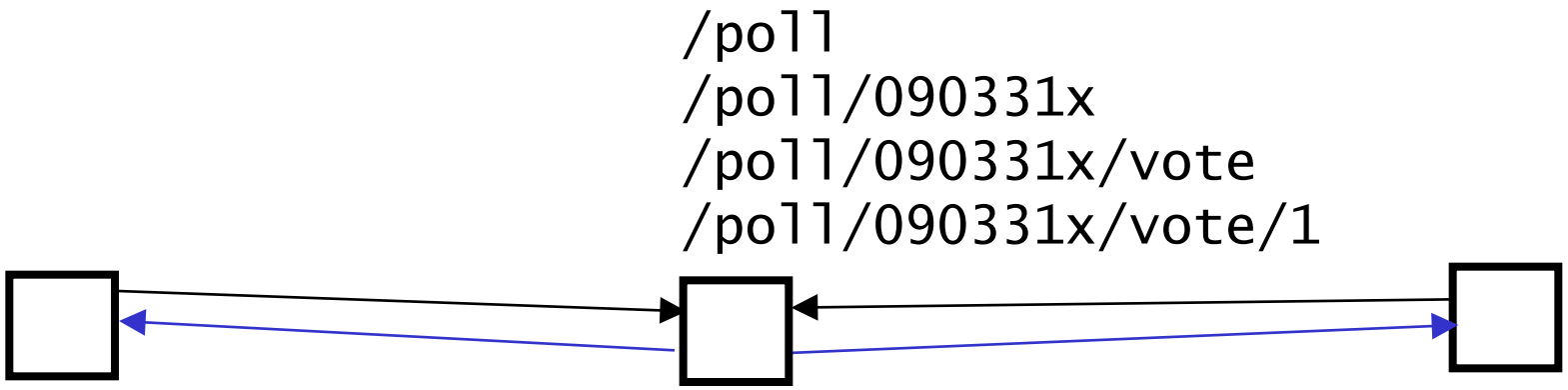



```
POST /poll/090331x/vote
<name>C. Pautasso</name>
<choice>B</choice>

201 Created
Location:
/poll/090331x/vote/1
```

```
GET /poll/090331x

200 OK
<options>A,B,C</options>
<votes><vote id="1">
<name>C. Pautasso</name>
<choice>B</choice>
</vote></votes>
```

Simple Doodle API Example

- Existing votes can be updated (access control headers not shown)

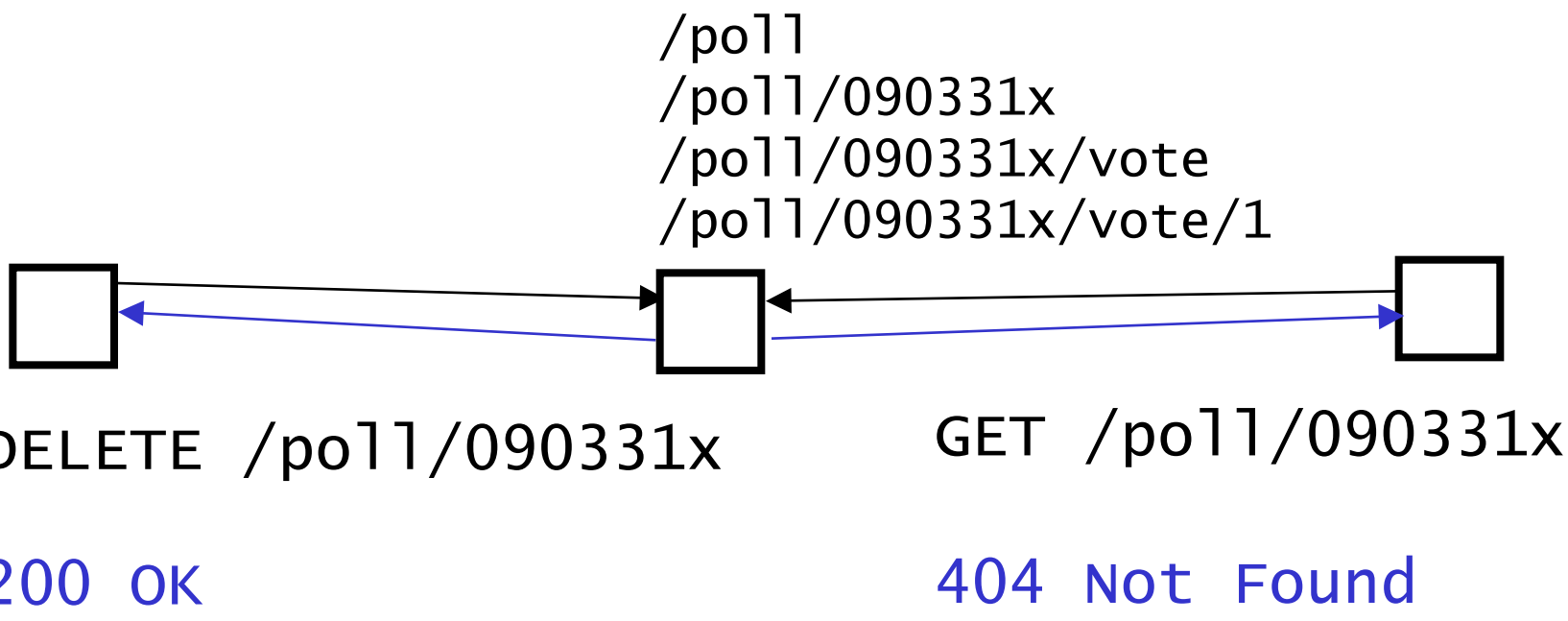



```
PUT /poll/090331x/vote/1
<name>C. Pautasso</name>
<choice>C</choice>

200 OK
```

```
GET /poll/090331x

200 OK
<options>A,B,C</options>
<votes><vote id="/1">
<name>C. Pautasso</name>
<choice>C</choice>
</vote></votes>
```

Simple Doodle API Example

- Polls can be deleted once a decision has been made



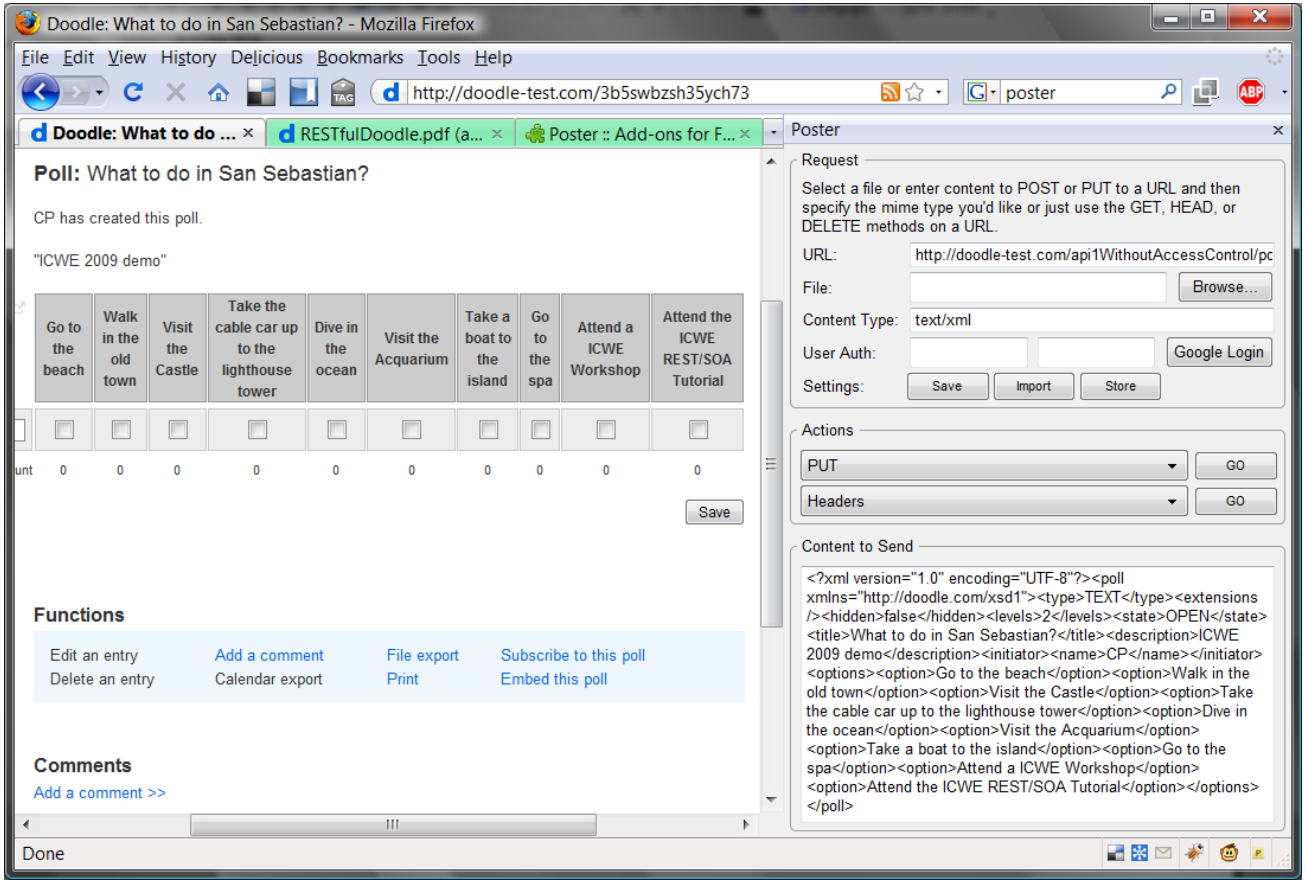
Real Doodle Demo

- Info on the real Doodle API:

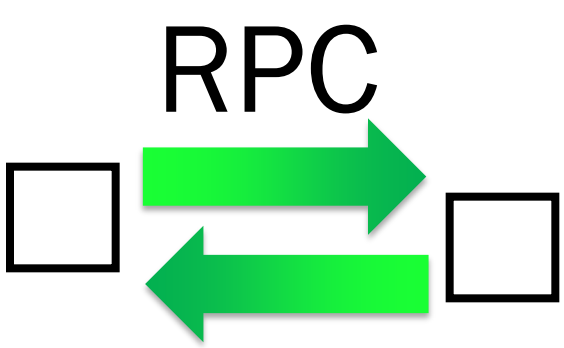
http://doodle.com/xsd1/RESTfulDoodle.pdf

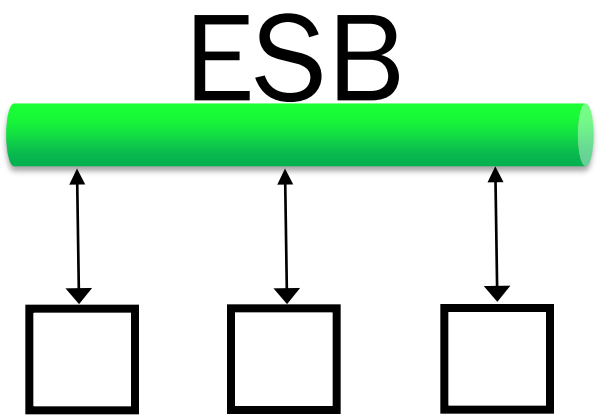
- Lightweight demo with Poster Firefox Extension:

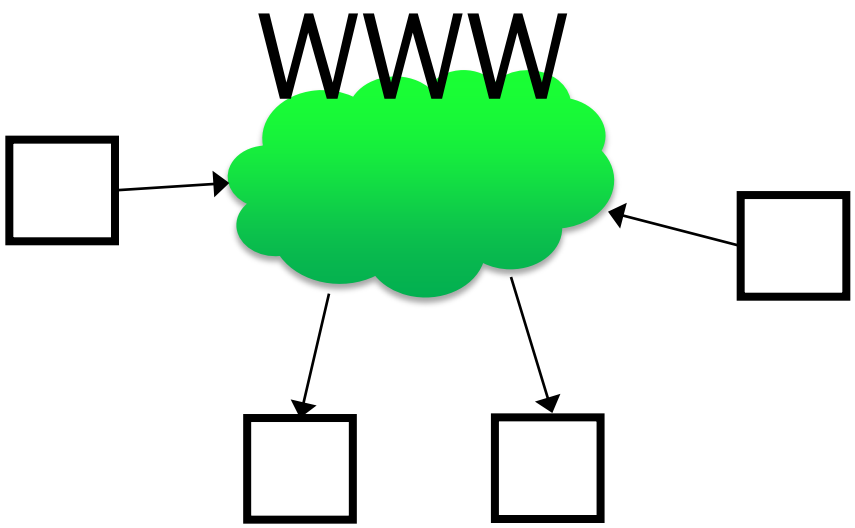
http://addons.mozilla.org/en-US/firefox/addon/2691



Different software connectors

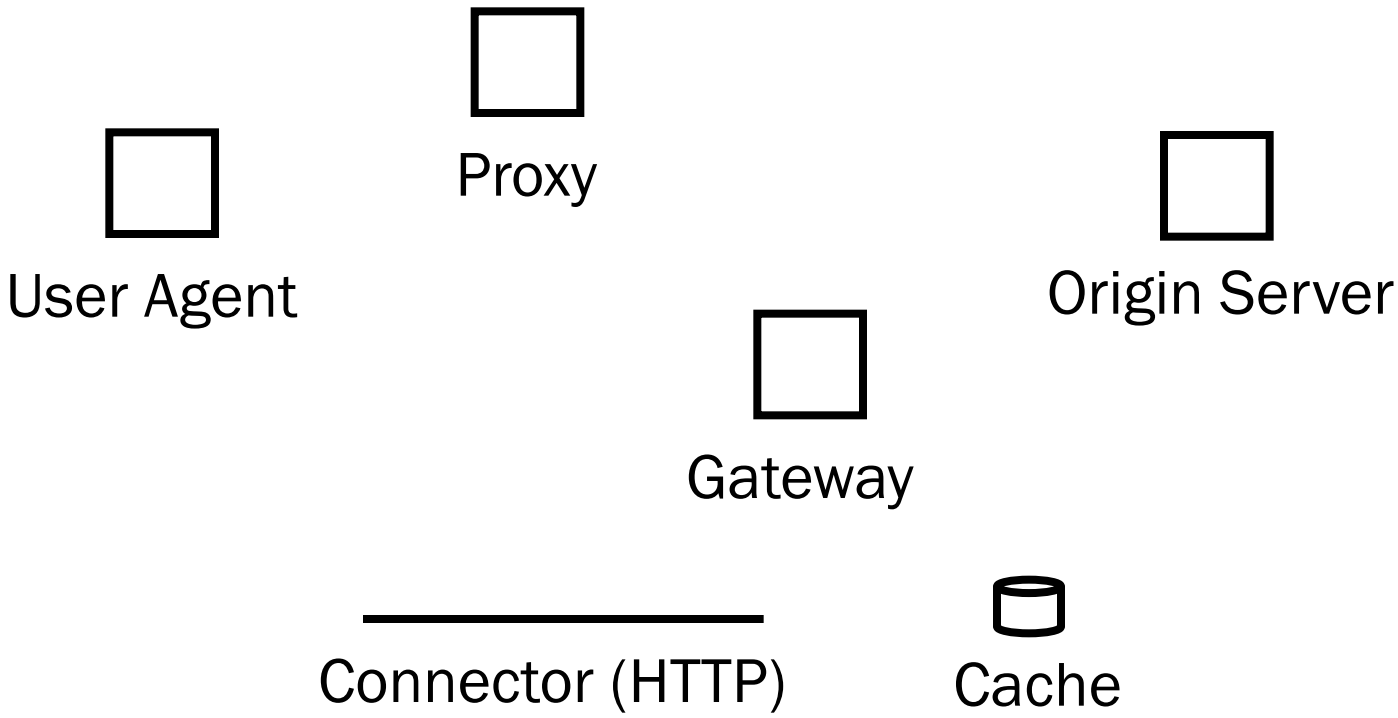




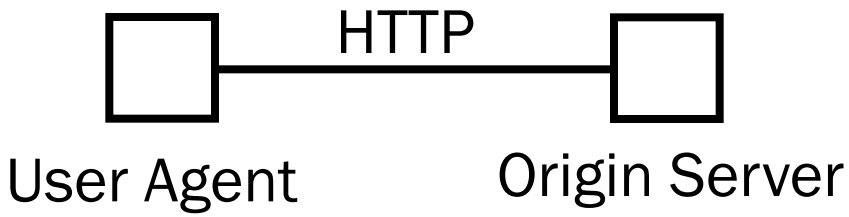


REST Architectural Elements

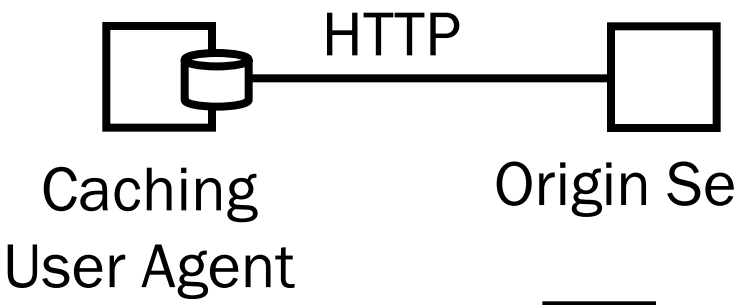
Client/Server Layered Stateless Communication Cache

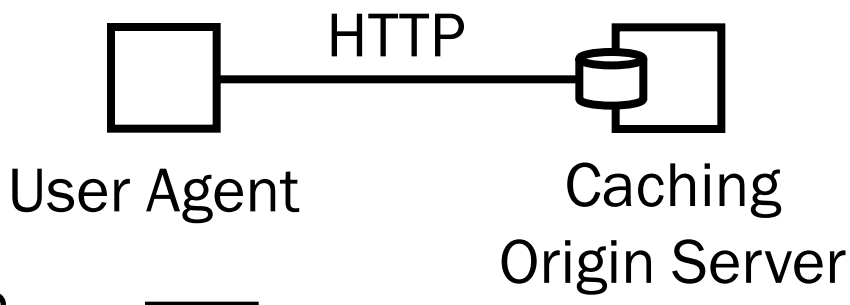


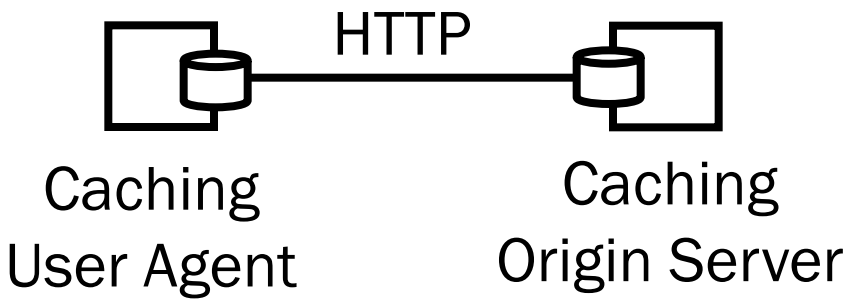
Basic Setup



Adding Caching

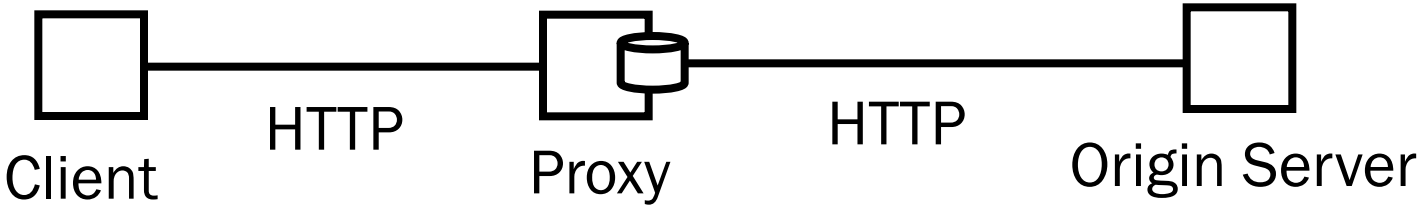




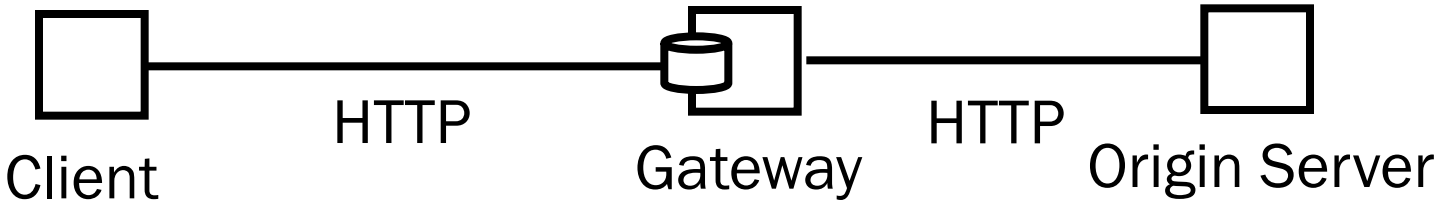


Proxy or Gateway?

Intermediaries forward (and may translate) requests and responses



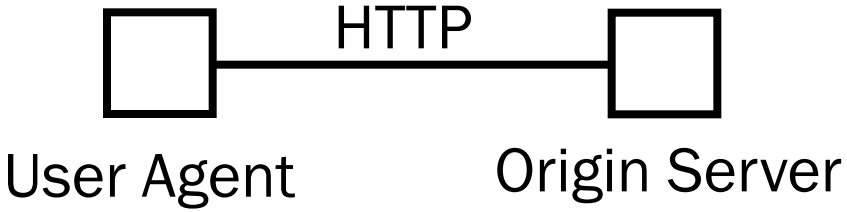
A proxy is chosen by the Client (for caching, or access control)



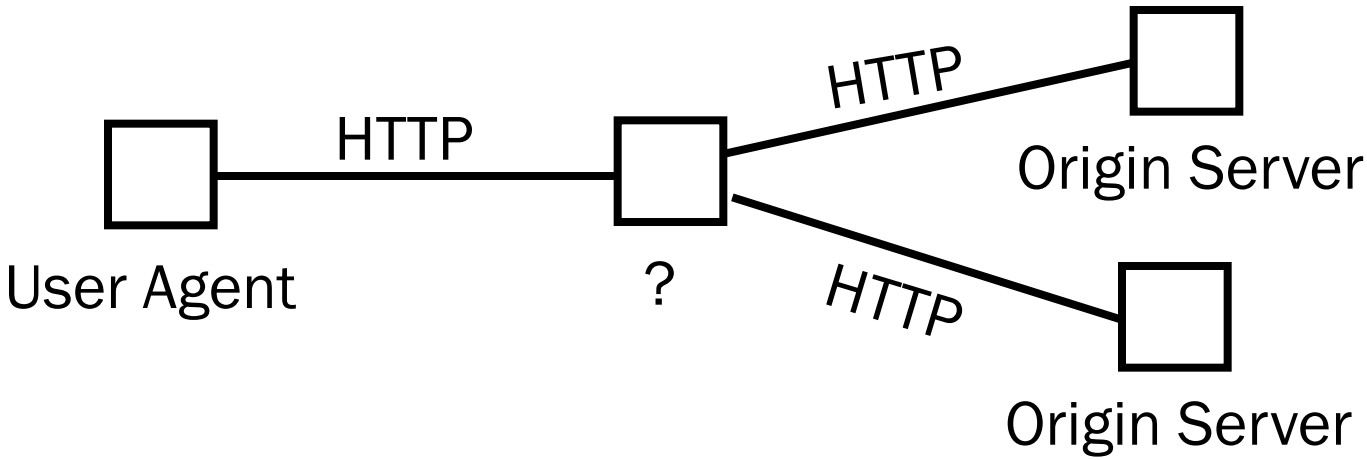
The use of a gateway (or reverse proxy) is imposed by the server

What about composition?

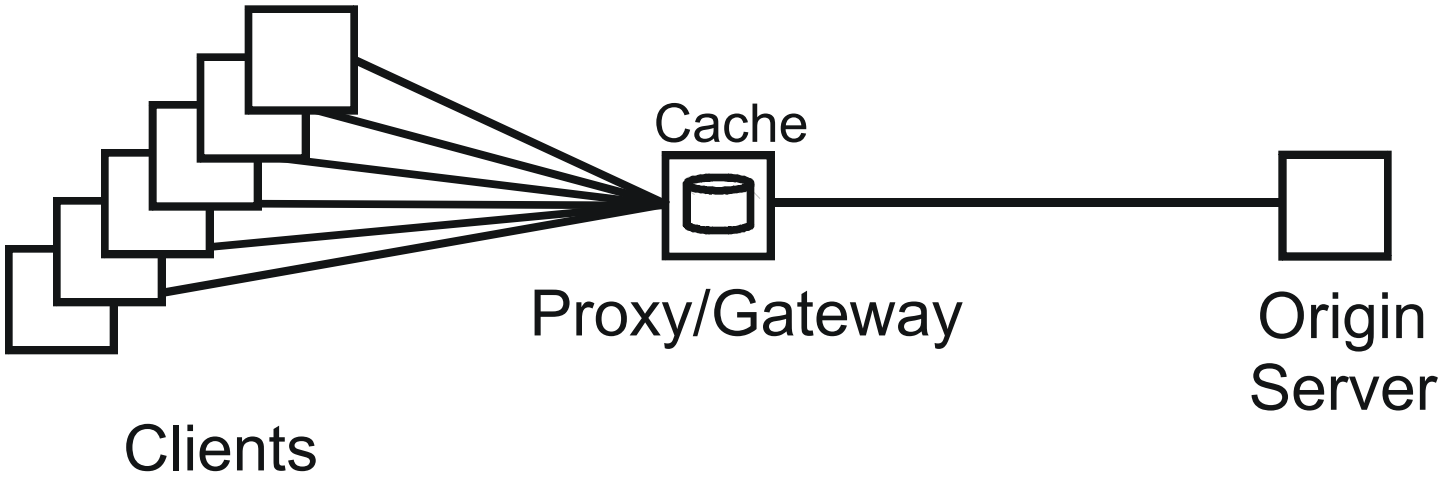
- The basic REST design elements do not take composition into account



- WS-BPEL is the standard Web service composition language. Business process models are used to specify how a collection of services is orchestrated into a composite service
- Can we apply WS-BPEL to RESTful services?

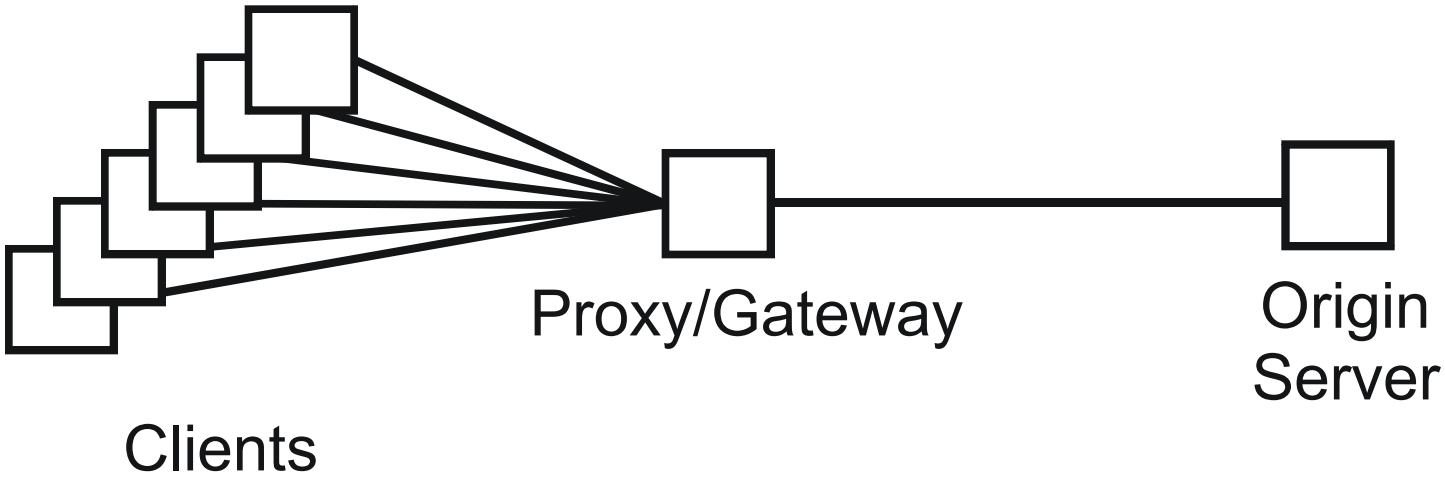


REST Scalability



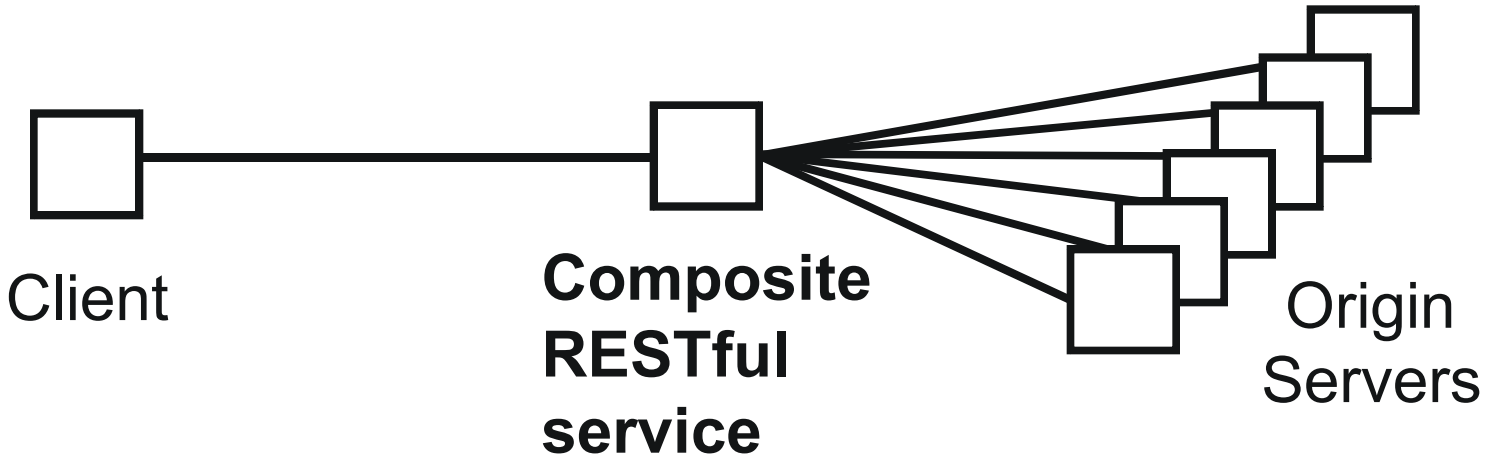
- One example of REST middleware is to help with the scalability of a server, which may need to service a very large number of clients

REST Composition



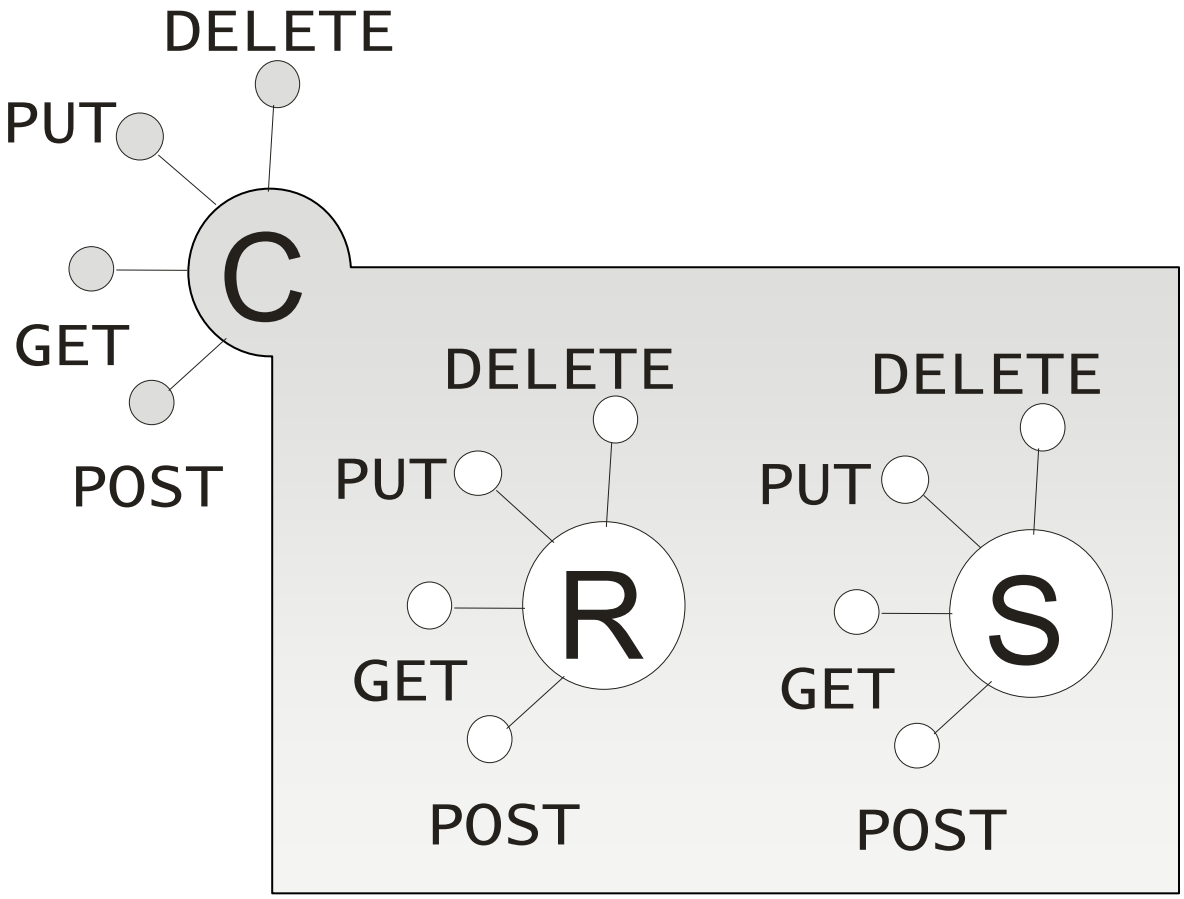
- Composition shifts the attention to the client which should consume and aggregate from many servers

REST Composition



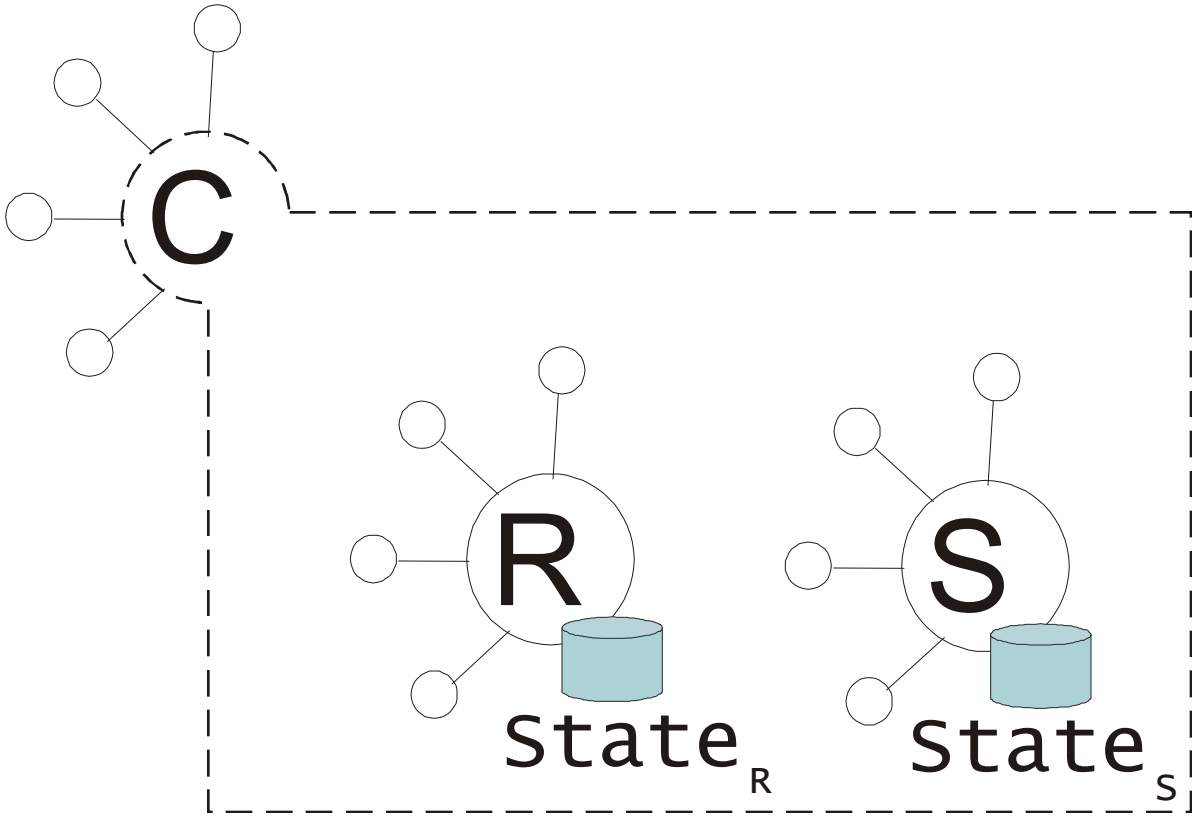
- The “proxy” intermediate element which aggregates the resources provided by multiple servers plays the role of a composite RESTful service
- Can/Should we implement it with BPM?

Composite Resources



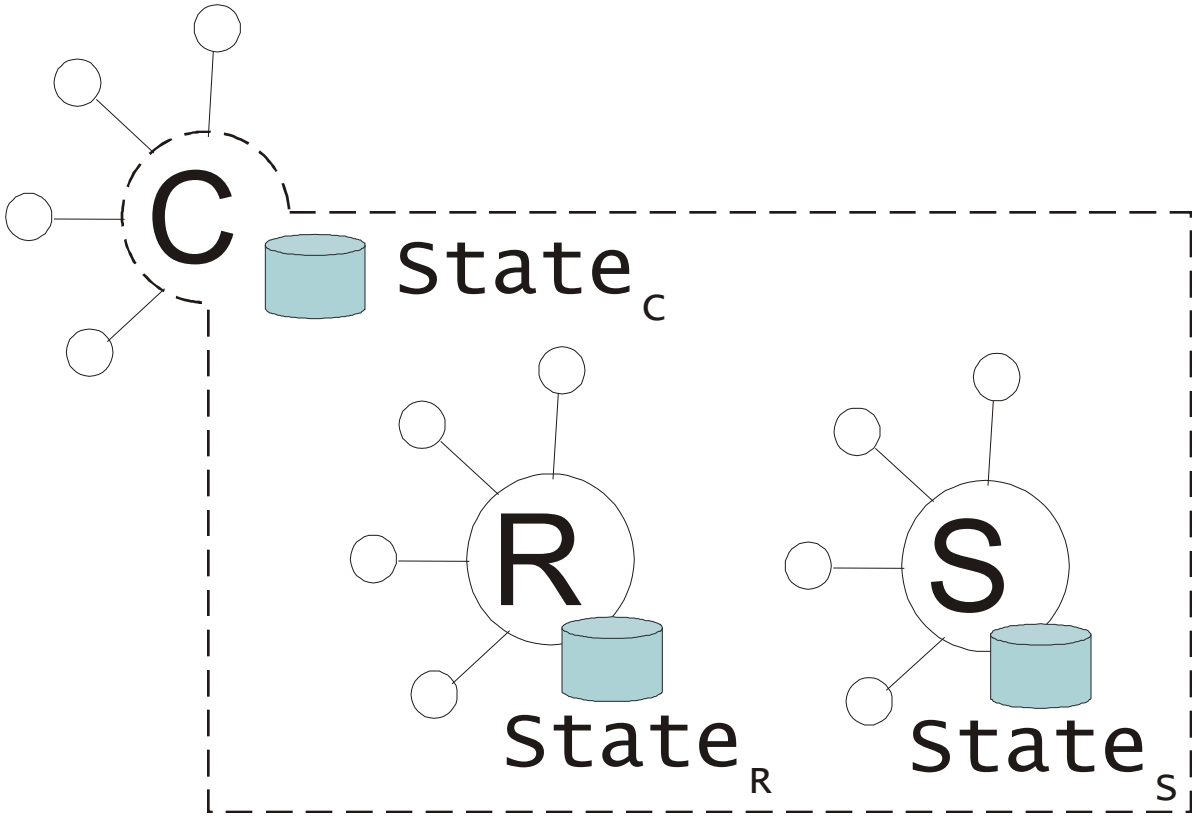
Composite Resources

- The composite resource only aggregates the state of its component resources

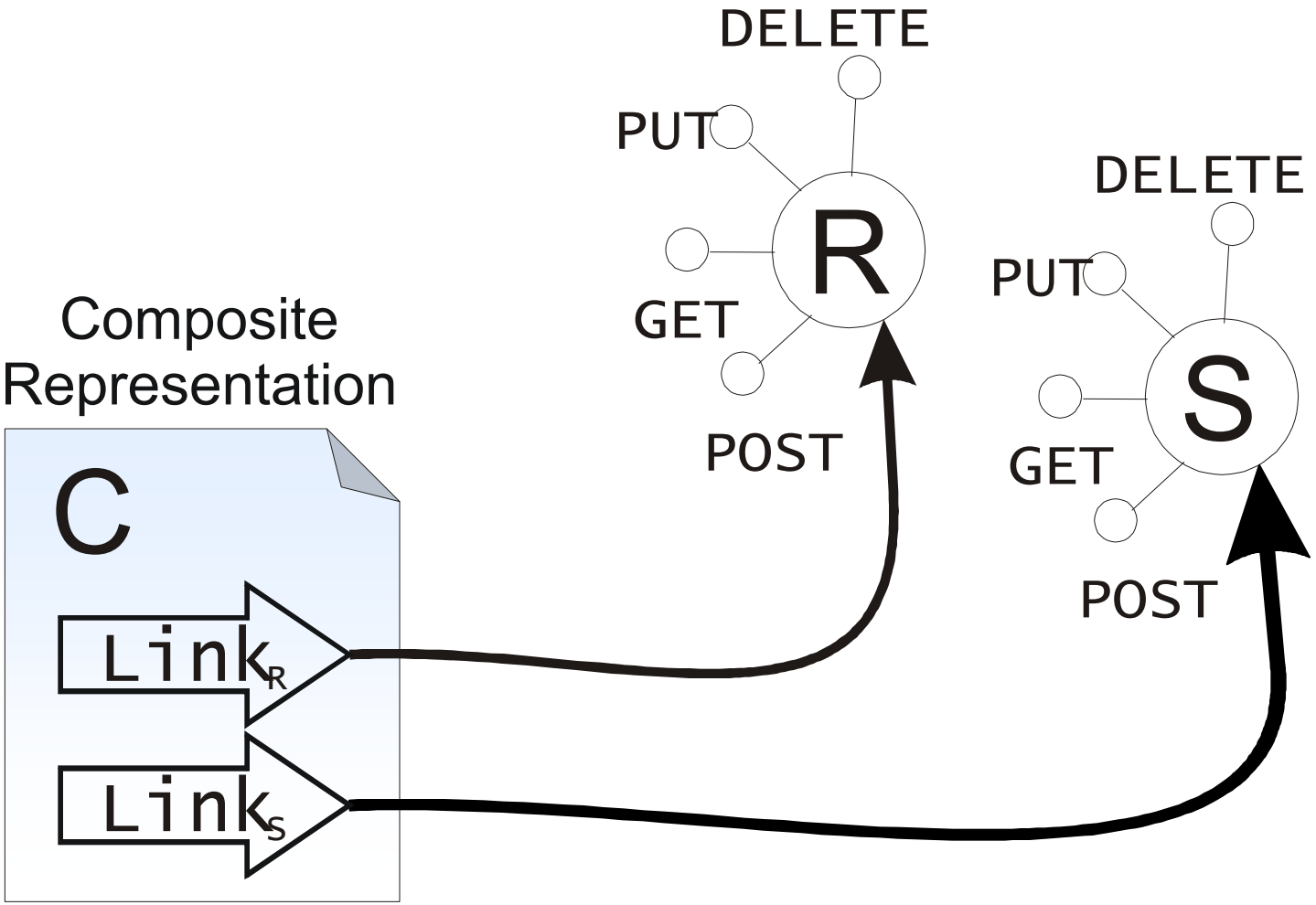


Composite Resources

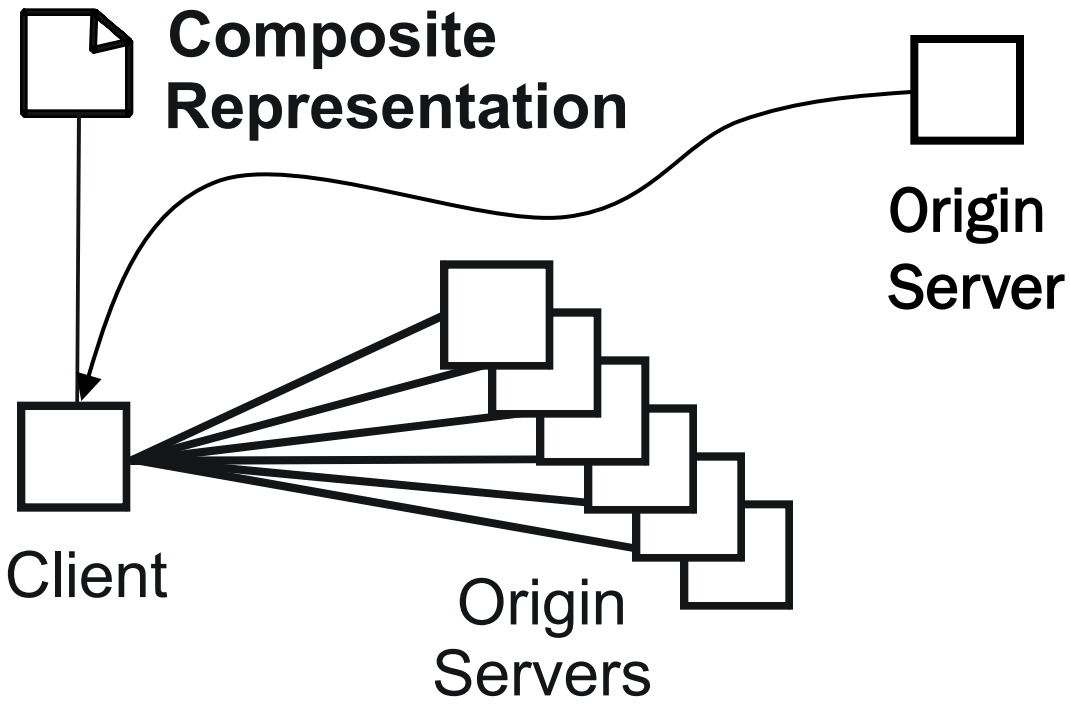
- The composite resource augments (or caches) the state of its component resources



Composite Representation

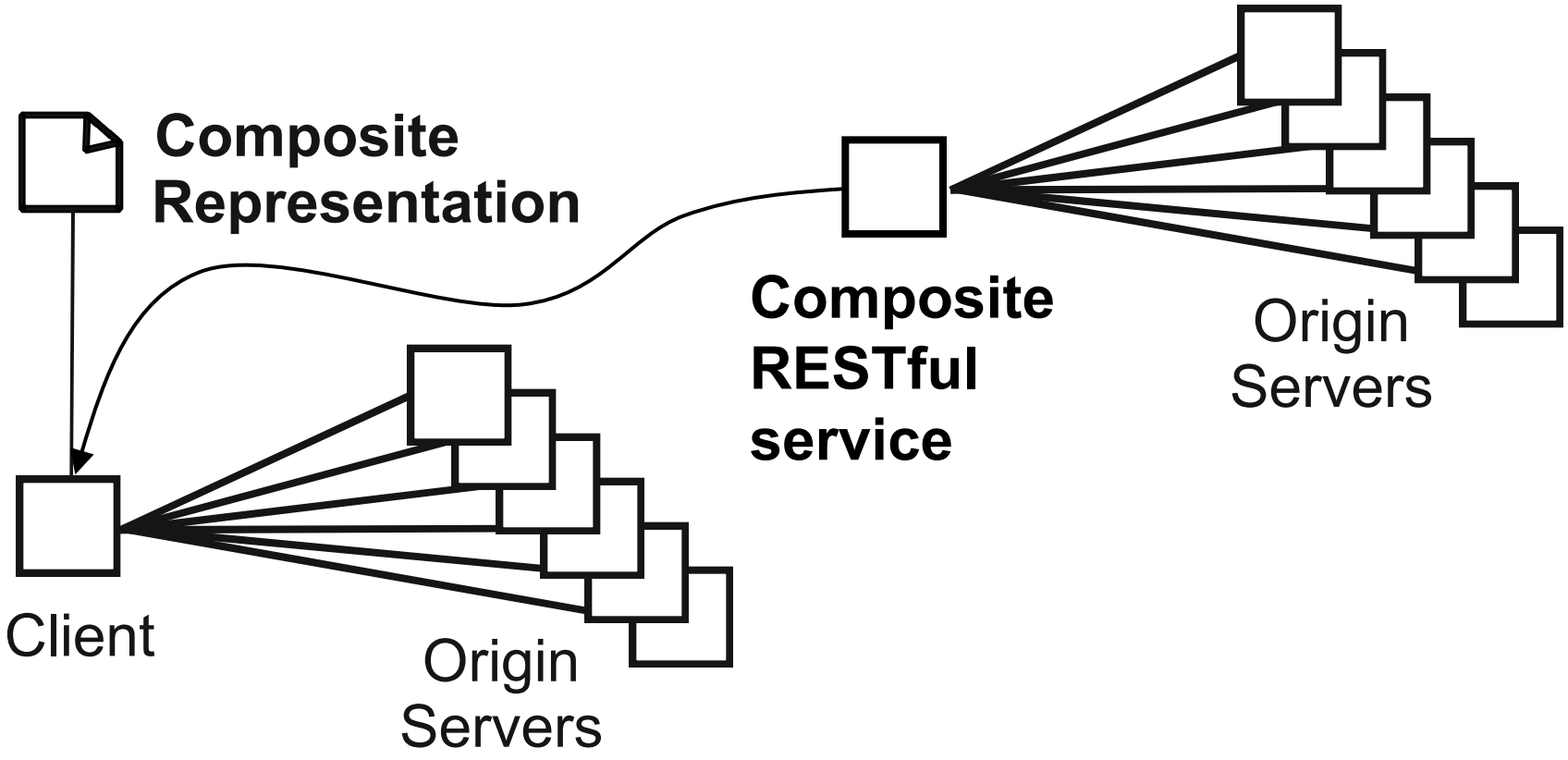


Composite Representation



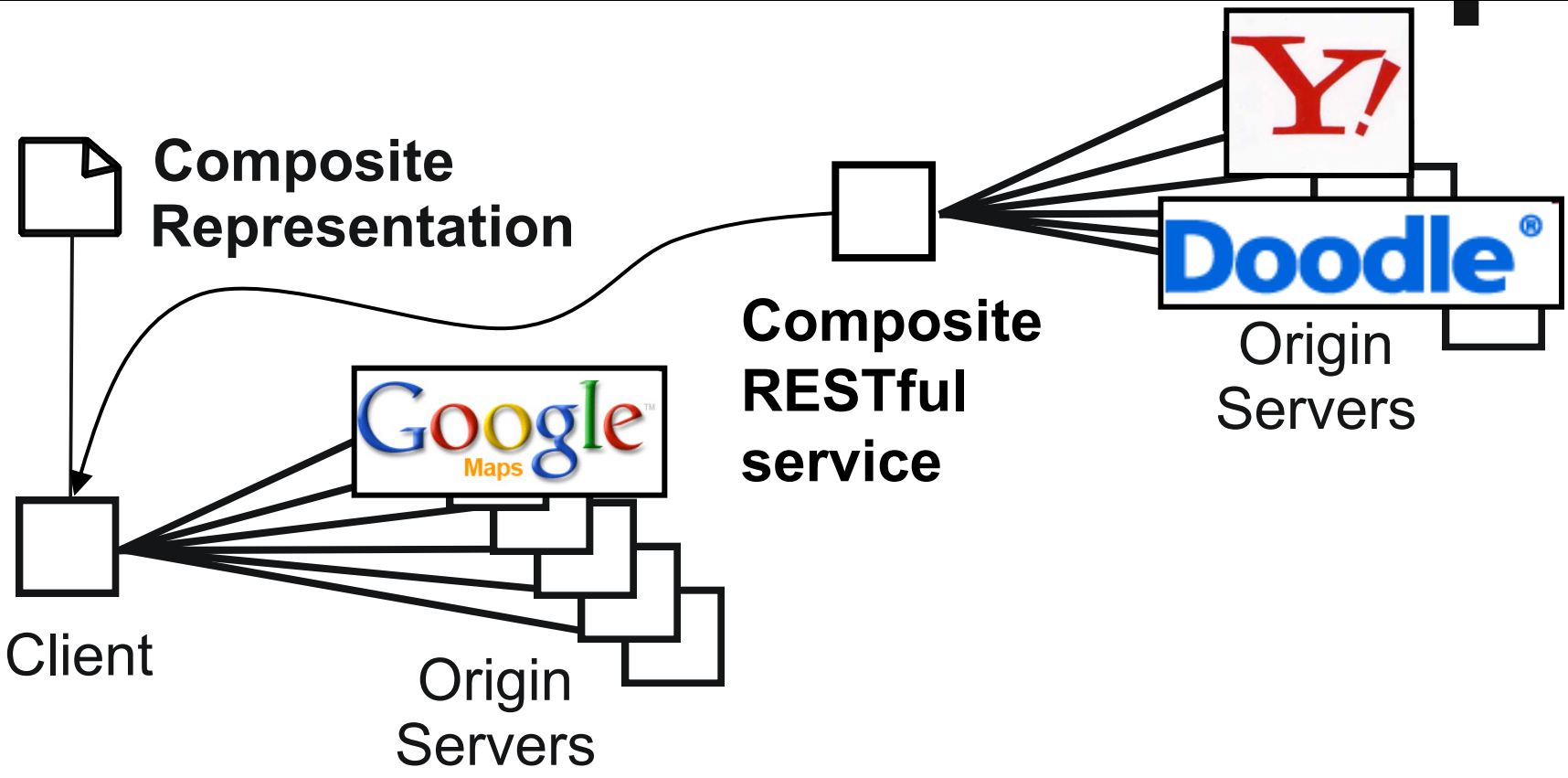
- A composite representation is interpreted by the client that follows its hyperlinks and aggregates the state of the referenced component resources

Bringing it all together



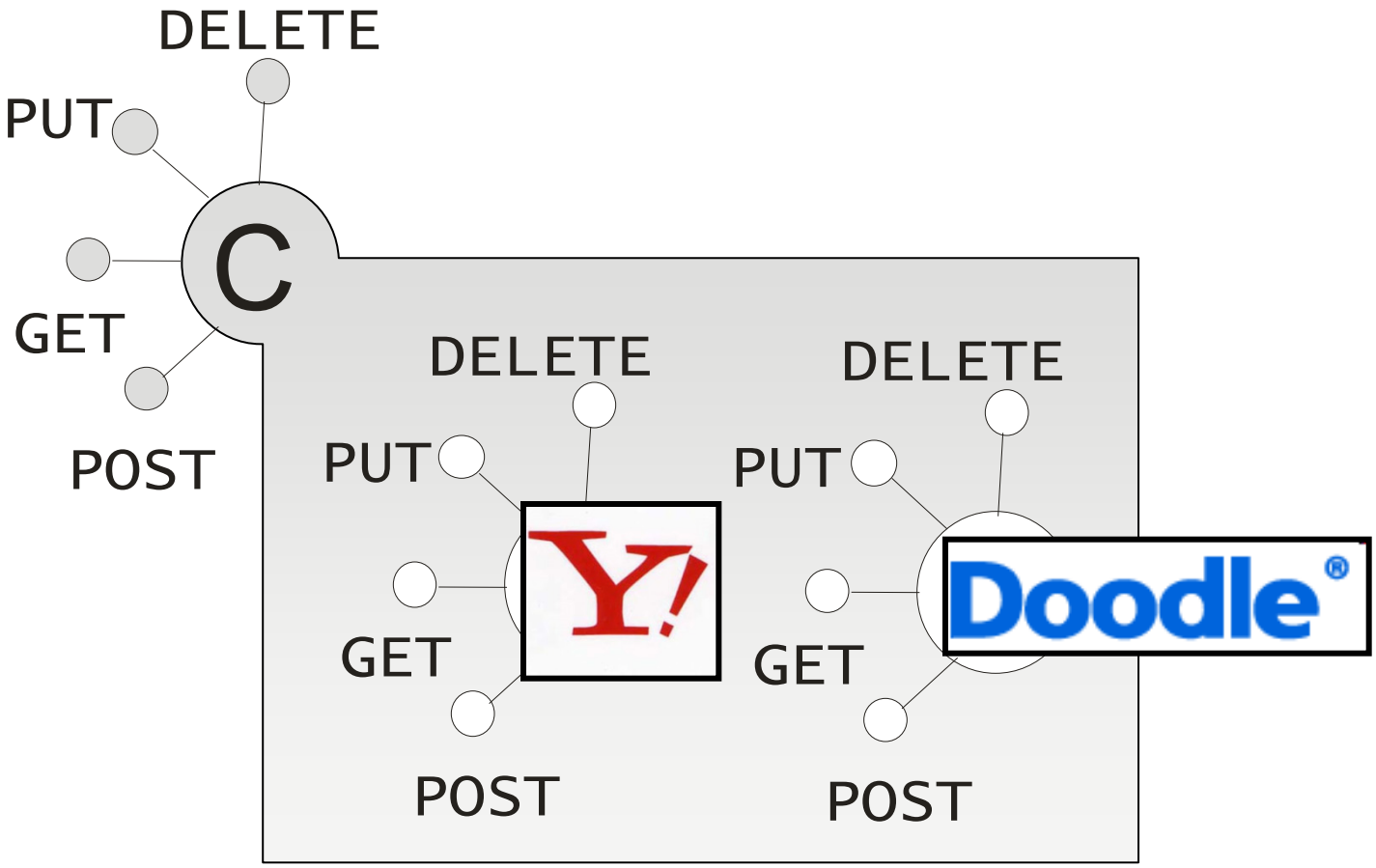
- A composite representation can be produced by a composite service too

Doodle Map Example

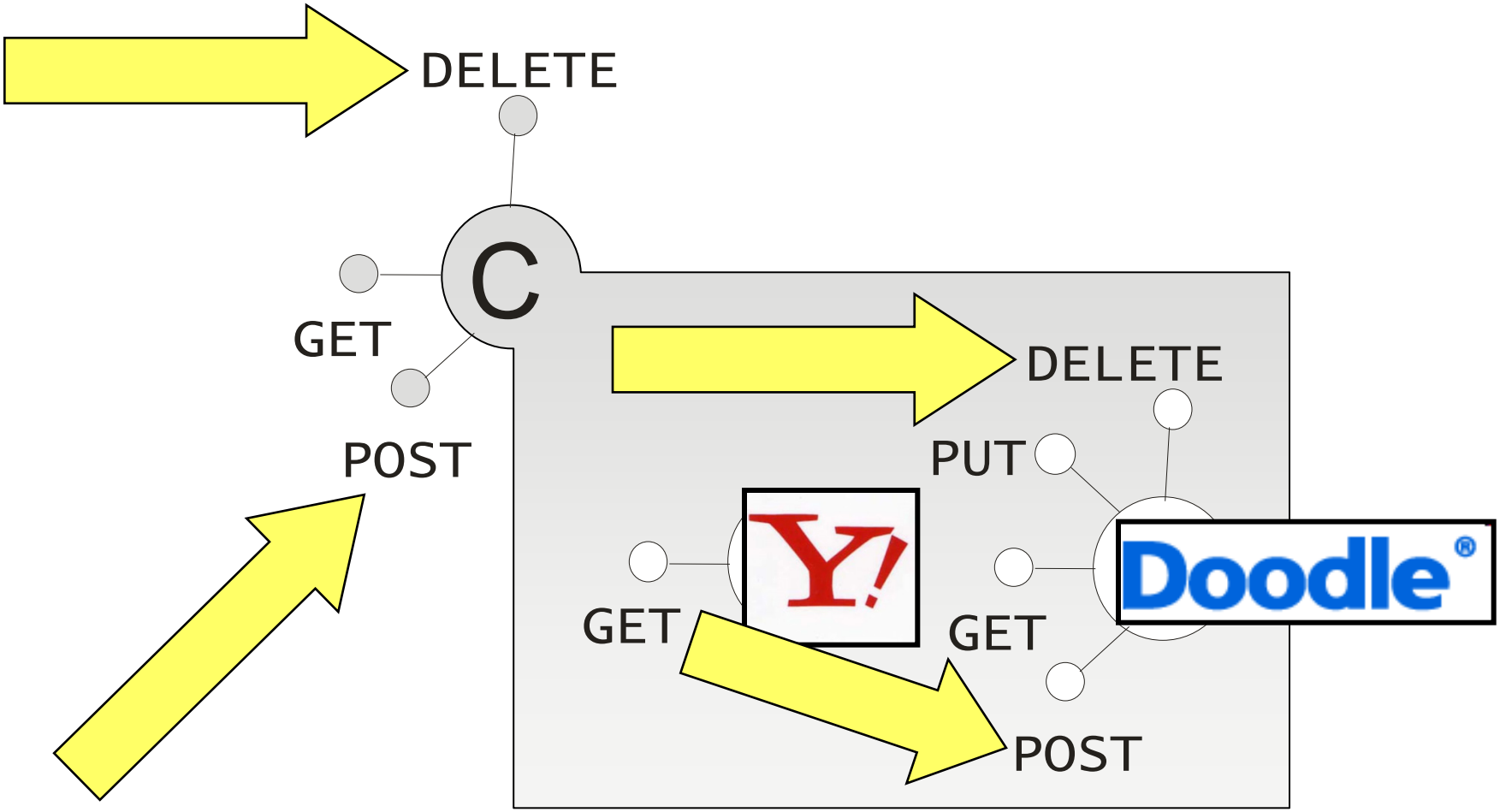


- Vote on a meeting place based on its geographic location

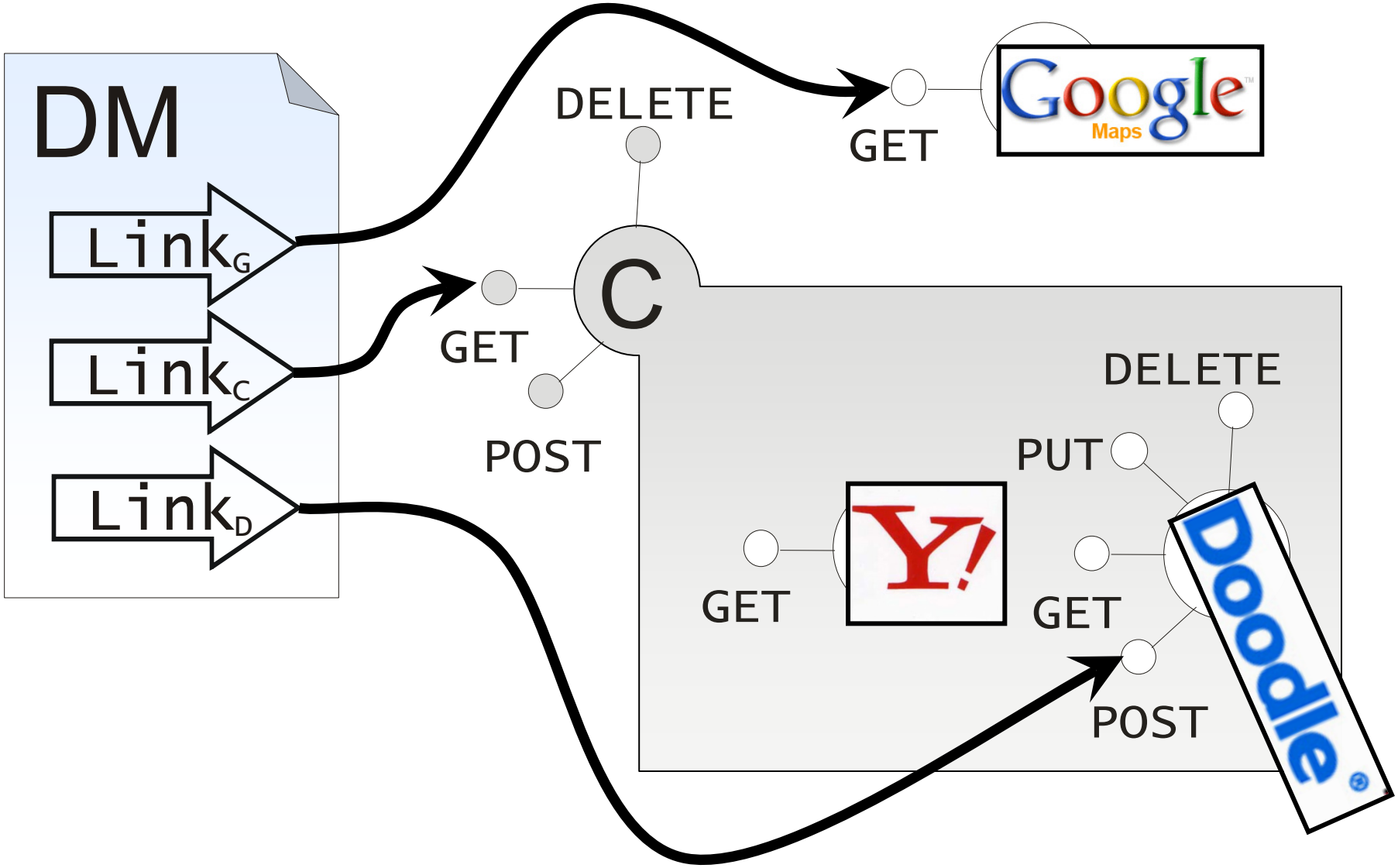
Composite Resource



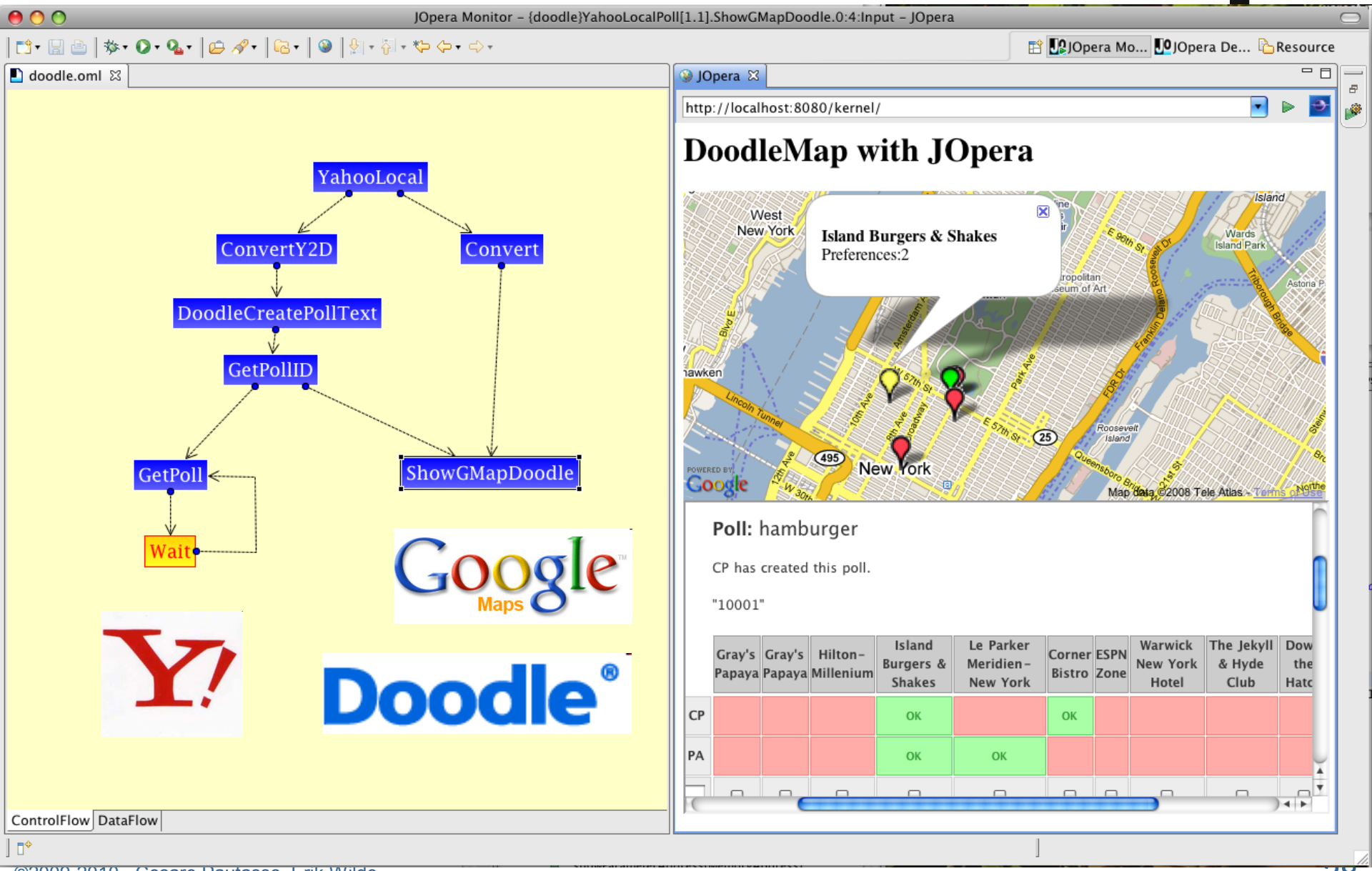
Composite Resource



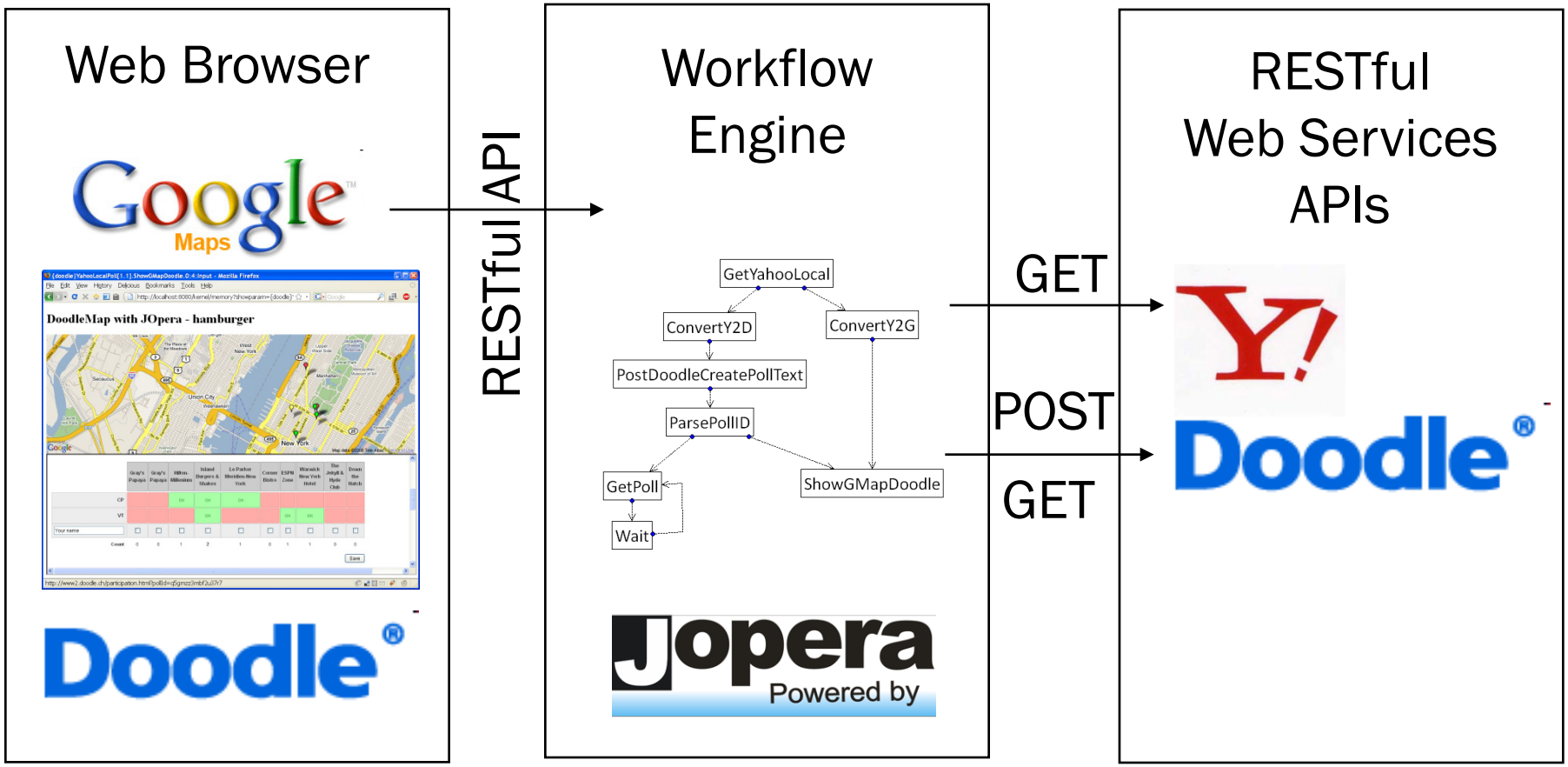
Composite Representation



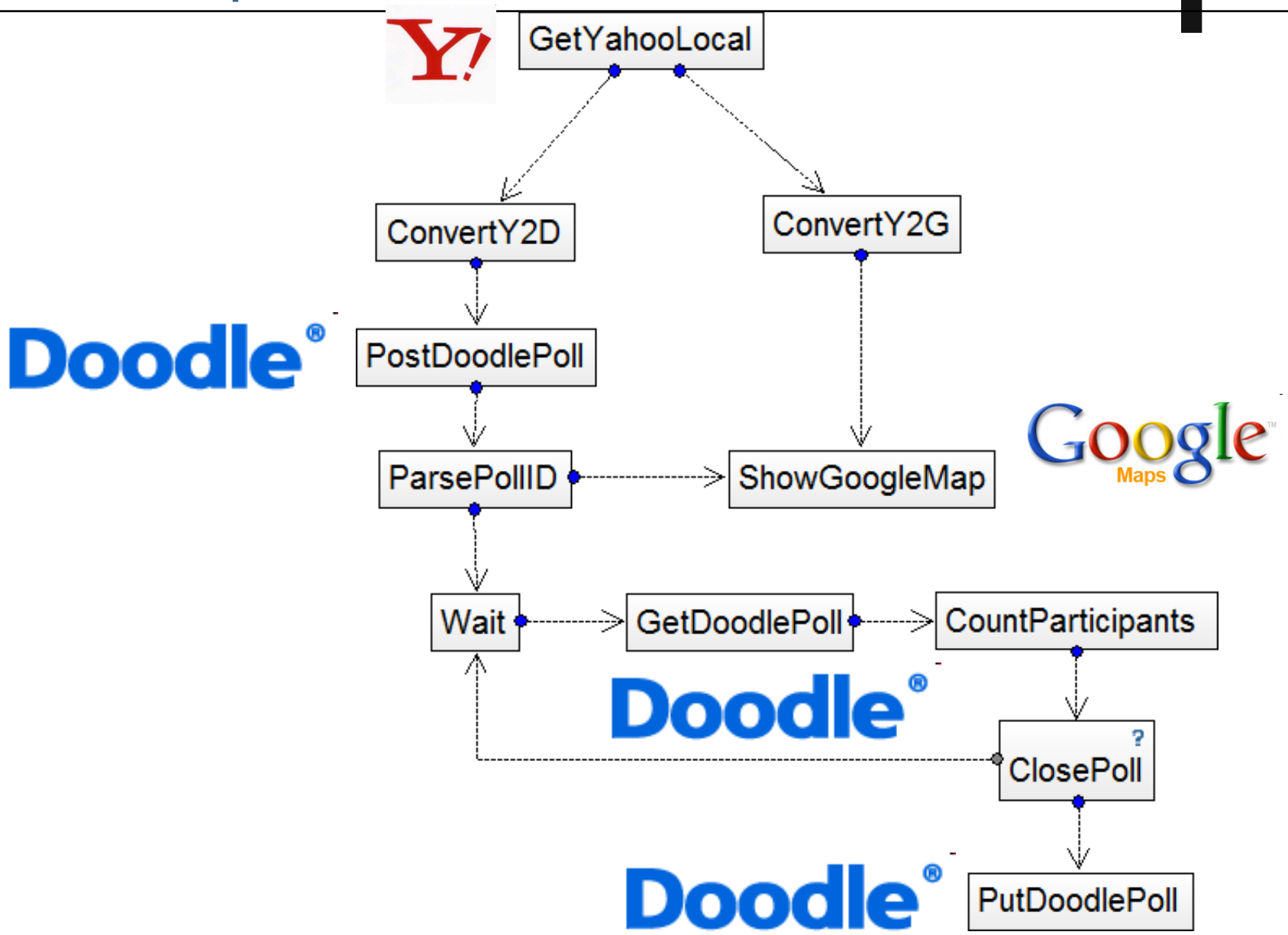
Demo


JOpera Monitor – {doodle}YahooLocalPoll[1.1].ShowGMapDoodle.0:4:Input – JOpera
doodle.oml
YahooLocal
ConvertY2D
Convert
DoodleCreatePollText
GetPollID
GetPoll
ShowGMapDoodle
Wait
Google Maps
Doodle
ControlFlow
DataFlow
JOpera
http://localhost:8080/kernel/
DoodleMap with JOpera
Island Burgers & Shakes
Preferences:2
New York
Poll: hamburger
CP has created this poll.
"10001"
Gray's Papaya
Gray's Papaya
Hilton–Millenium
Island Burgers & Shakes
Le Parker Meridien–New York
Corner Bistro
ESPN Zone
Warwick New York Hotel
The Jekyll & Hyde Club
CP
PA
OK

Doodle Map Architecture

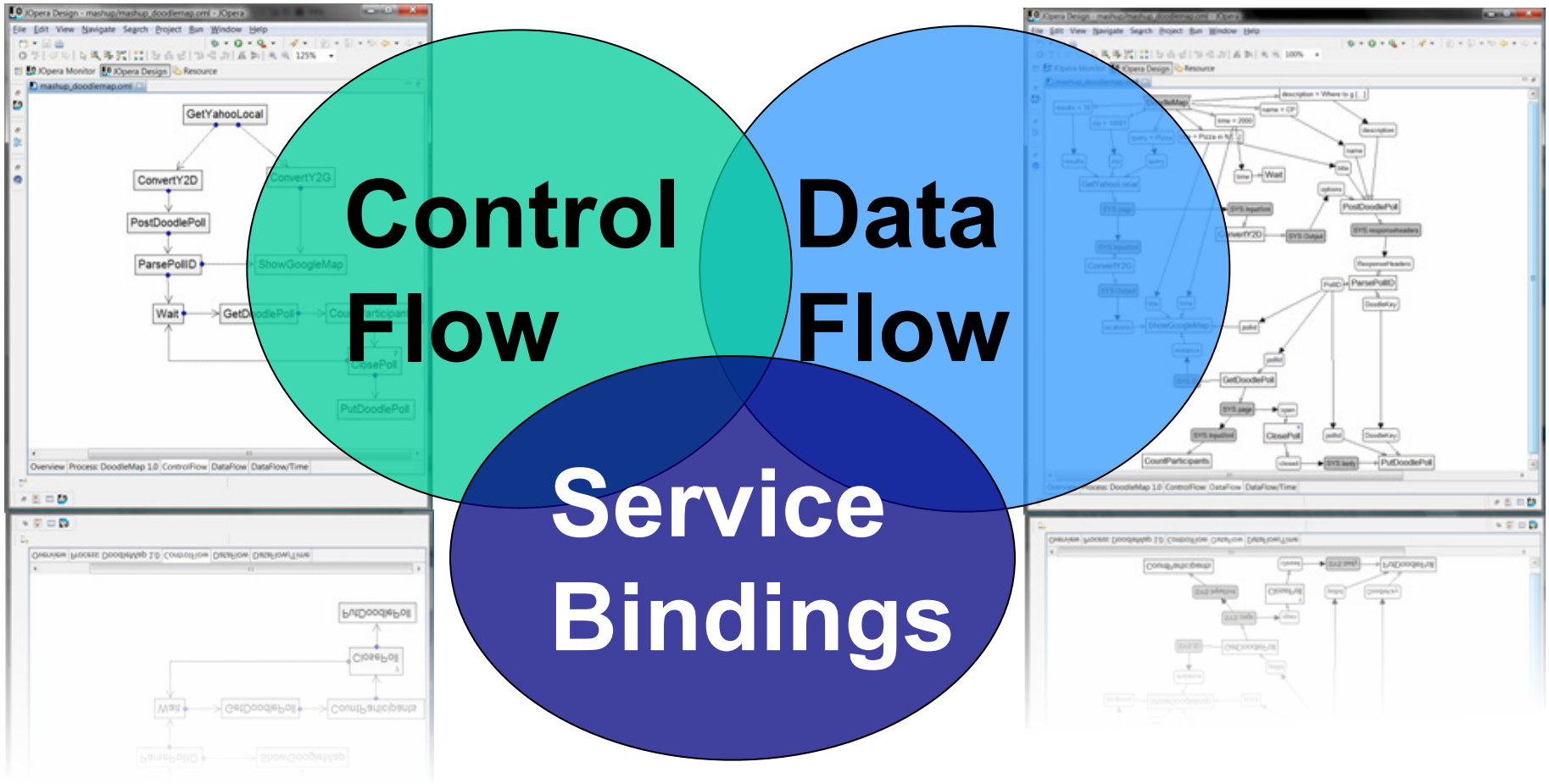


DoodleMap Model



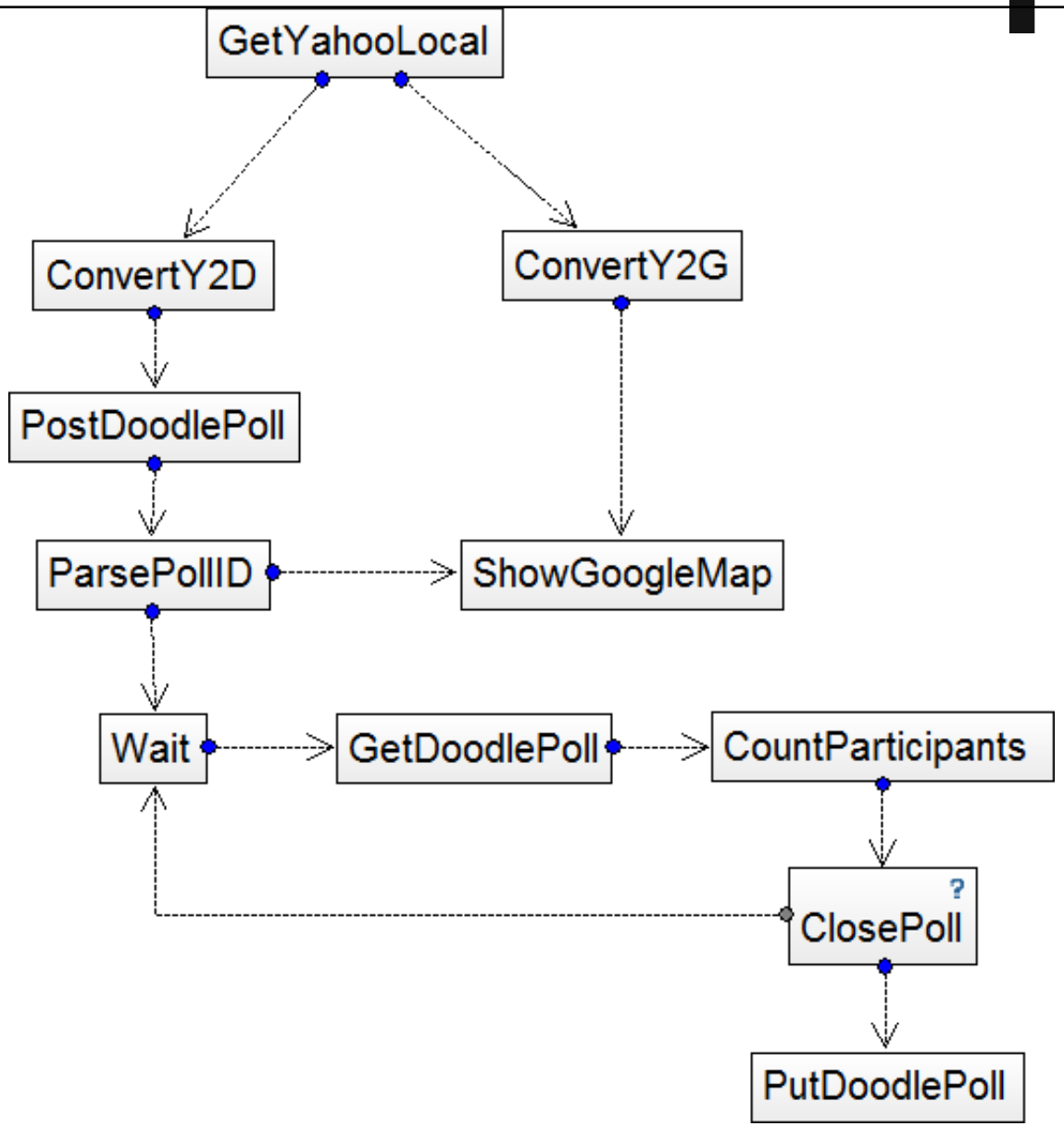
Viewpoints

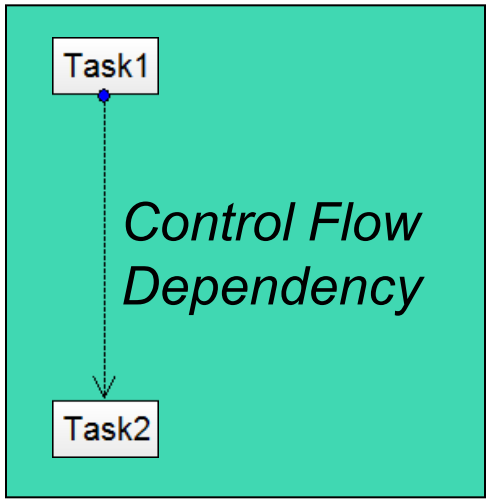
Jopera
Process Support for Web Services



JAVA | XPATH | XSLT | HTML | HTTP | ...

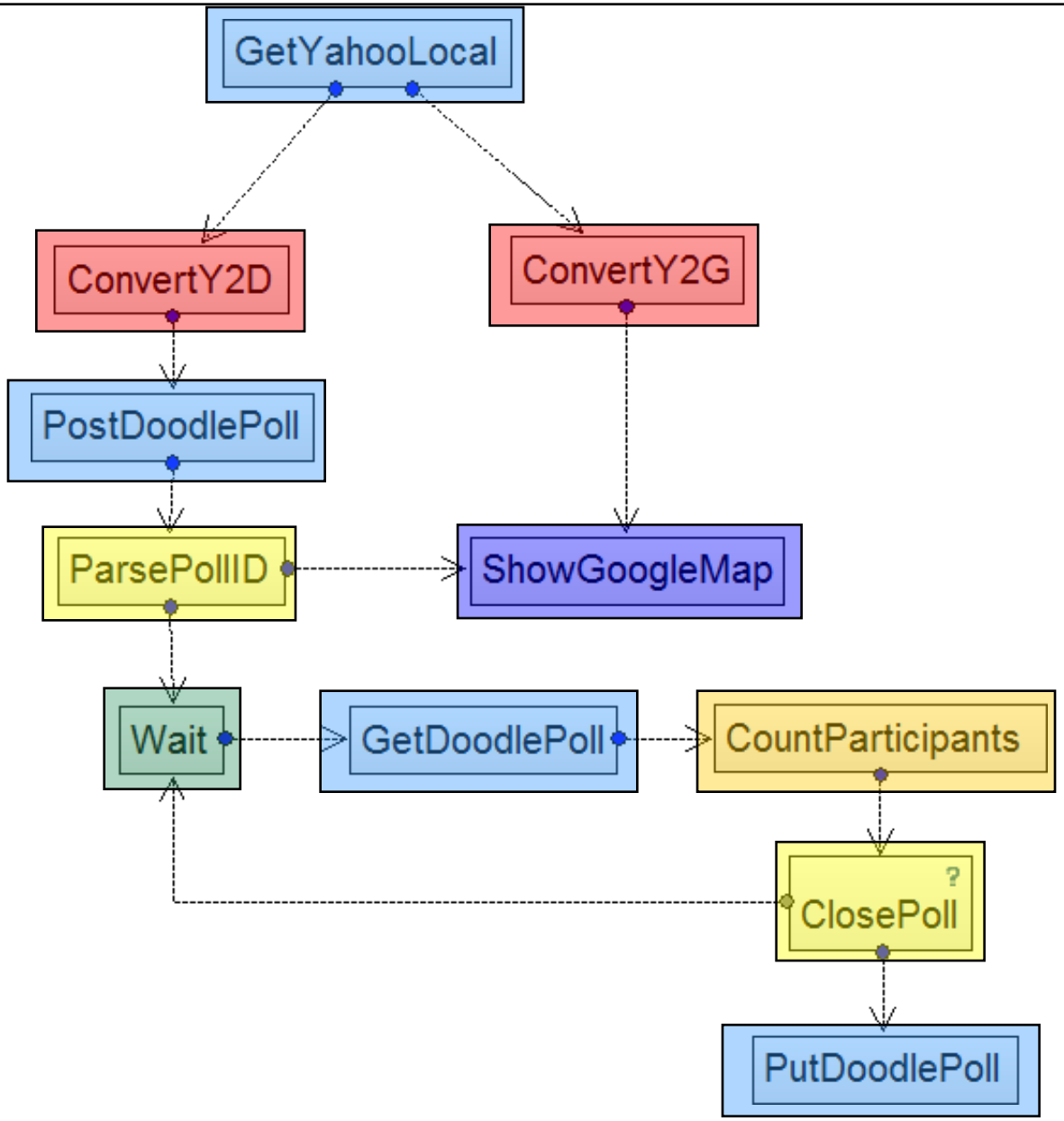
Control Flow



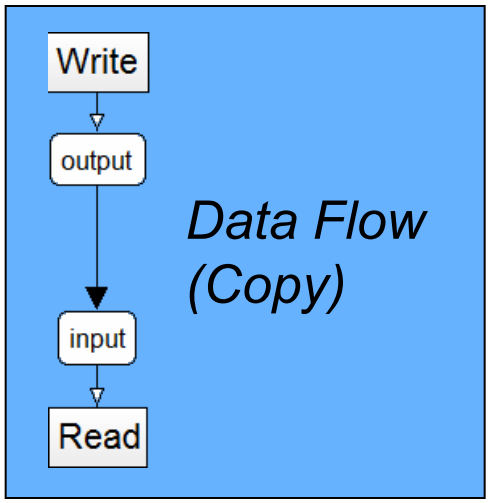


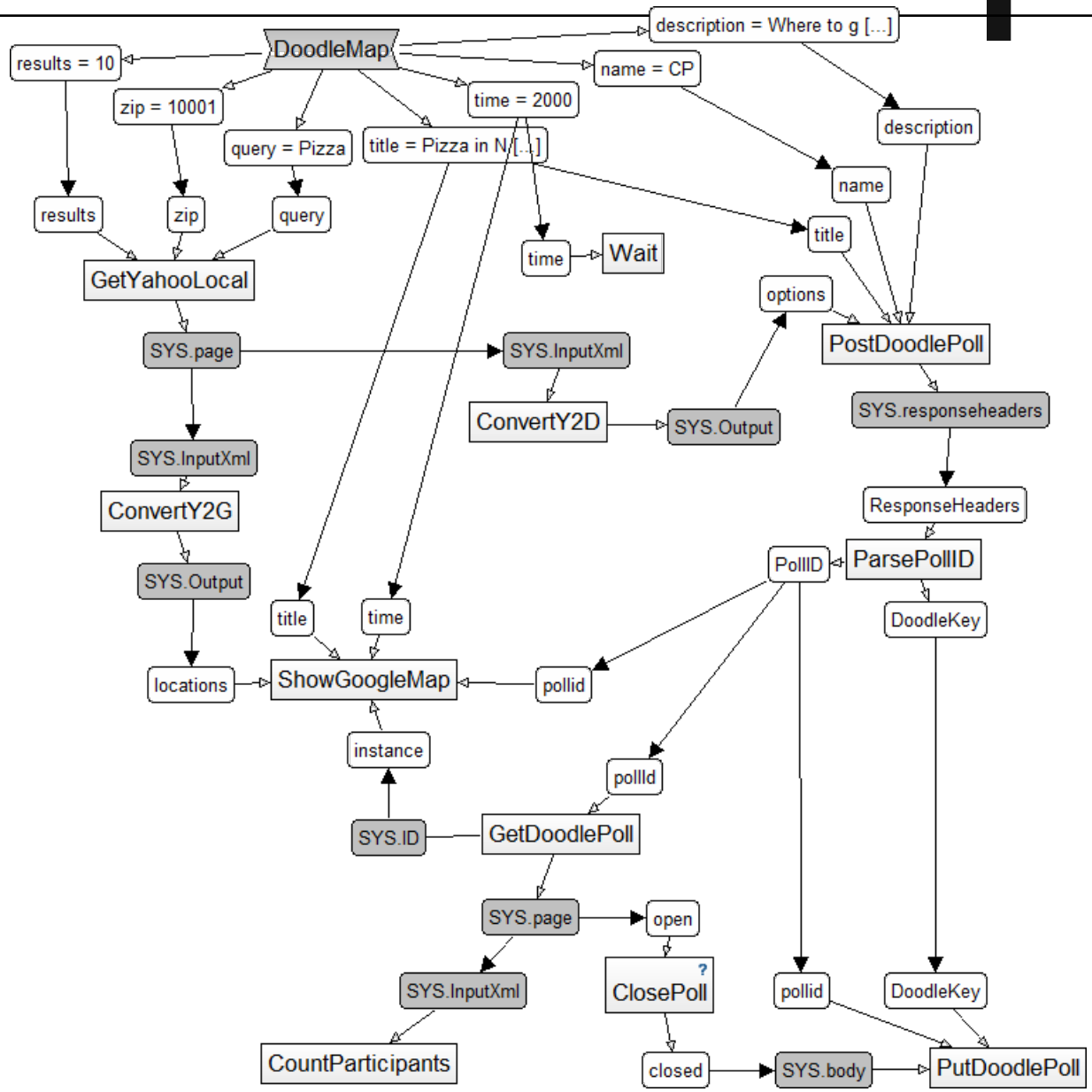
Service Bindings

- HTTP
- HTML
- XSLT
- XPATH
- JAVA
- ...

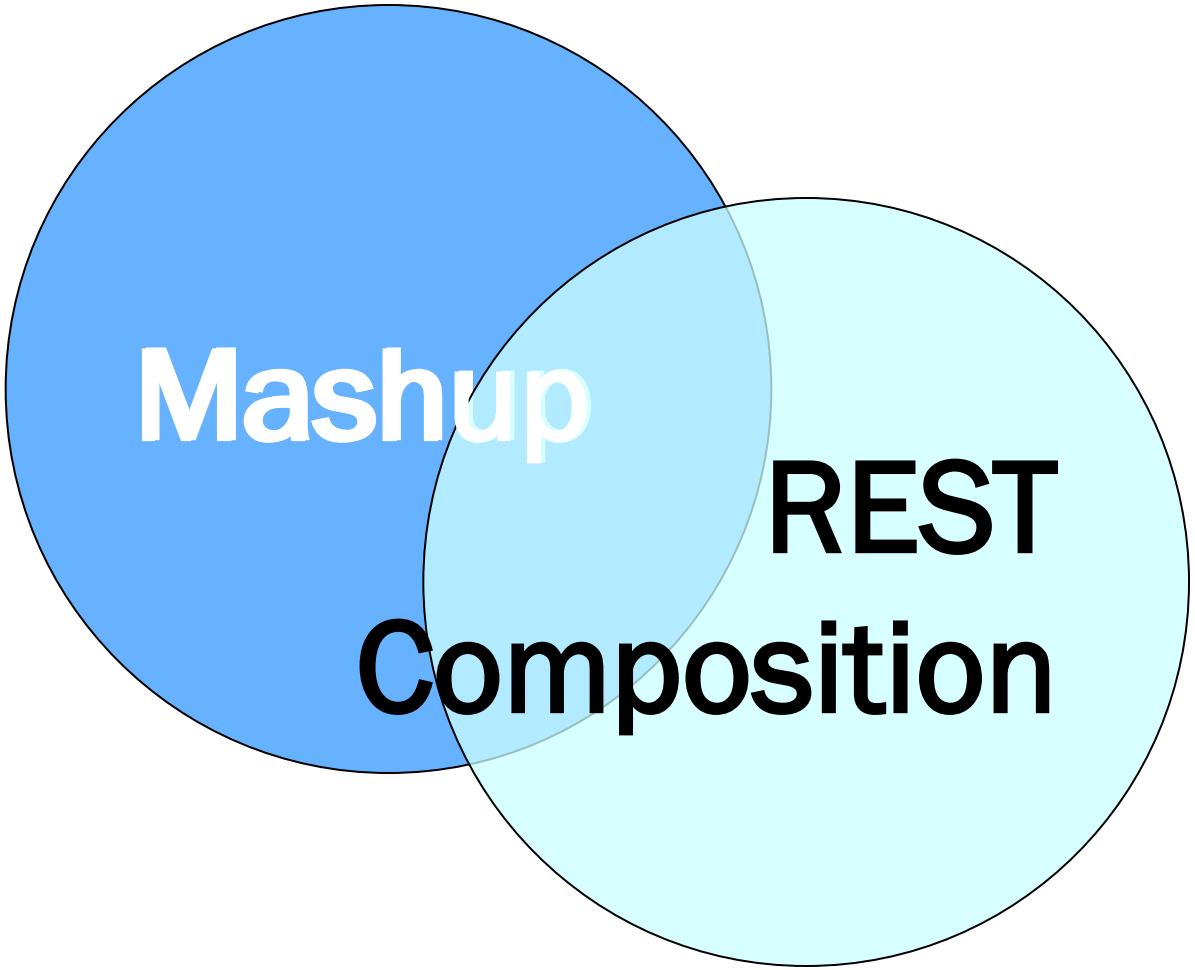


Data Flow


Write
output
input
Read
Data Flow (Copy)


DoodleMap
description = Where to g [...]
results = 10
name = CP
zip = 10001
time = 2000
query = Pizza
title = Pizza in N [...]
description
name
results
zip
query
title
time
Wait
GetYahooLocal
options
SYS.page
SYS.InputXml
PostDoodlePoll
ConvertY2D
SYS.Output
SYS.responseheaders
SYS.InputXml
ResponseHeaders
ConvertY2G
PollID
ParsePollID
SYS.Output
title
time
DoodleKey
locations
ShowGoogleMap
pollid
instance
pollId
SYS.ID
GetDoodlePoll
SYS.page
open
SYS.InputXml
ClosePoll
pollid
DoodleKey
CountParticipants
closed
SYS.body
PutDoodlePoll

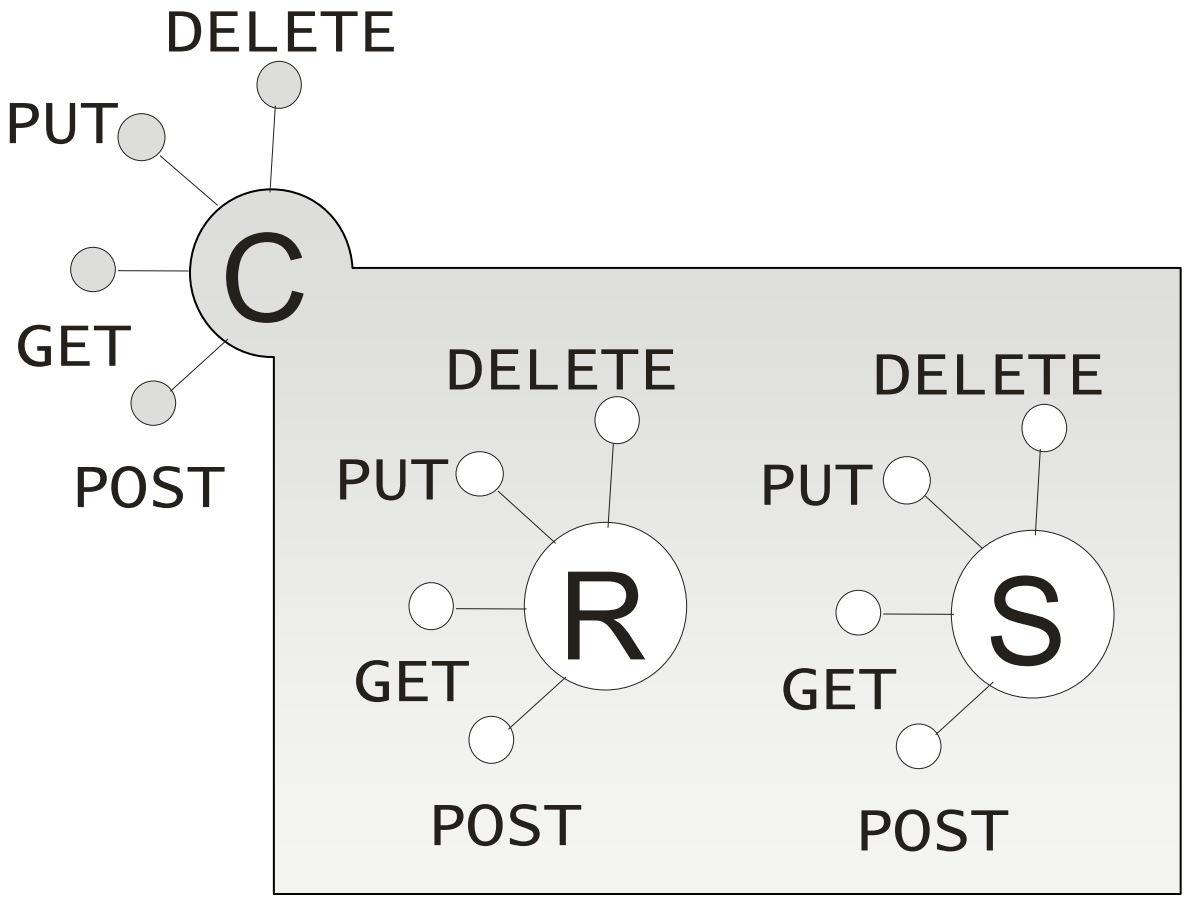
Was it just a mashup?



(It depends on the definition of Mashup)

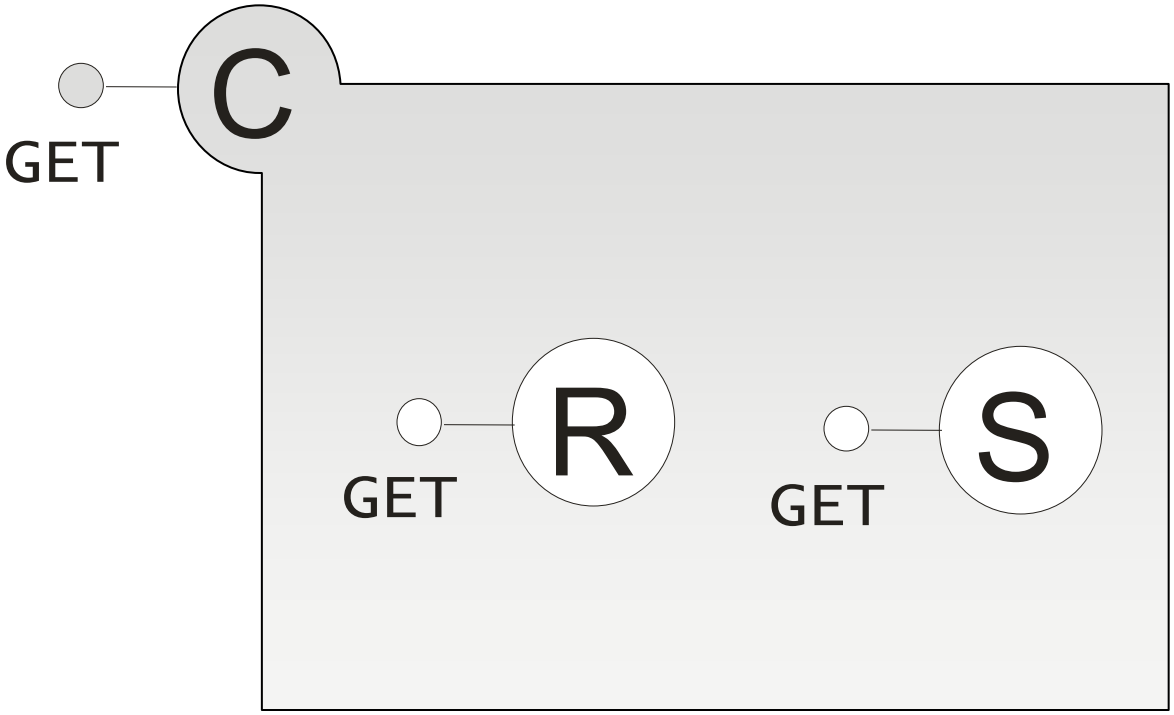
Moving state around

- Read-only vs. Read/Write



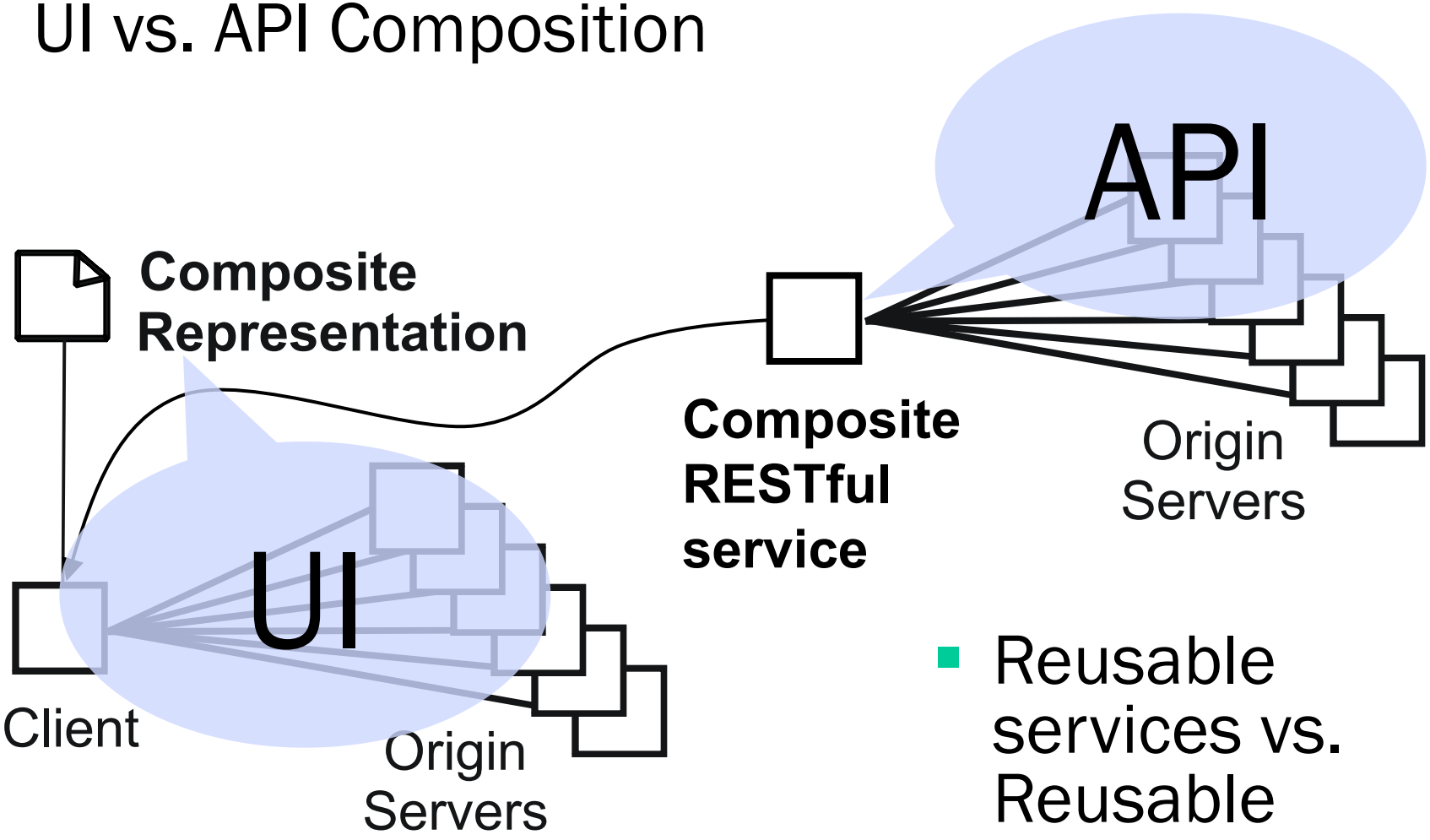
Simply aggregating data

- Read-only vs. Read/write



Is your composition reusable?

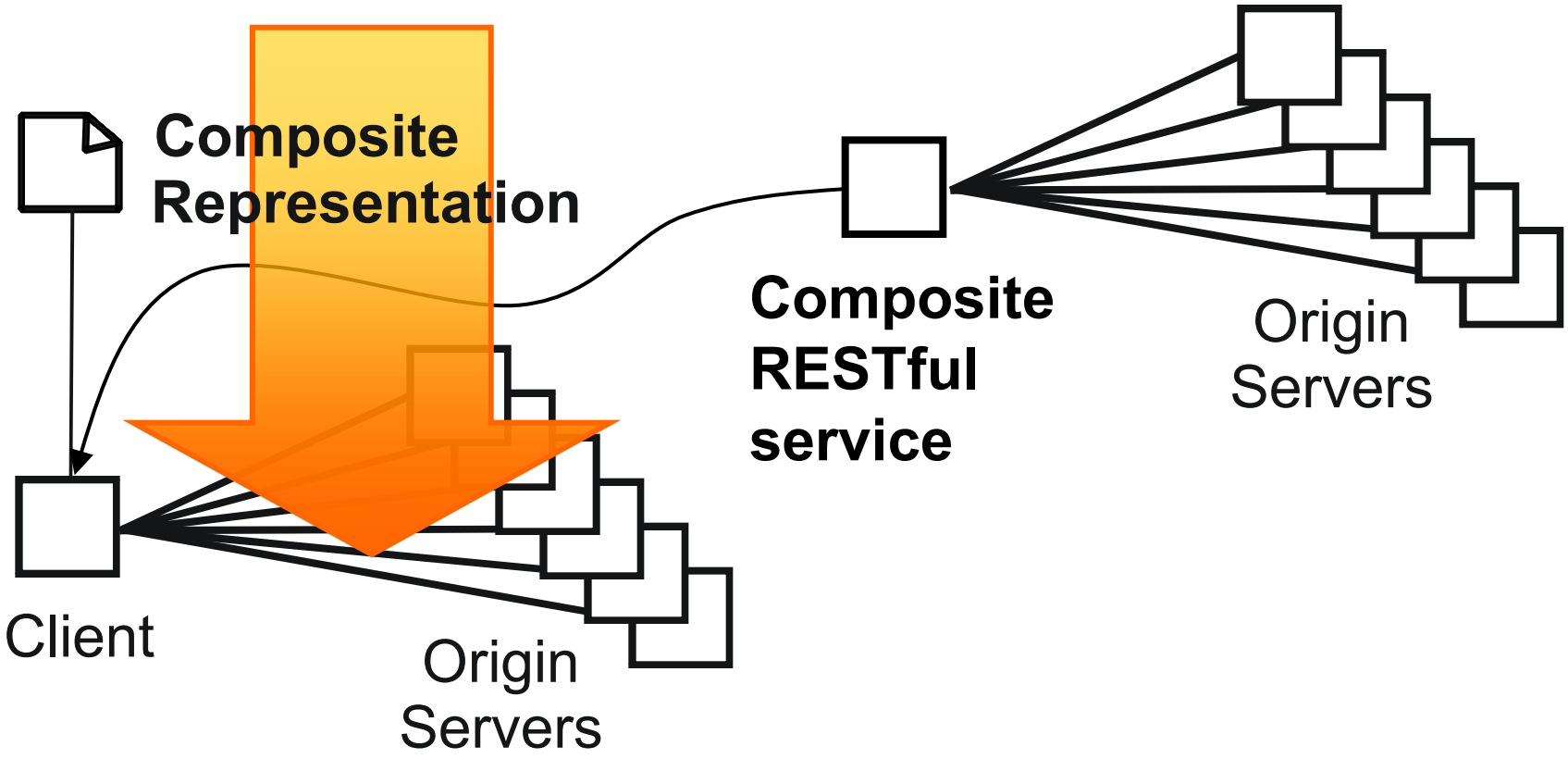
- UI vs. API Composition



- Reusable services vs. Reusable Widgets

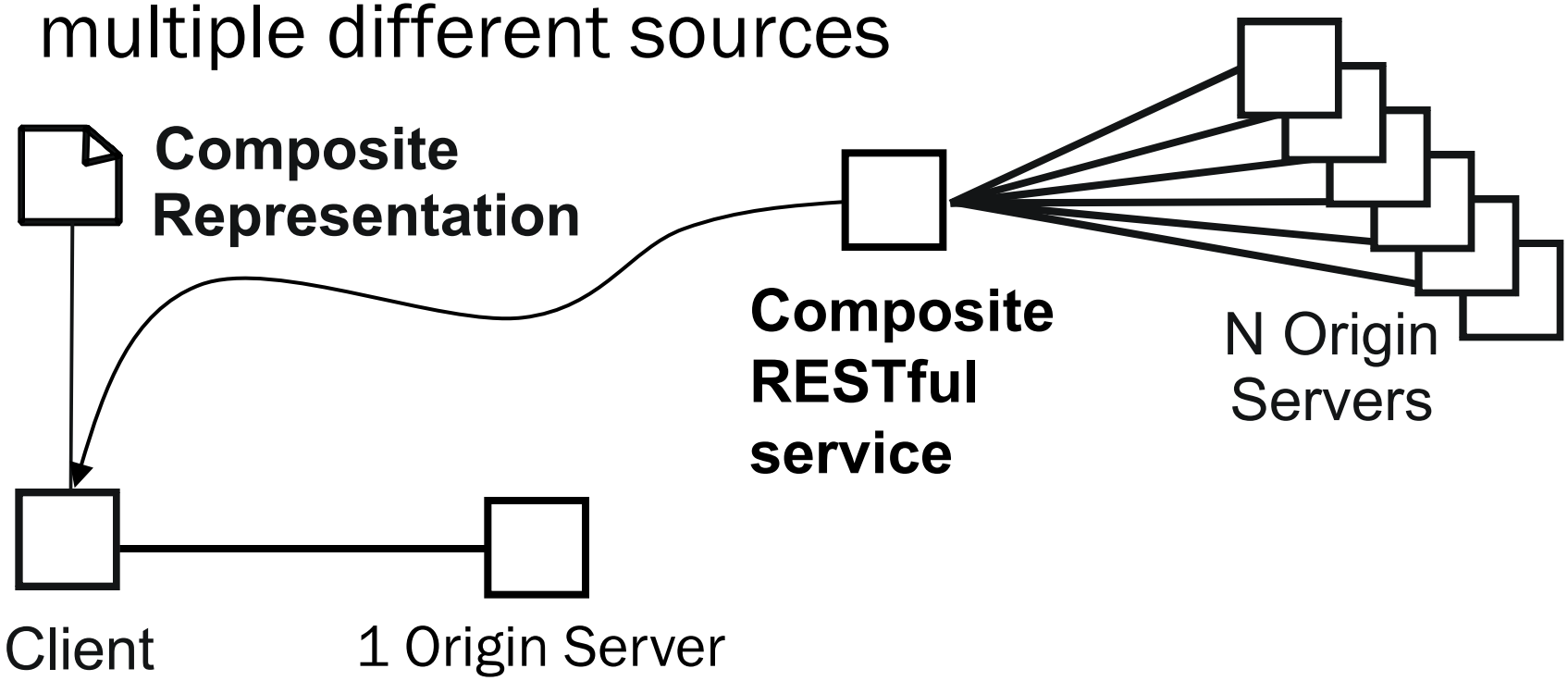
Single-Origin Sandbox

- Can you always do this from a web browser?



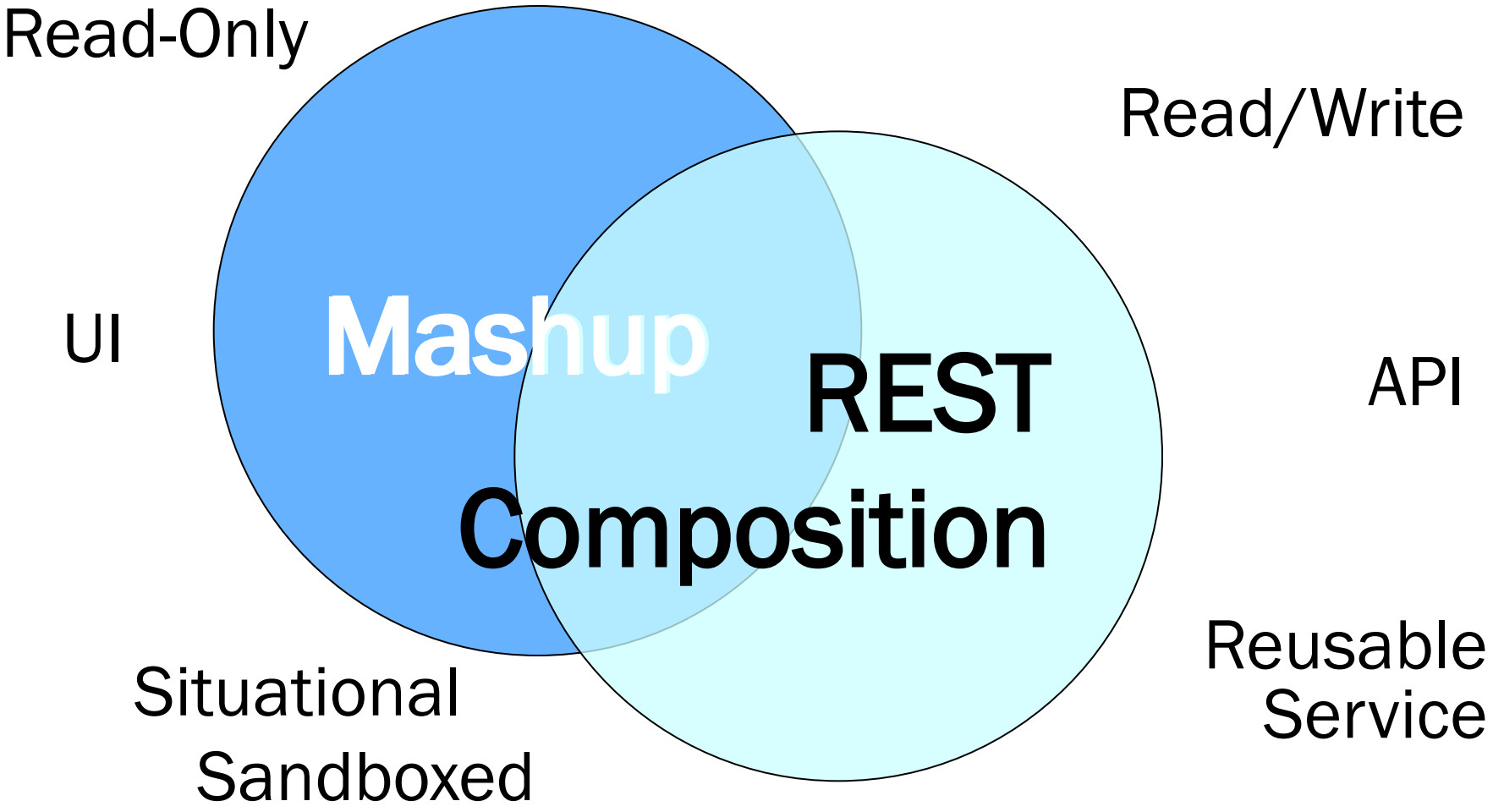
Single-Origin Sandbox

- Security Policies on the client may not always allow it to aggregate data from multiple different sources



- This will change very soon with HTML5

Complementary



Conclusions

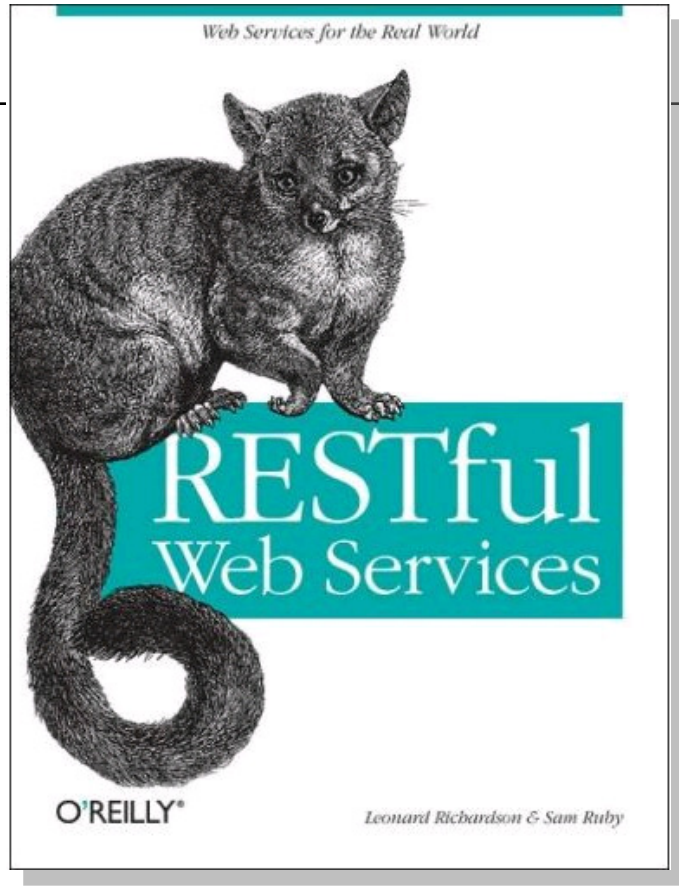
- REST brings a new perspective and new problems to service composition
- RESTful services can be composed on the server by defining composite resources and on the client with composite representations
- Composing RESTful services helps to put the integration logic of a mashup into a reusable service API and keep it separate from its UI made out of reusable widgets
- RESTful Web service composition is different than mashups, but both can be built using BPM tools like JOpera
- GET http://www.jopera.org/

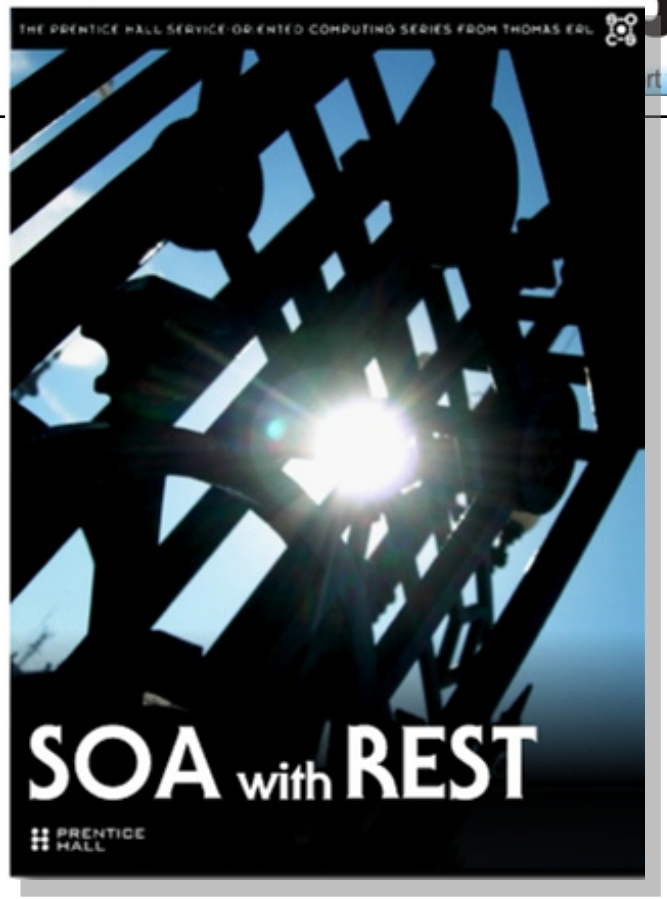
References

- Roy Fielding, Architectural Styles and the Design of Network-based Software Architectures, PhD Thesis, University of California, Irvine, 2000
- Leonard Richardson, Sam Ruby, **RESTful Web Services**, O'Reilly, May 2007
- Jim Webber, Savas Parastatidis, Ian Robinson, **REST in Practice: Hypermedia and Systems Architecture**, O'Reilly, 2010
- Subbu Allamaraju, **RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity**, O'Reilly, 2010
- Stevan Tilkov, **HTTP und REST**, dpunkt Verlag, 2009, http://rest-http.info/
- Thomas Erl, Raj Balasubramanians, Cesare Pautasso, Benjamin Carlyle, **SOA with REST**, Prentice Hall, end of 2010
- Martin Fowler, **Richardson Maturity Model: steps toward the glory of REST,** http://martinfowler.com/articles/richardsonMaturityModel.html

Self-References

- Cesare Pautasso, Olaf Zimmermann, Frank Leymann, RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision, Proc. of the 17th International World Wide Web Conference (WWW2008), Bejing, China, April 2008.
- Cesare Pautasso and Erik Wilde. Why is the Web Loosely Coupled? A Multi-Faceted Metric for Service Design, Proc of the 18th International World Wide Web Conference (WWW2009), Madrid, Spain, April 2009.
- Cesare Pautasso, BPEL for REST, Proc. of the 6th International Conference on Business Process Management (BPM 2008), Milan, Italy, September 2008.
- Cesare Pautasso, RESTful Web Service Composition with JOpera, Proc. Of the International Conference on Software Composition (SC 2009), Zurich, Switzerland, July 2009.
- Cesare Pautasso, Gustavo Alonso: **From Web Service Composition to Megaprogramming** In: Proceedings of the 5th VLDB Workshop on Technologies for E-Services (TES-04), Toronto, Canada, August 2004.





Leonard Richardson, Sam Ruby,
RESTful Web Services,
O'Reilly, May 2007

Raj Balasubramanians, Benjamin Carlyle,Thomas Erl, Cesare Pautasso,
SOA with REST,
Prentice Hall, end of 2010


ECOWS'10
8th European Conference on Web Services
Ayia Napa, Cyprus
December 1-3, 2010
http://www.cs.ucy.ac.cy/ecows10
http://twitter.com/ecows2010
Abstract Submission: Friday, July 16, 2010