Università della Svizzera italiana

Faculty of Informatics

SOA 2nd international symposium

Techniques for Composing REST services

Cesare Pautasso
Faculty of Informatics
University of Lugano, Switzerland

c.pautasso@ieee.org
http://www.pautasso.info

Abstract

- Novel trends in Web services technology challenge the assumptions made by current standards for process-based service composition. For example, most existing RESTful Web service APIs (which do not rely on the Web Service Description Language), cannot natively be composed using the WS-BPEL language.
- In this talk we introduce the problem of composing RESTful services and compare it to Web 2.0 service mashups. We cover several real-world examples demonstrating how existing composition languages can be evolved to cope with REST. We conclude by showing that the uniform interface and hyper-linking capabilities of RESTful services provides an excellent abstraction for exposing in a controlled way the state of business process as a resource.

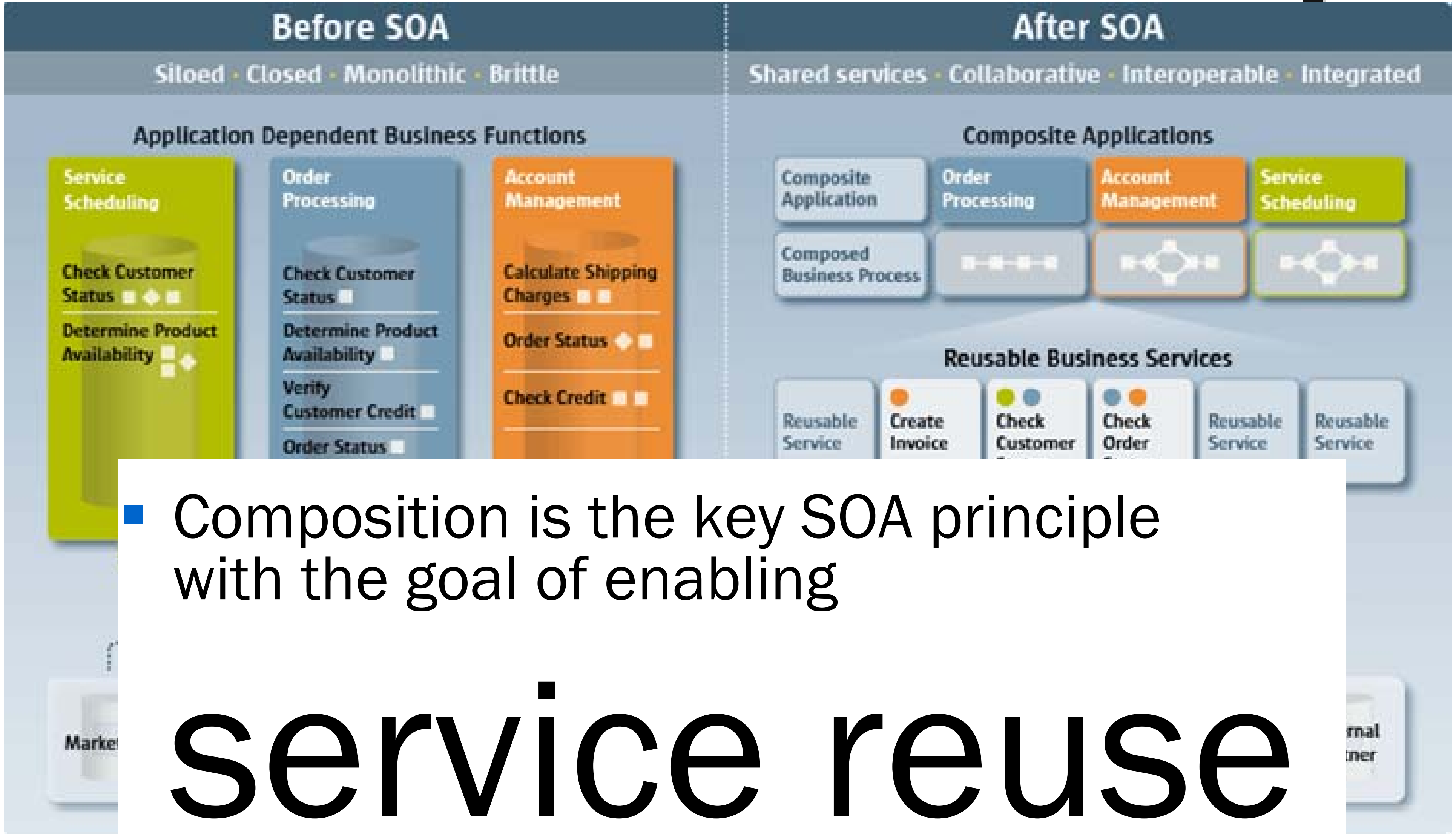
About Cesare Pautasso

- Assistant Professor at the Faculty of Informatics, University of Lugano, Switzerland (since Sept 2007)

  Research Projects:
  - SOSOA – Self Organizing Service Oriented Architectures
  - CLAVOS – Continuous Lifelong Analysis and Verification of Open Services
  - BPEL for REST
- Researcher at IBM Zurich Research Lab (2007)
- Post Doc at ETH Zürich
  - Software: JOpera: Process Support for more than Web services http://www.jopera.org/
- Ph.D. at ETH Zürich, Switzerland (2004)

- Representations: http://www.pautasso.info/ (Web) http://twitter.com/pautasso/ (Twitter Feed)

Why Composition?



- Composition is the key SOA principle with the goal of enabling

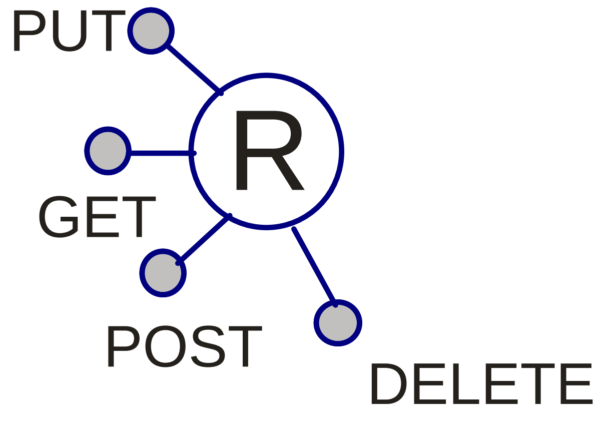
service reuse

REST and Reuse

- Uniform Interface (Reuse Contract)
- Status Codes (Reuse Metadata)
- Representations (Reuse Media Types)
- Middleware (Reuse caching, security, load balancing, proxies components)

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

REST and Reuse

- Uniform Interface (Reuse Contract)
- Status Codes (Reuse Metadata)
- Representations (Reuse Media Types)
- Middleware (Reuse caching, security, load balancing, proxies components)

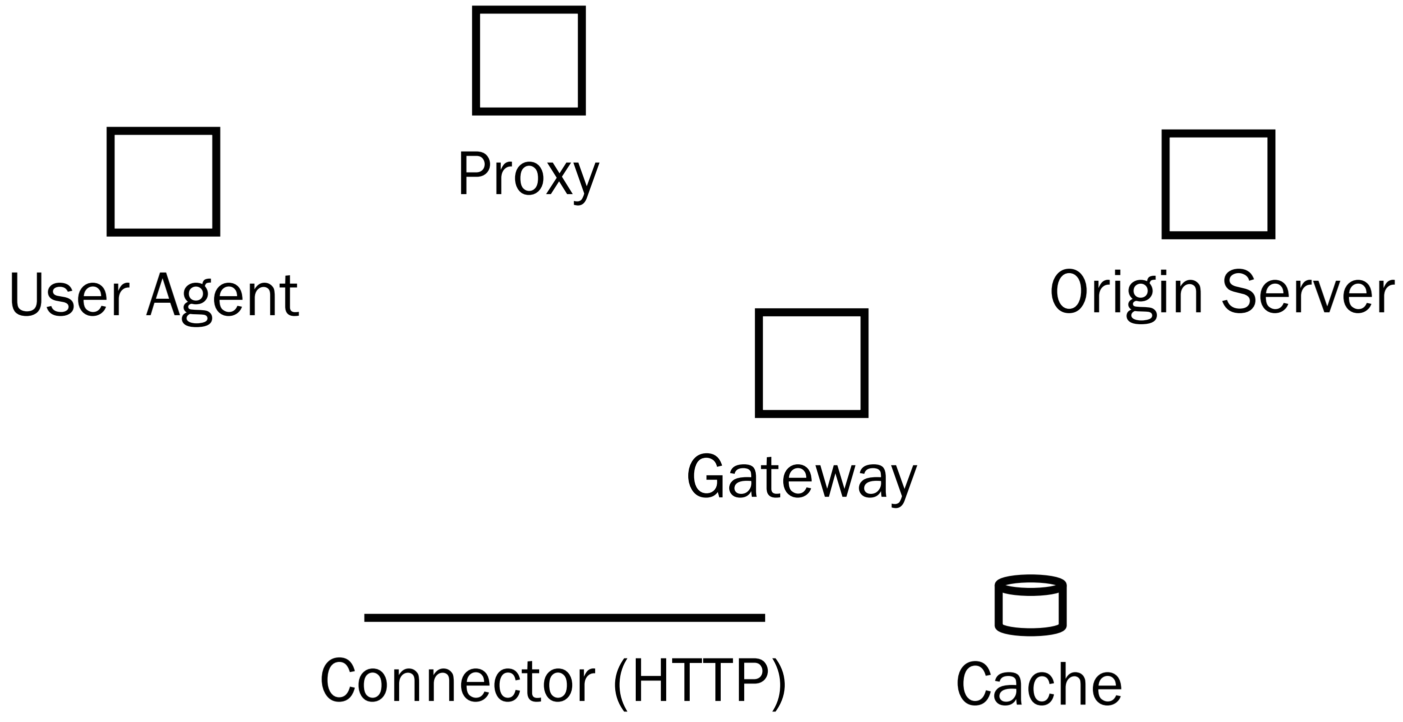
- **Yes, but what about reusing entire RESTful services?**

RESTful Composition Techniques

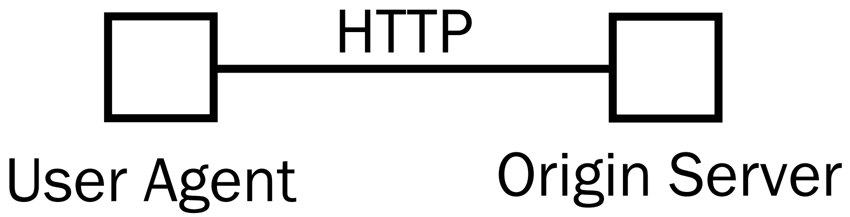
1. Defining RESTful service composition
2. Example: DoodleMap
3. What about mashups?
4. BPM and REST

REST Architectural Elements

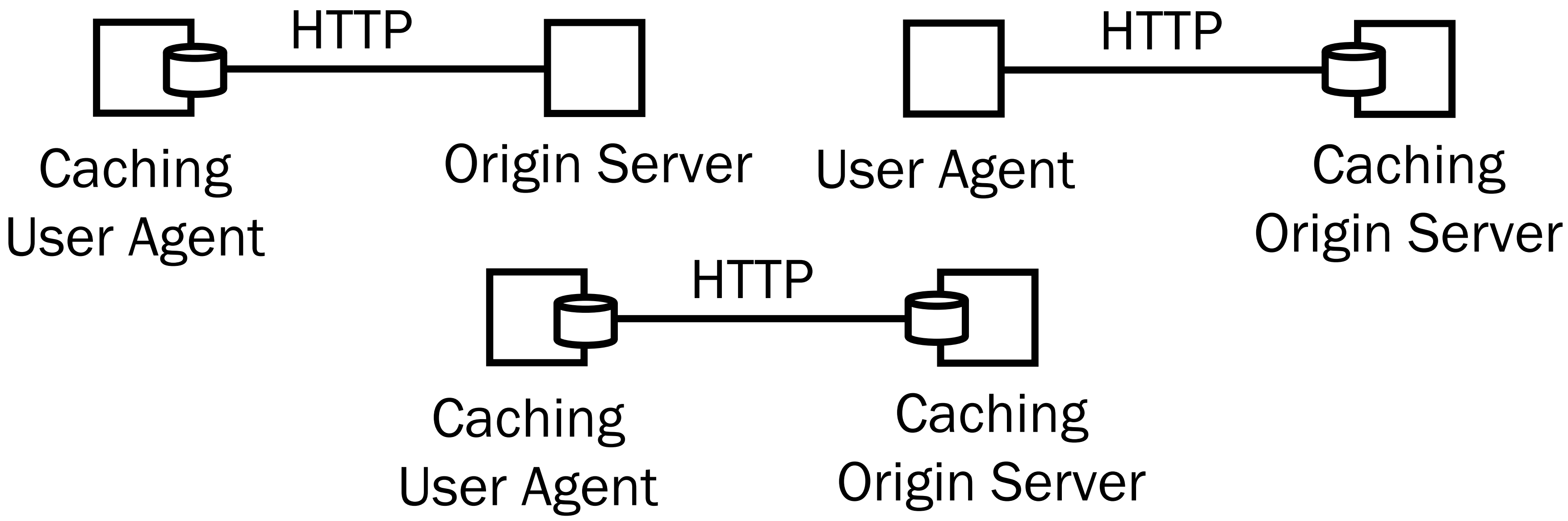
Client/Server Layered Stateless Communication Cache



Basic Setup

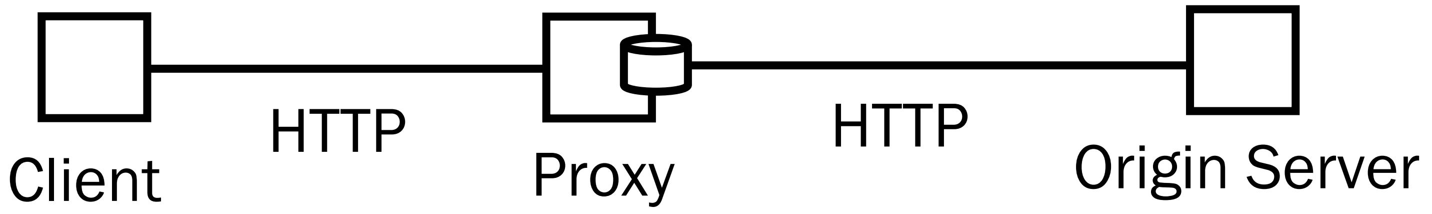


Adding Caching

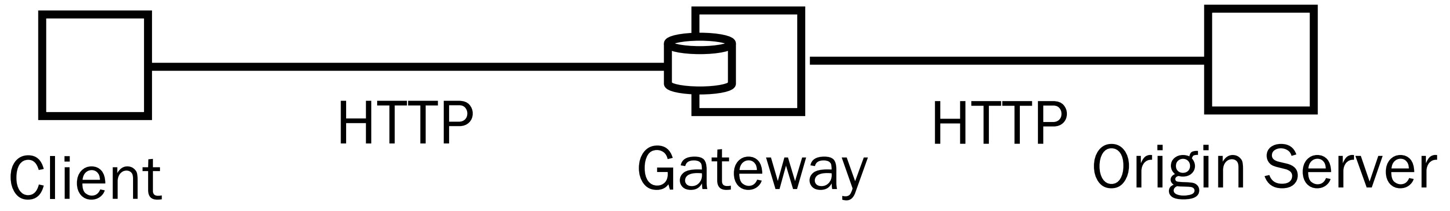


Proxy or Gateway?

Intermediaries forward (and may translate) requests and responses

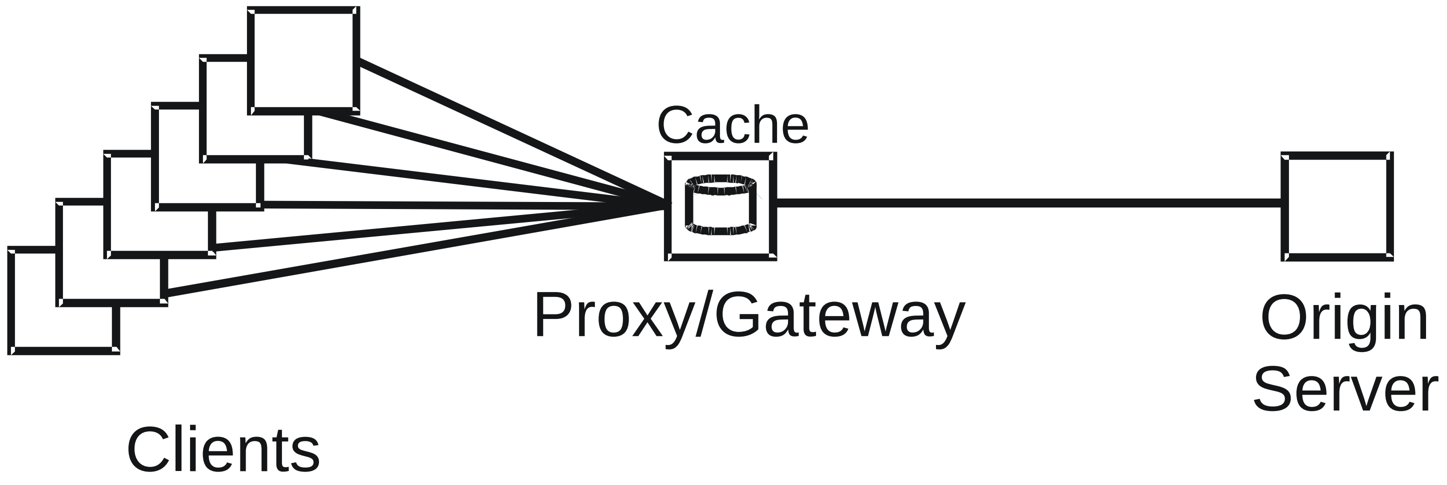


A proxy is chosen by the Client (for caching, or access control)



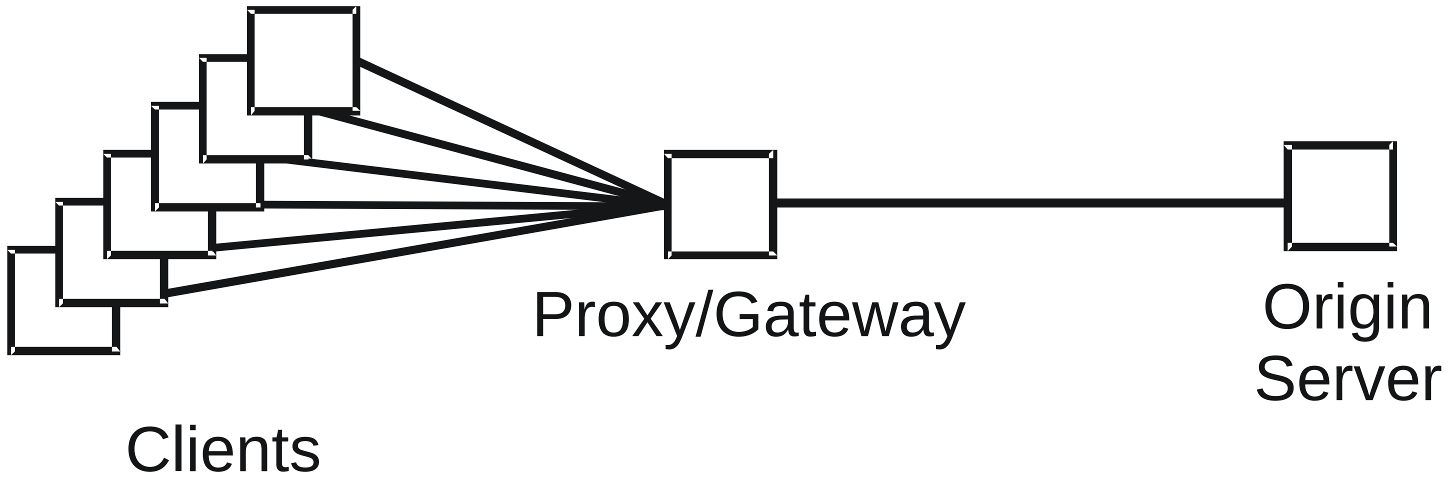
The use of a gateway (or reverse proxy) is imposed by the server

REST Middleware for 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

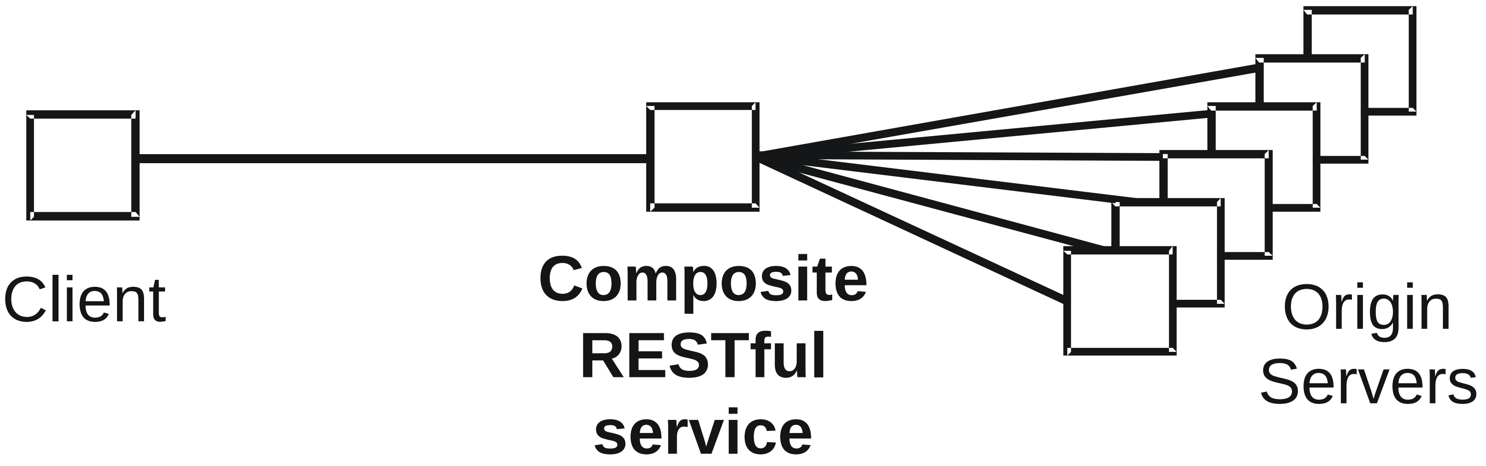
REST Middleware for Composition



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

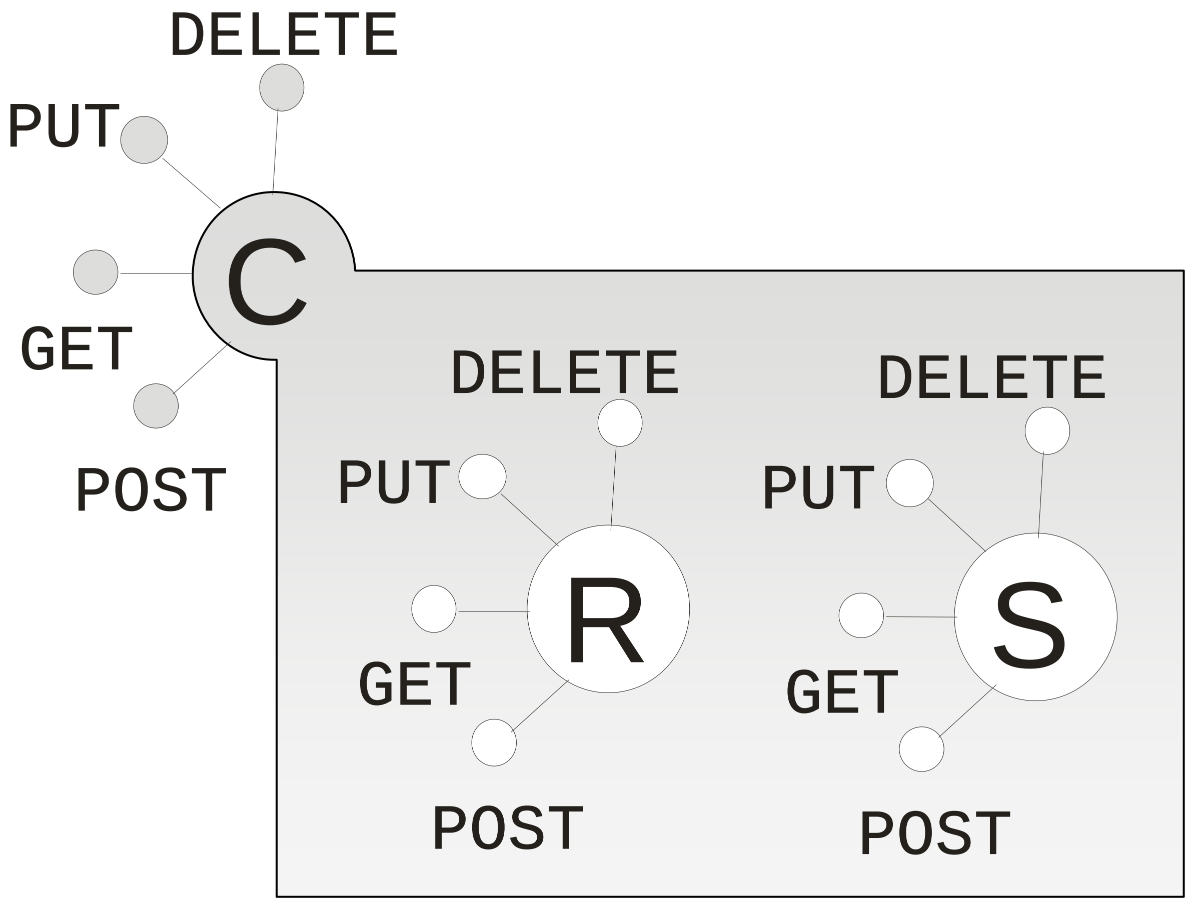
REST Middleware for Composition

Università della Svizzera italiana



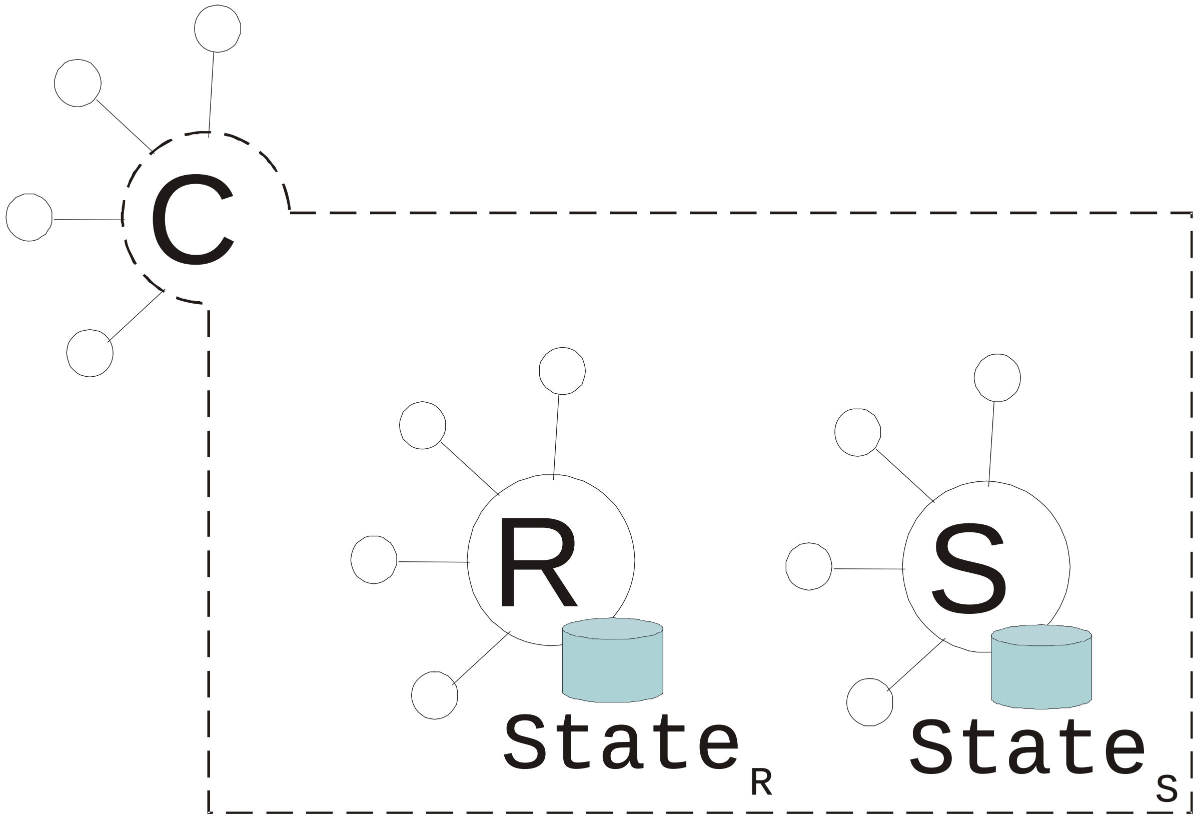
- The "proxy" intermediate element which aggregates the resources provided by multiple servers plays the role of the composition controller of a composite RESTful service

Composite Resources



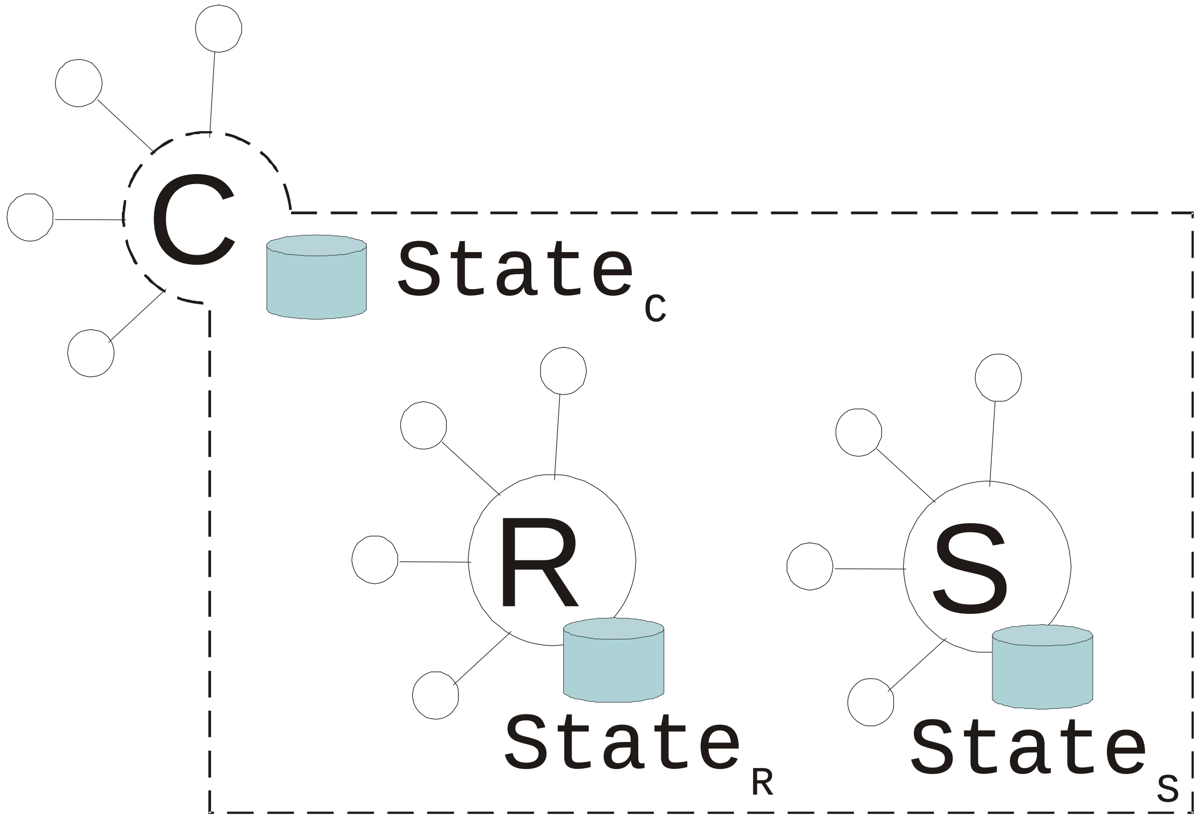
Composite Resources

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



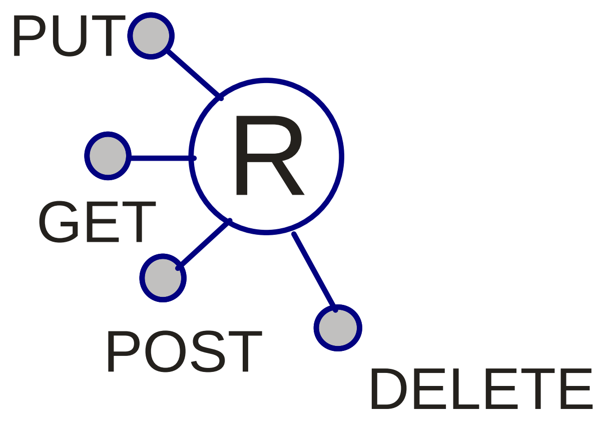
Composite Resources

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

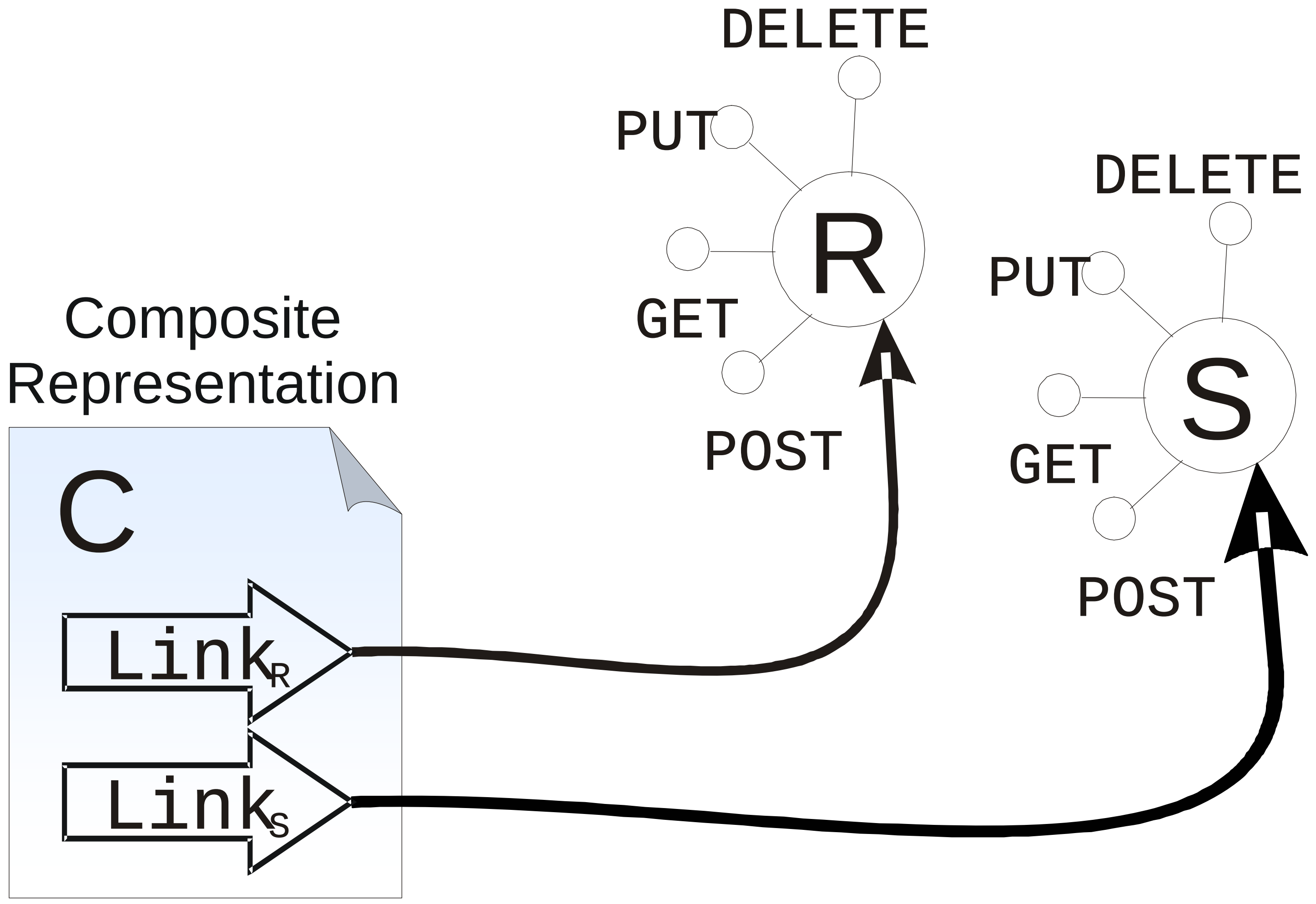


Enter HATEOAS

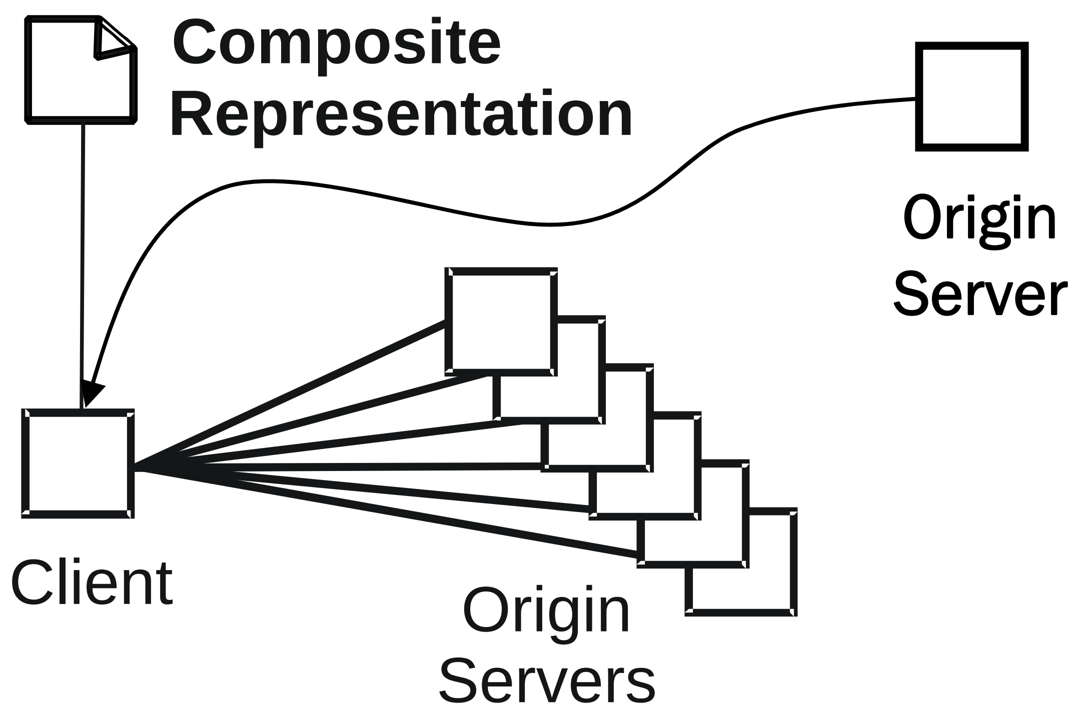
- Web Services expose their data and functionality trough **resources** identified by **URI**
- **Uniform Interface** Principle: Clients interact with resources through a fix set of verbs. Example HTTP: GET (read), POST (create), PUT (update), DELETE
- **Multiple representations** for the same resource
- **Hyperlinks** model resource relationships and valid state transitions for dynamic protocol description and discovery



Composite Representations

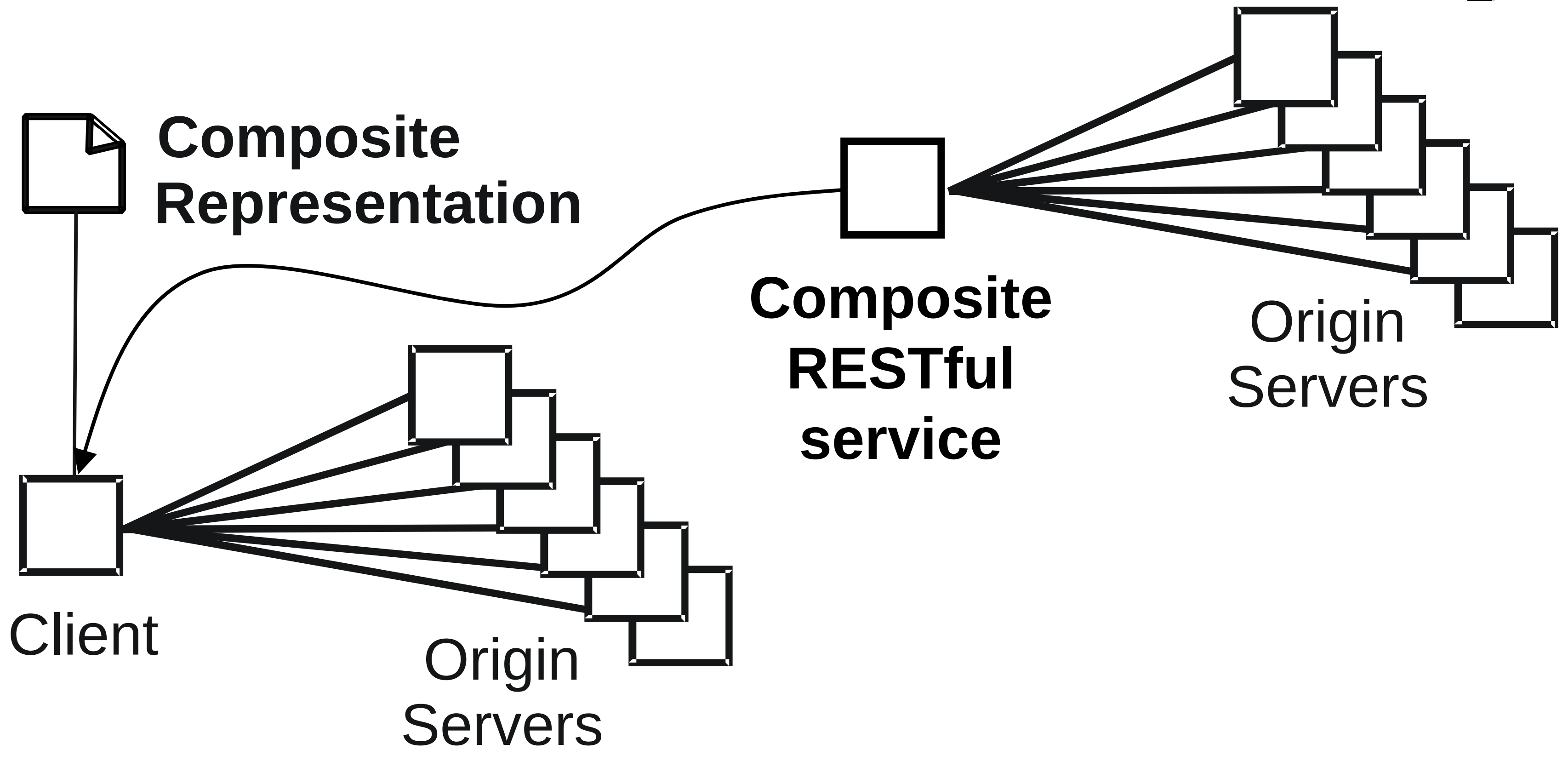


Composite Representation Pattern



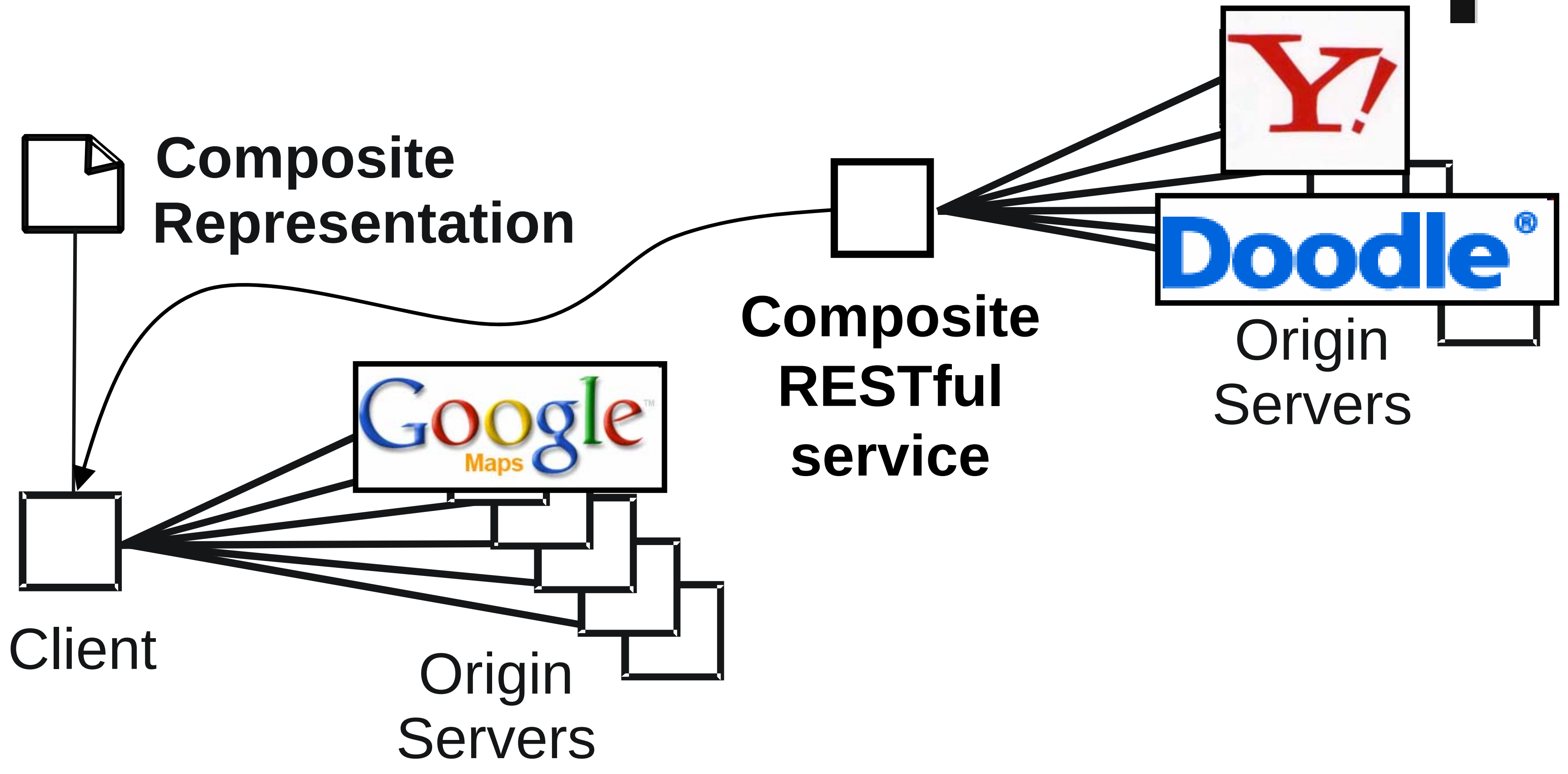
- A composite representation is interpreted by the client that follows its hyperlinks and aggregates the state of the referenced component resources

Bringing it all together



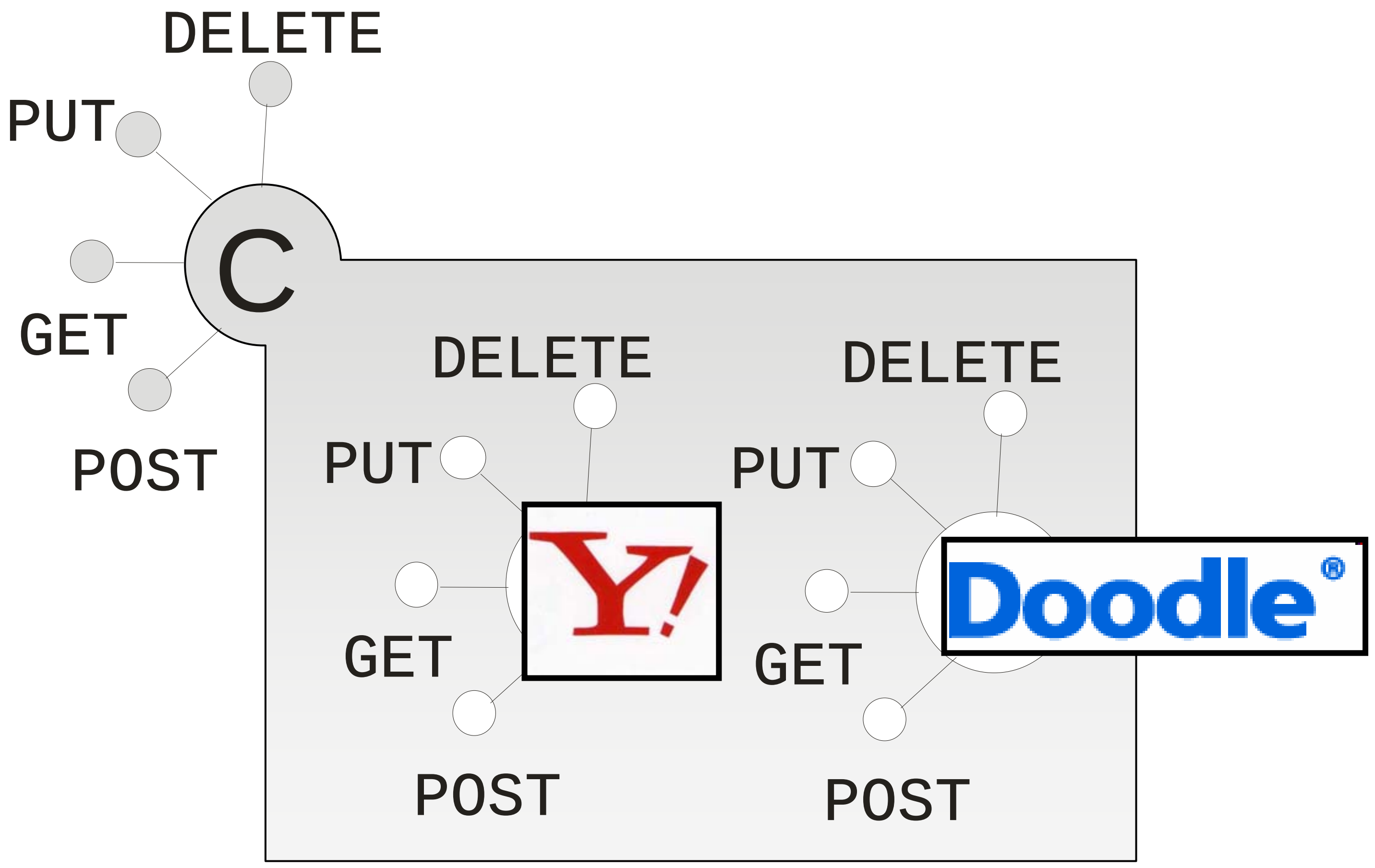
- A composite representation can be produced by a composite service too

Doodle Map Example

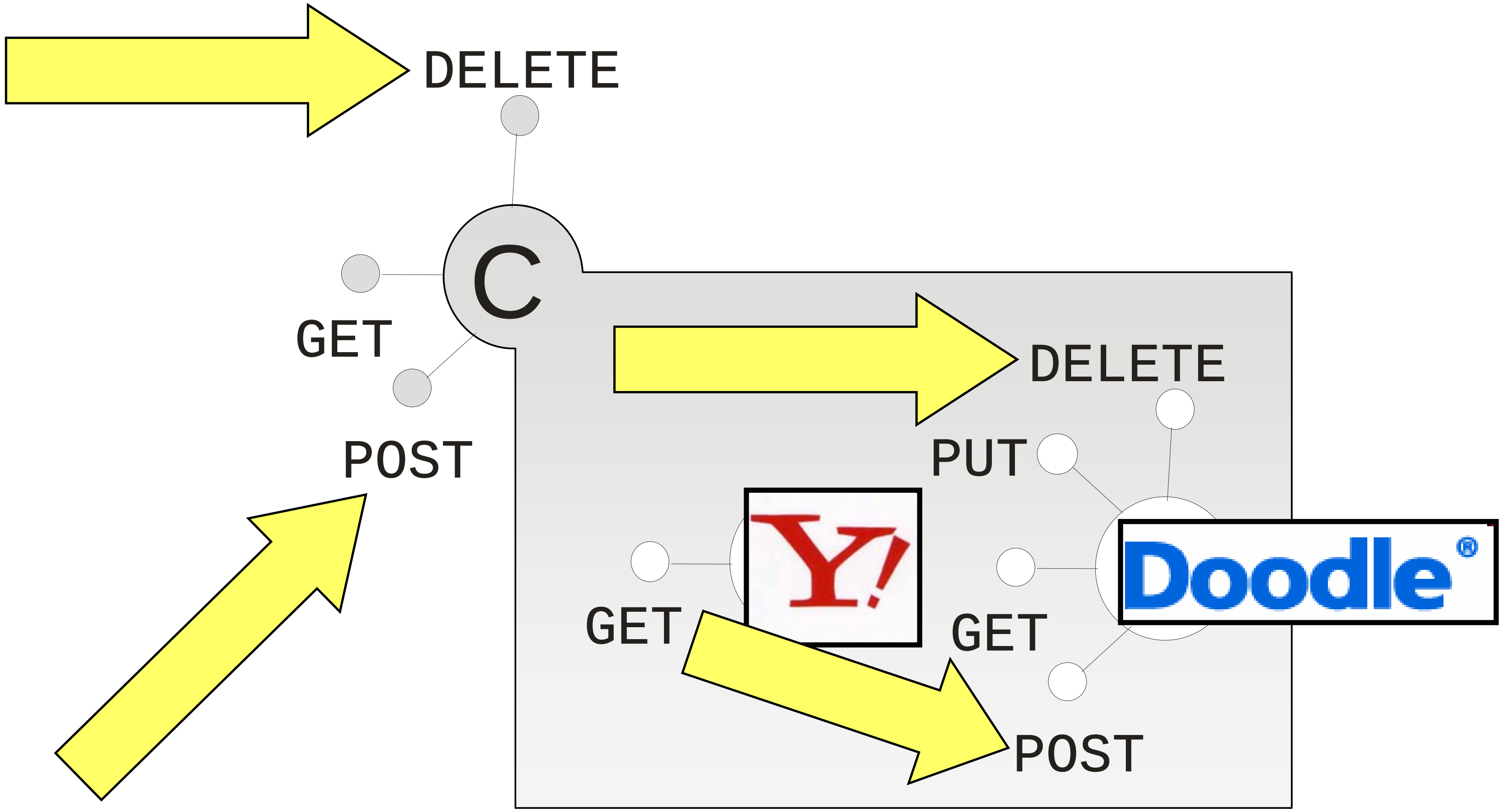


- Vote on a meeting place based on its geographic location

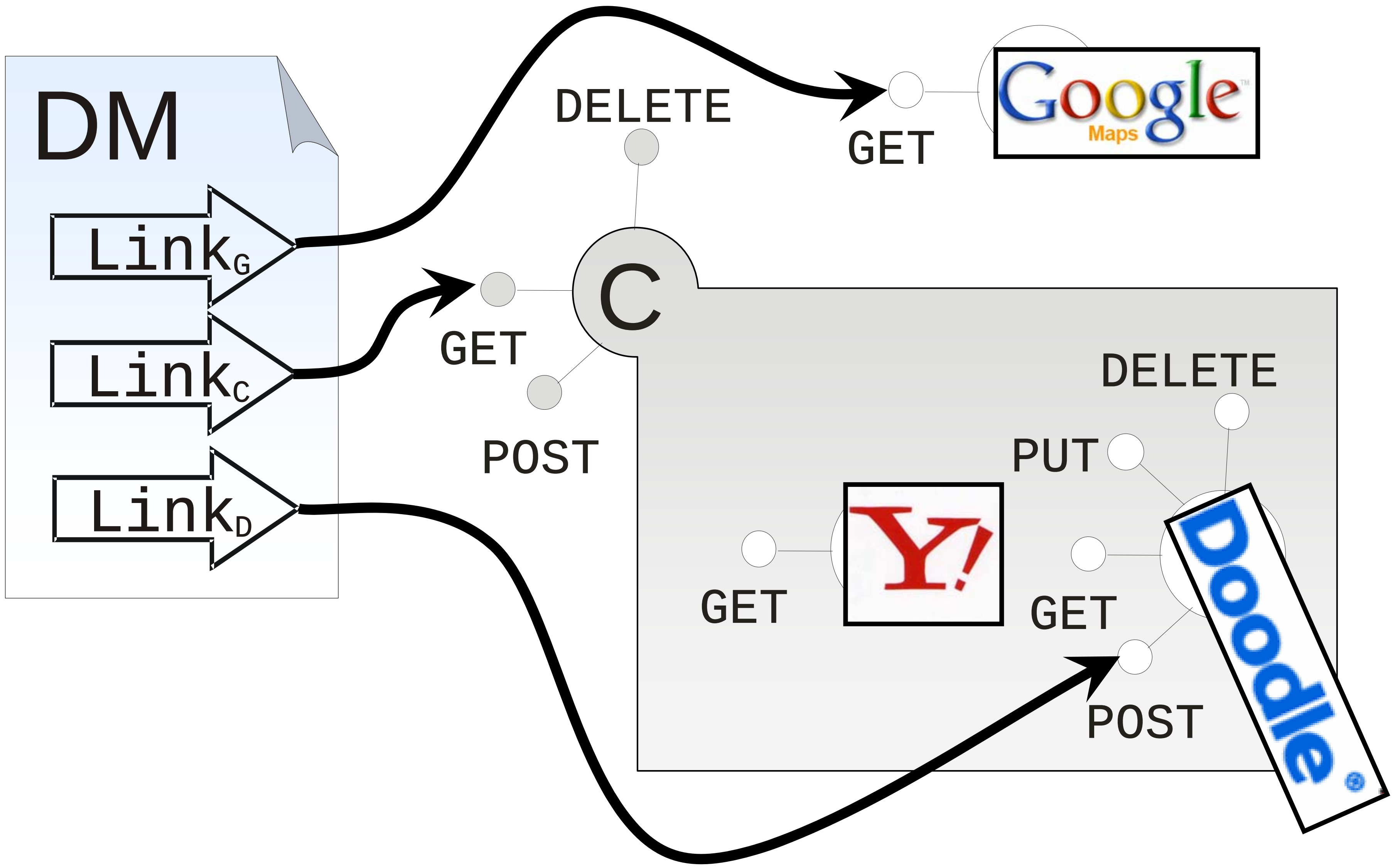
1. Composite Resource

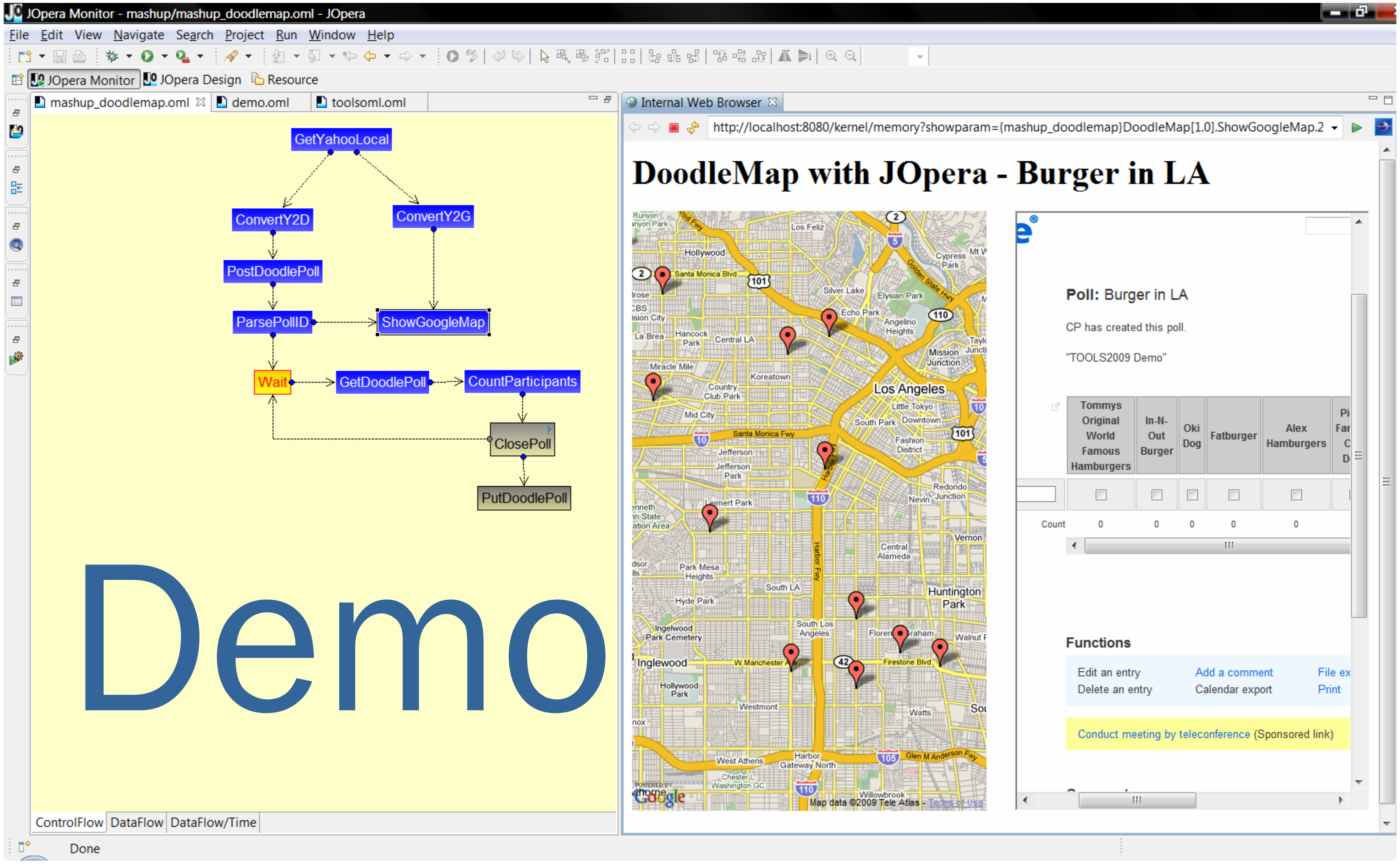


1. Composite Resource



2. Composite Representation





Demo

DoodleMap with JOpera - Burger in LA

Poll: Burger in LA

CP has created this poll.

"TOOLS2009 Demo"

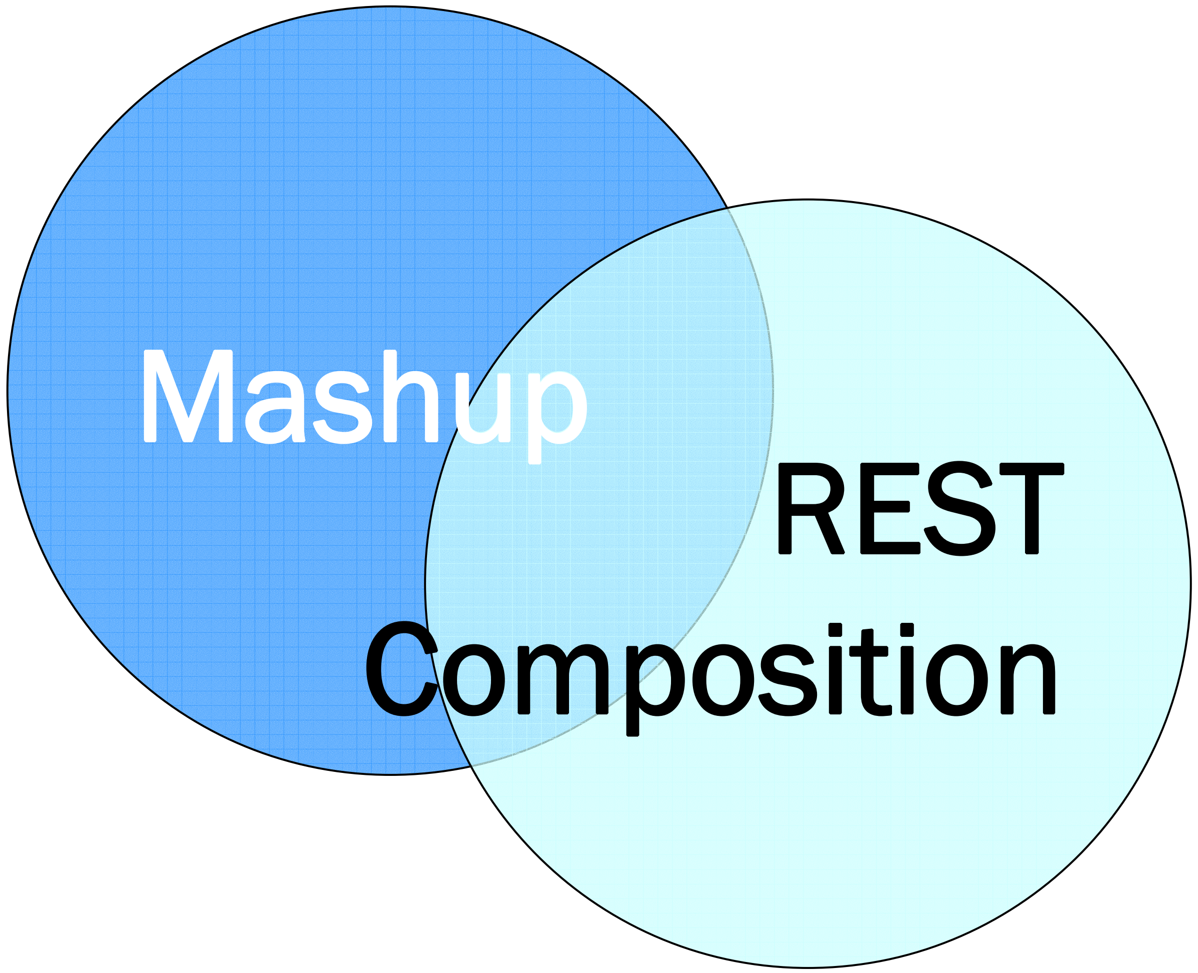
	Tommys Original World Famous Hamburgers	In-N-Out Burger	Oki Dog	Fatburger	Alex Hamburgers	Pi Far C D
	☐	☐	☐	☐	☐	
Count	0	0	0	0	0	

Functions

Edit an entry	Add a comment	File ex
Delete an entry	Calendar export	Print

Conduct meeting by teleconference (Sponsored link)

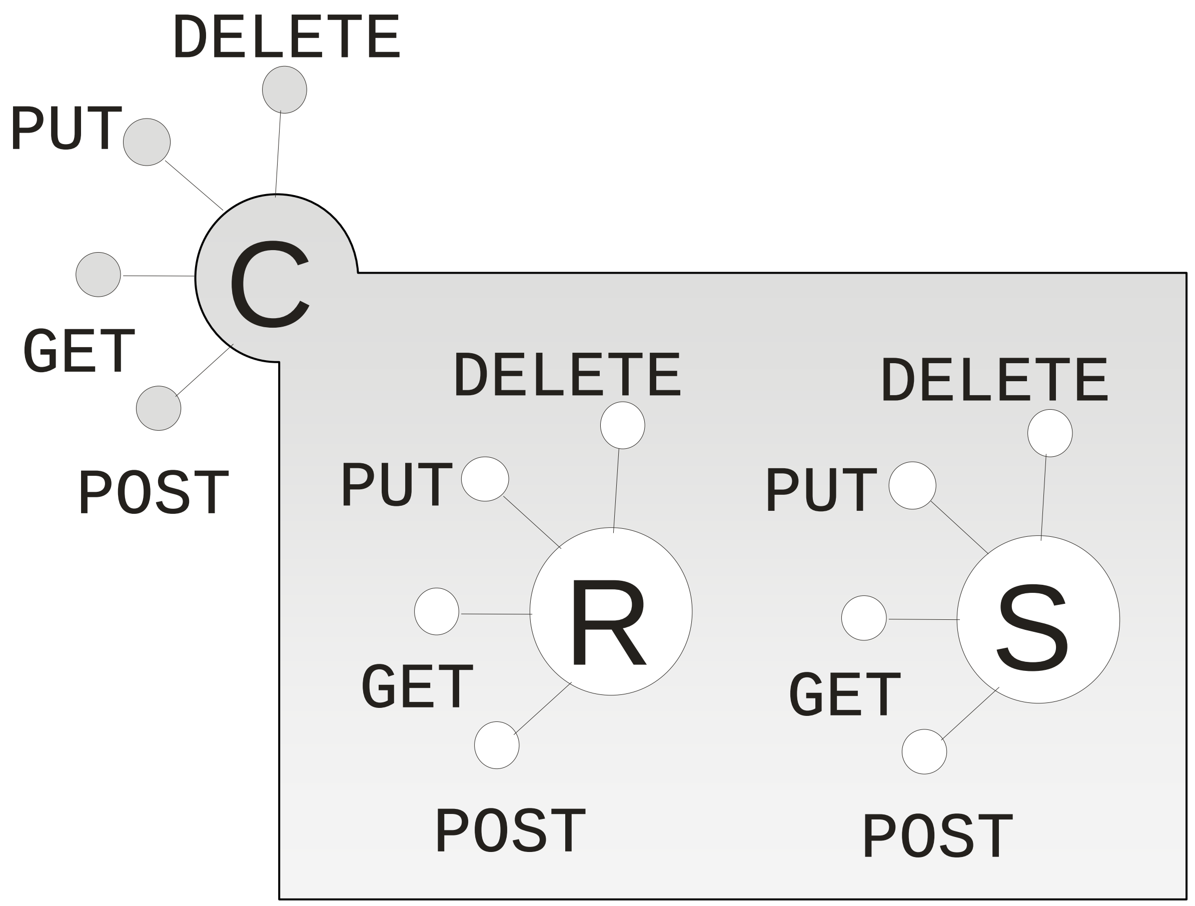
Was it just a Mashup?



(It depends on the definition of Mashup)

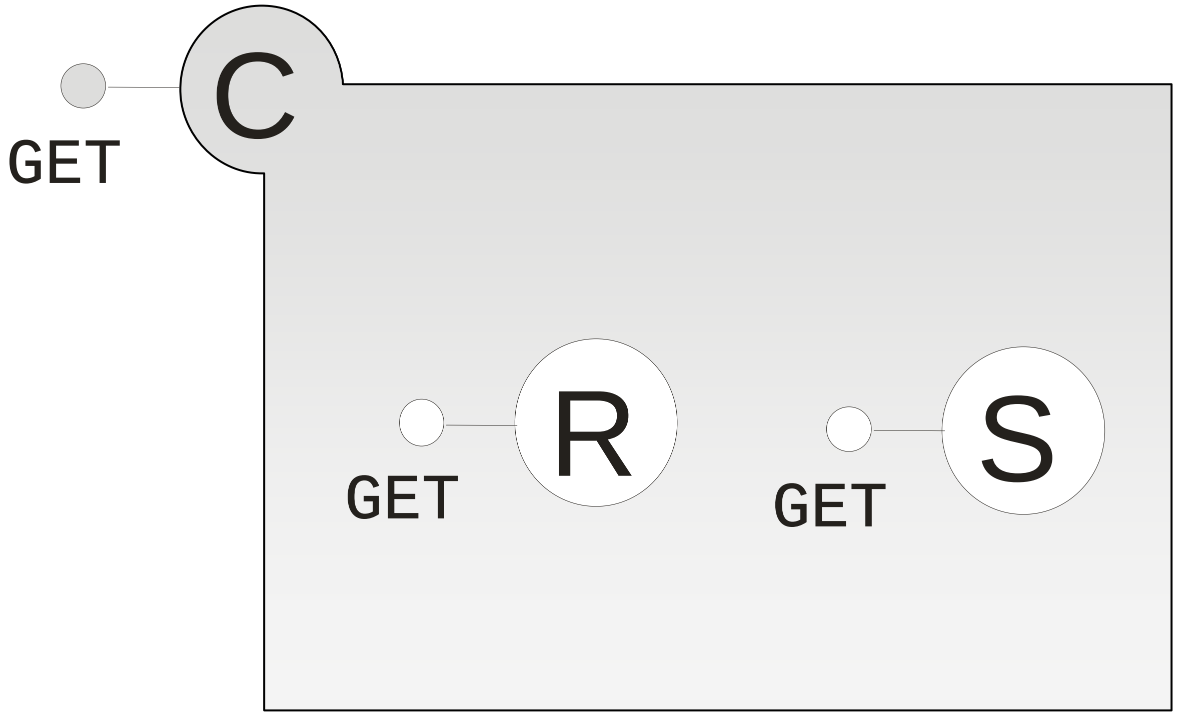
Moving state around

- Read-only vs. <u>Read/Write</u>



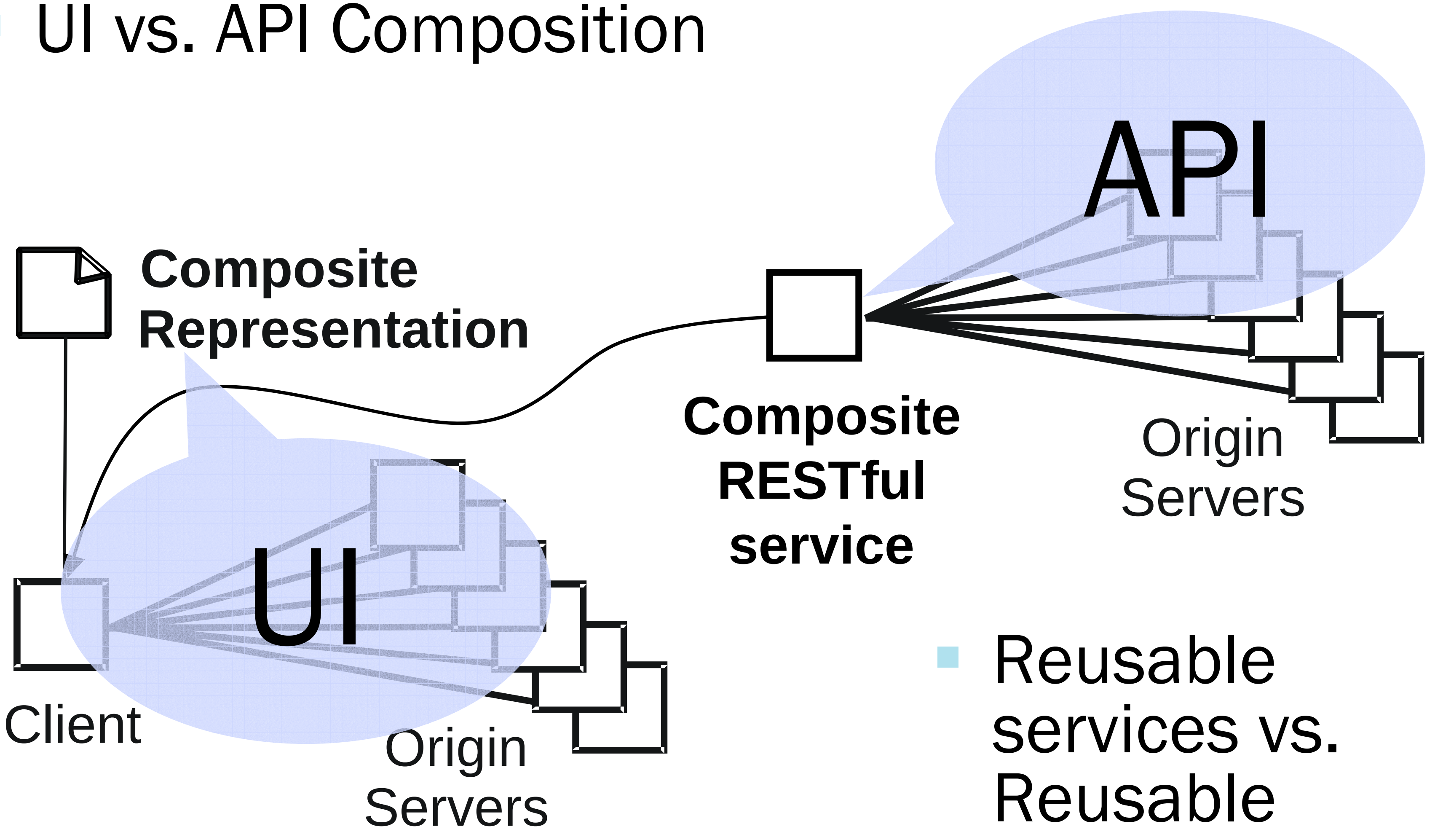
Simply aggregating data (feeds)

- <u>Read-only</u> vs. Read/write



Is your composition reusable?

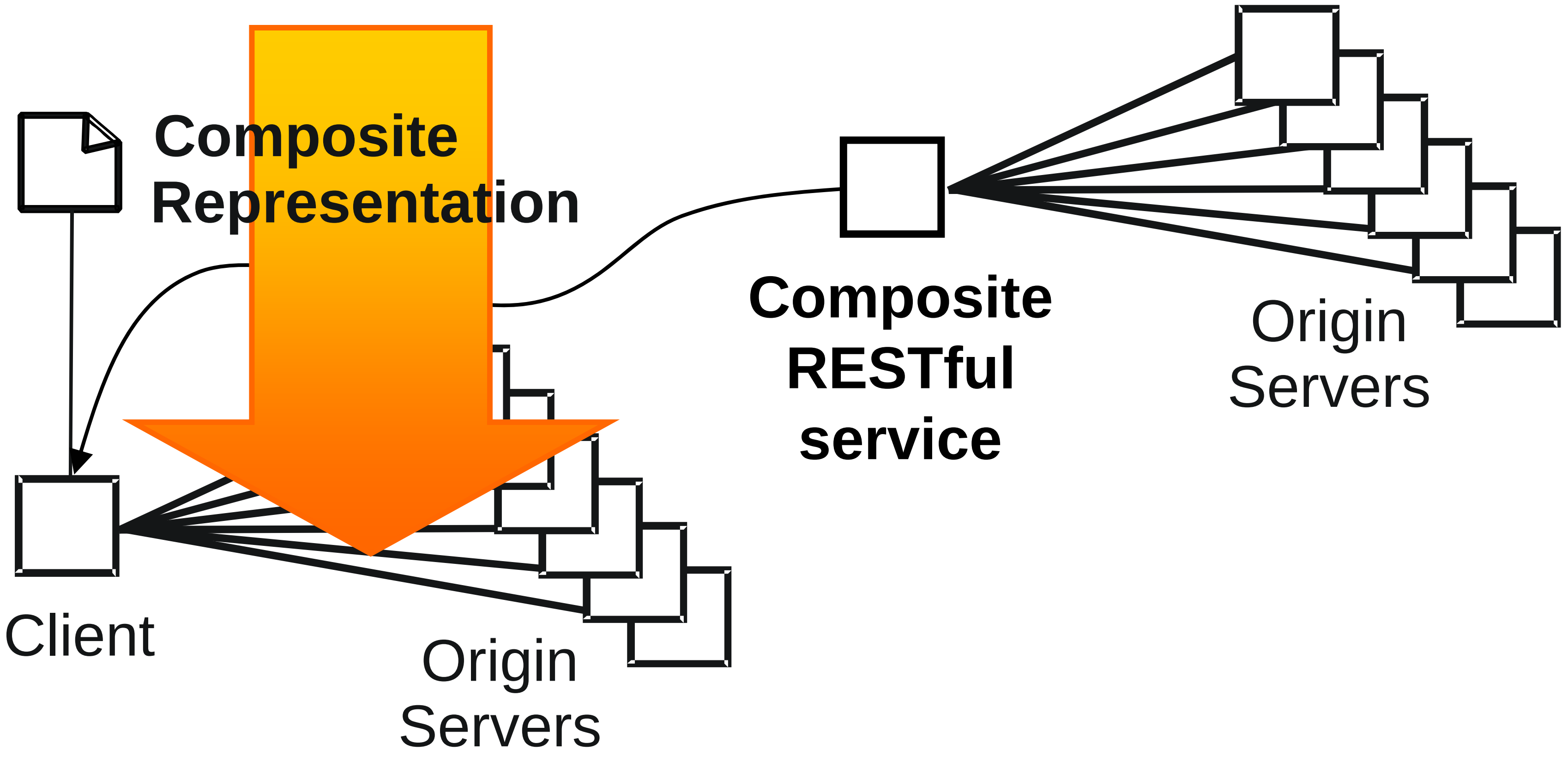
- UI vs. API Composition



- Reusable services vs. Reusable Widgets

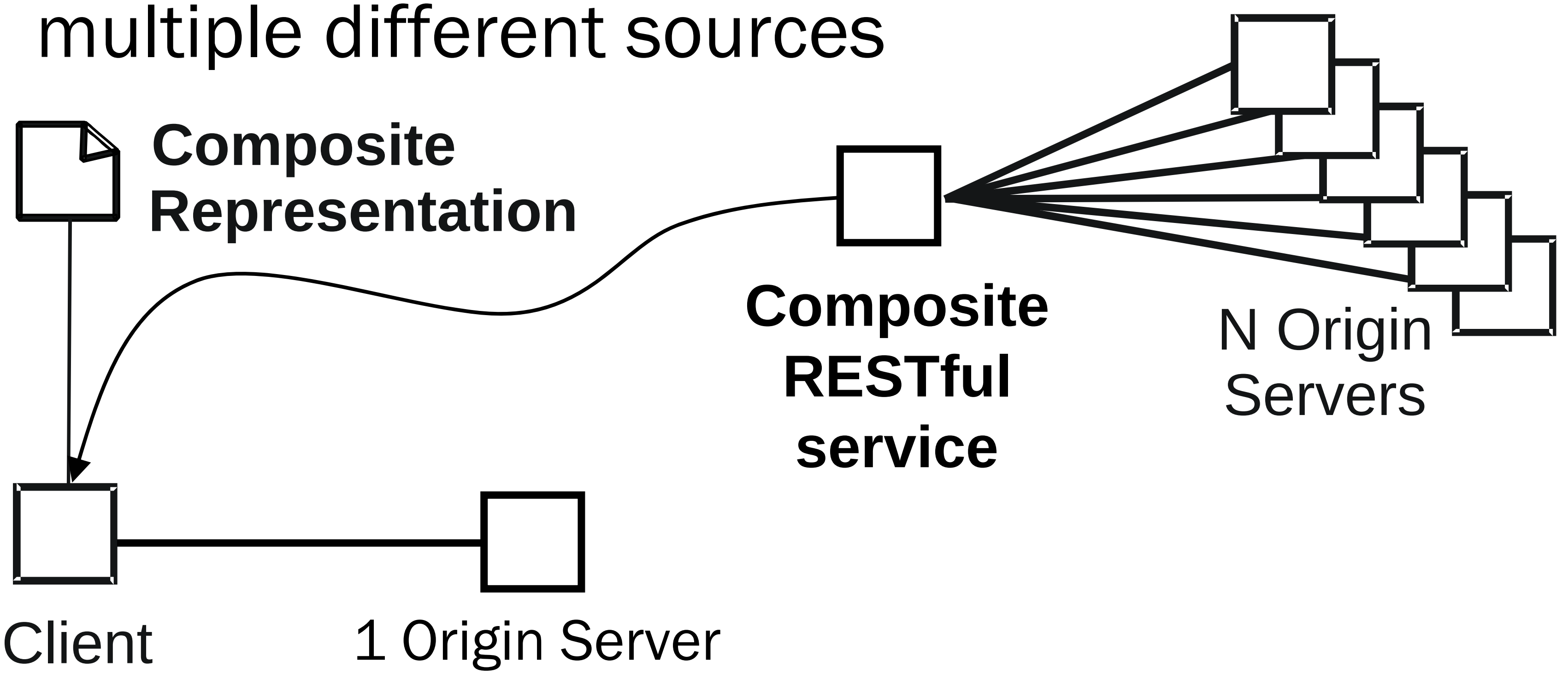
Single-Origin Sandbox

- Can you always do this from a web browser?

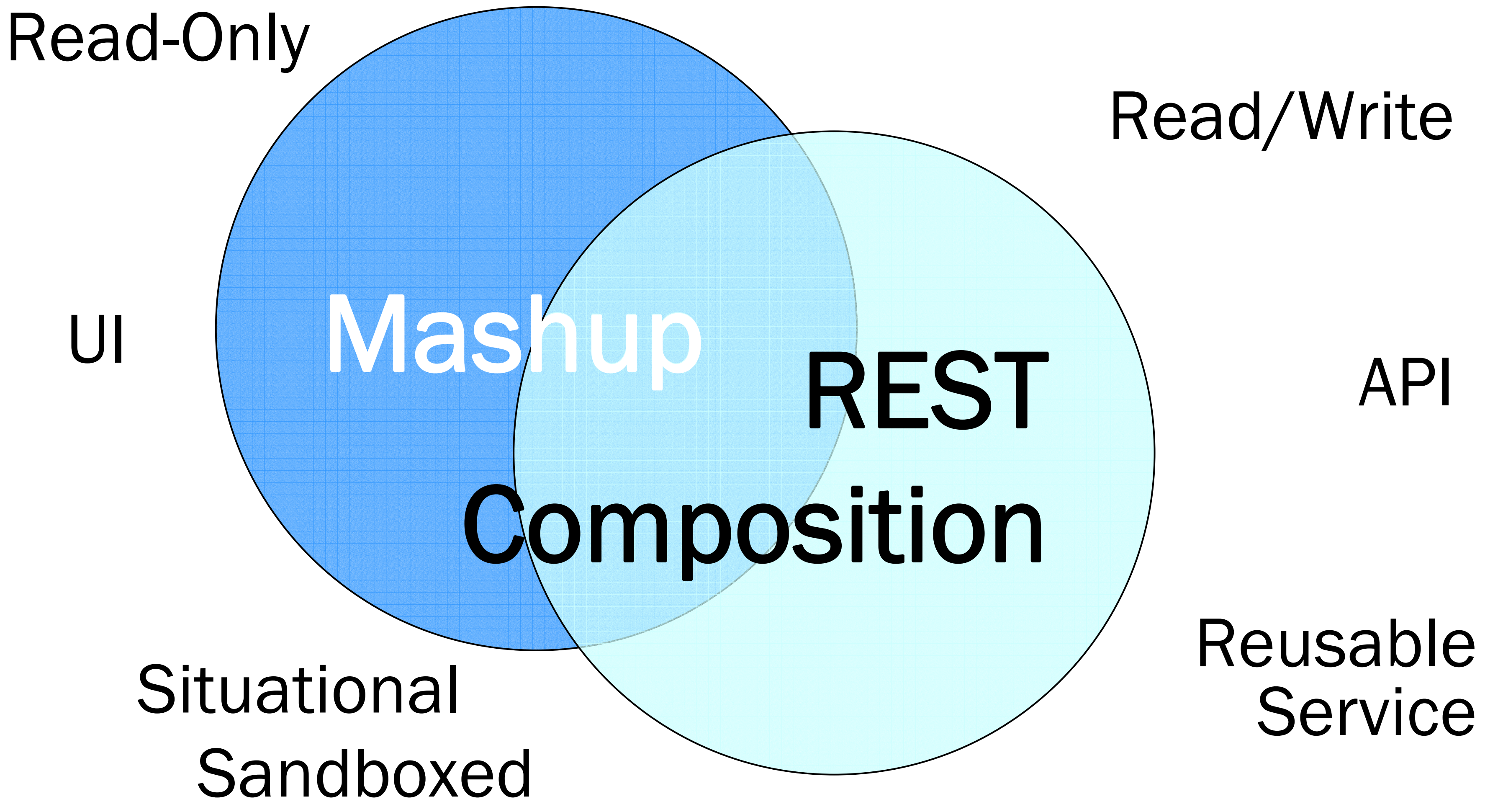


Single-Origin Sandbox

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



Complementary



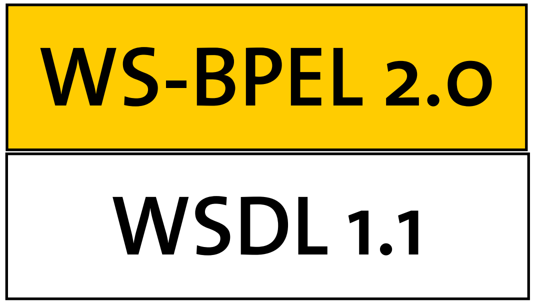
RESTful Composition Techniques

1. Defining RESTful service composition
2. Example: DoodleMap
3. What about mashups?
4. BPM and REST

Web Service Composition Today

“The WS-BPEL process model is layered on top of the service model defined by WSDL 1.1. [...]
Both the process and its partners are exposed as WSDL services”

[BPEL 2.0 Standard, Section 3]



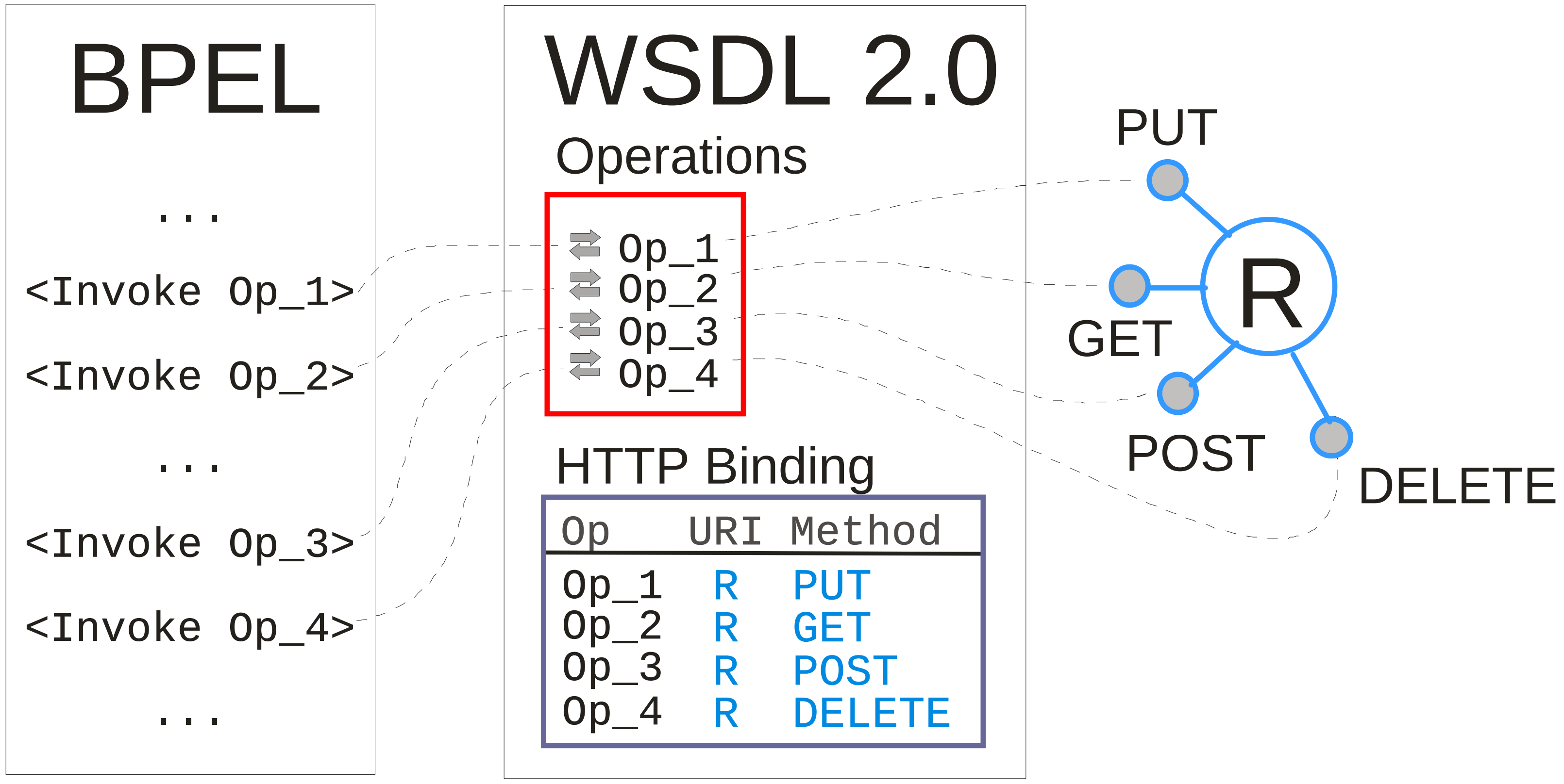
RESTful Web Services APIs...

...do not use WSDL 1.1

BPEL/WSDL 2.0

WSDL 2.0 HTTP Binding can wrap RESTful Web Services

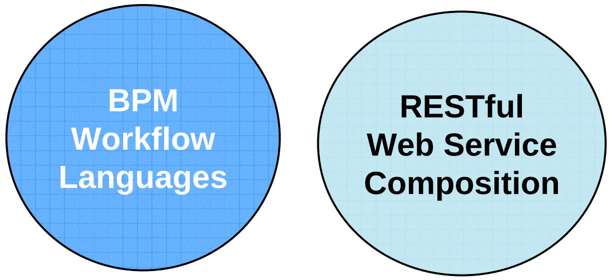
(WS-BPEL 2.0 does not support WSDL 2.0)



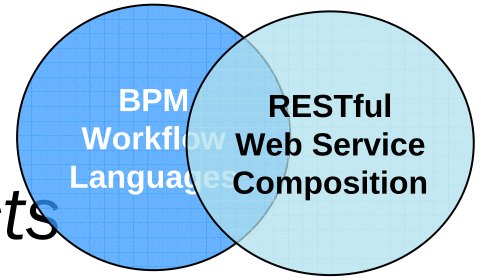
BPM
Workflow
Languages

RESTful
Web Service
Composition

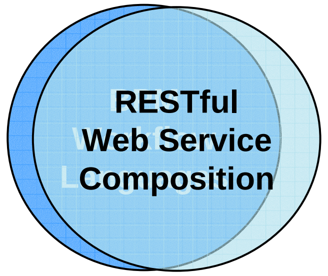
Solutions



1. Abstract Workflow
   - *Service invocation technology does not matter*
2. Concrete Workflow
   - *Expose service invocation technologies as explicit constructs in the workflow language*



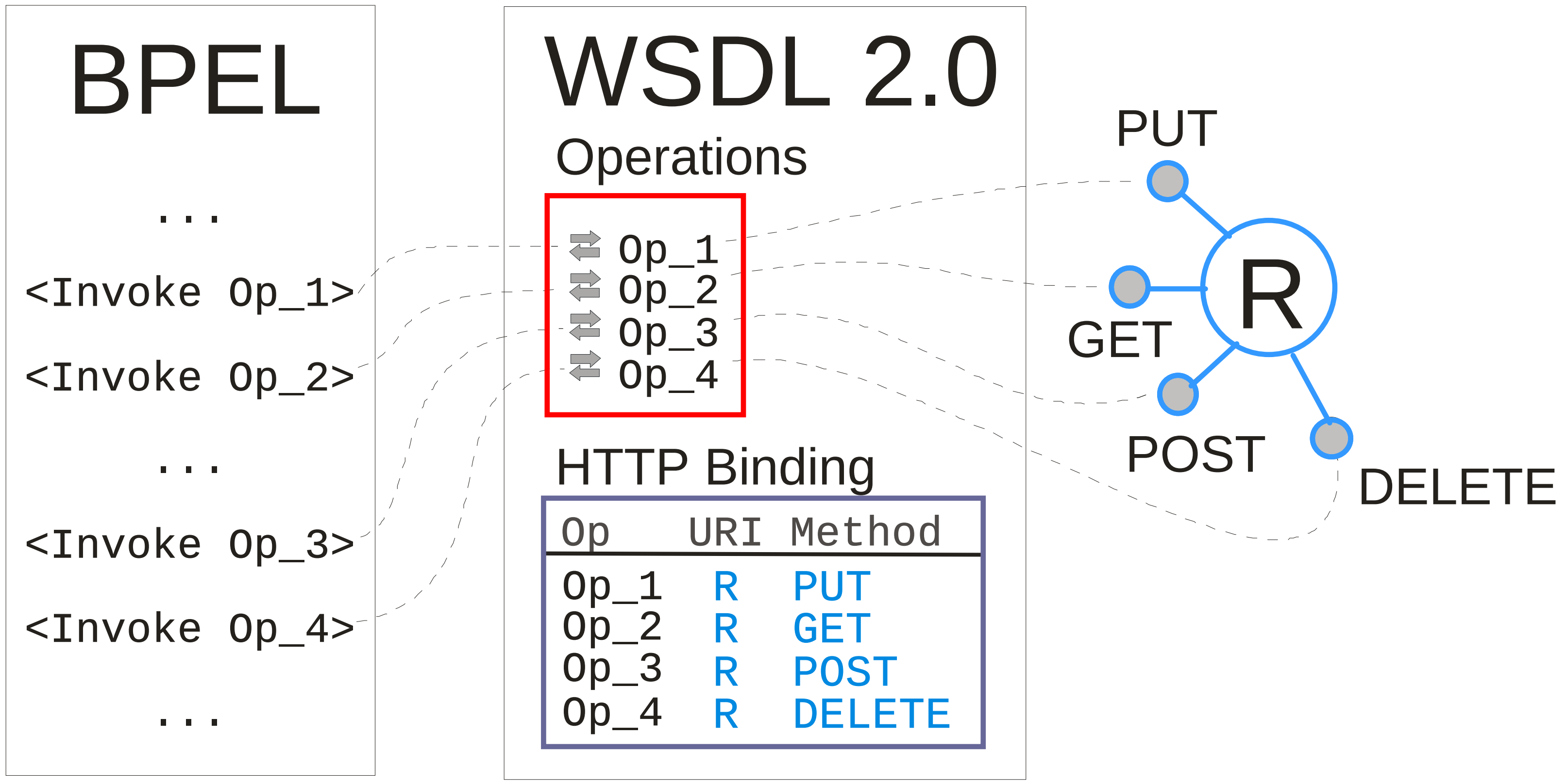
3. RESTful Workflow
   - *Workflow as one kind of resource exposed by a RESTful service*



BPEL/WSDL 2.0

WSDL 2.0 HTTP Binding can wrap RESTful Web Services

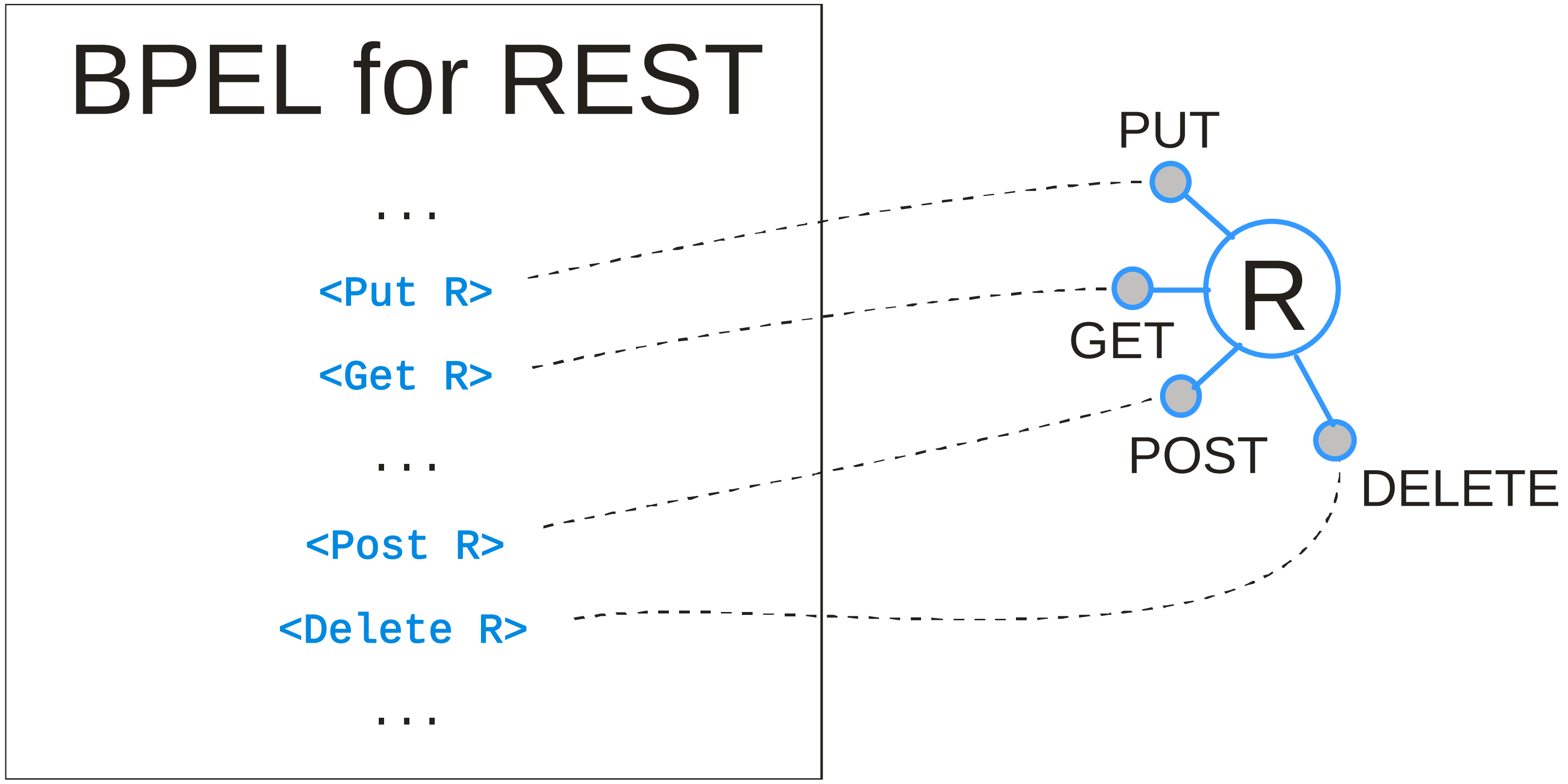
(WS-BPEL 2.0 does not support WSDL 2.0)



Op	URI	Method
Op_1	R	PUT
Op_2	R	GET
Op_3	R	POST
Op_4	R	DELETE

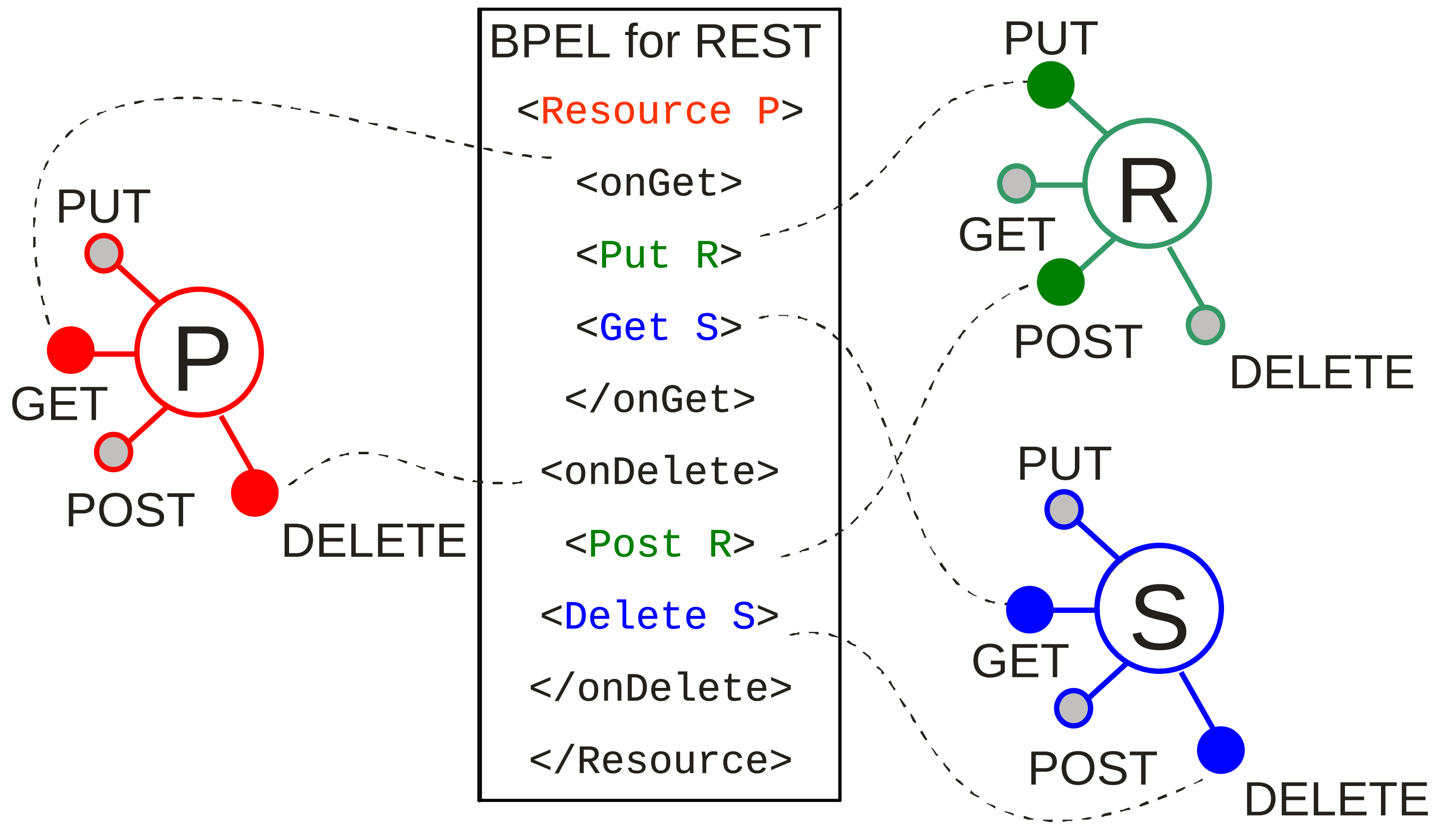
BPEL for REST

Make REST interaction primitives first-class language constructs

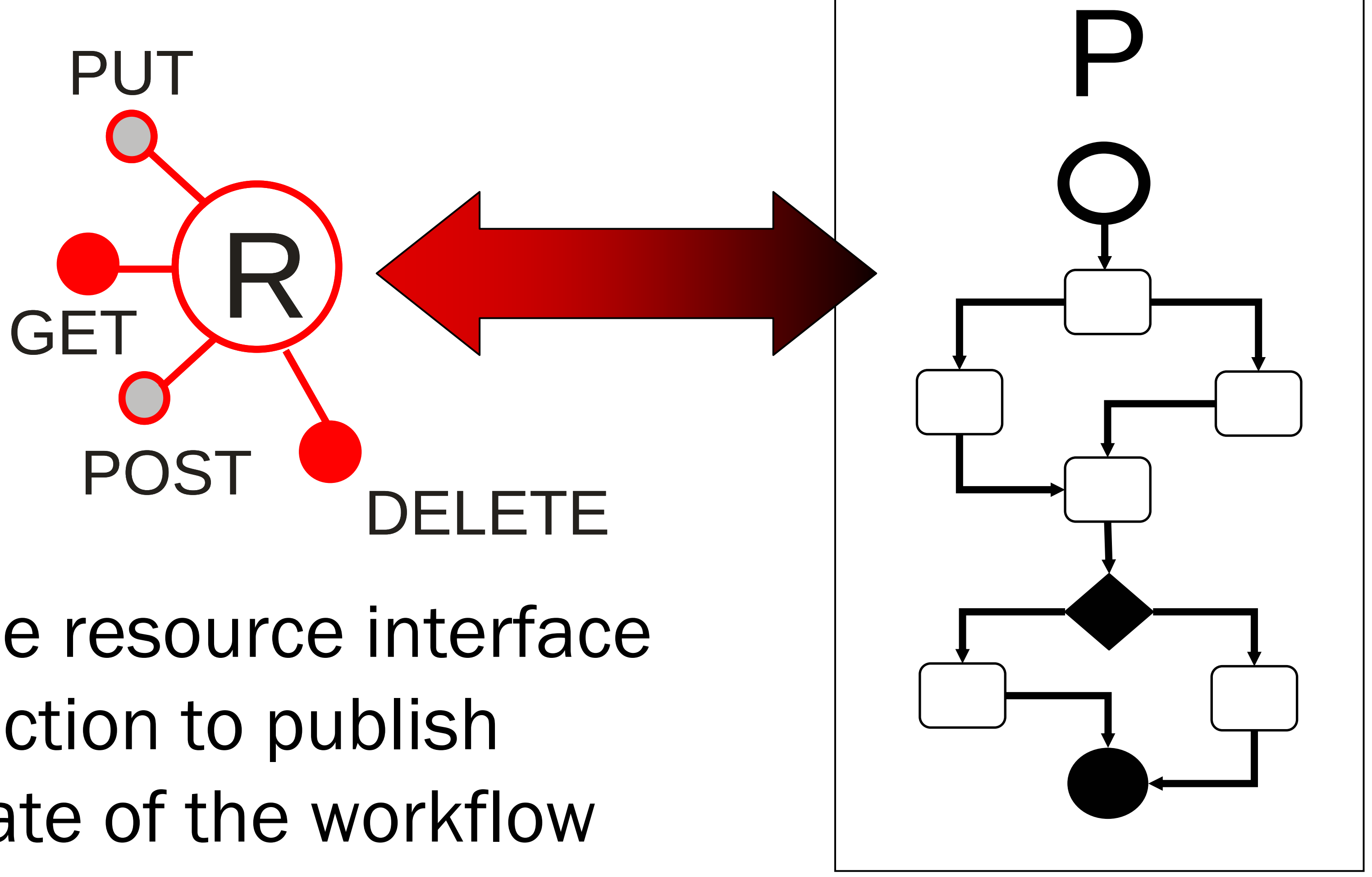


BPEL for REST

- Dynamically publish resources from BPEL processes and handle client requests

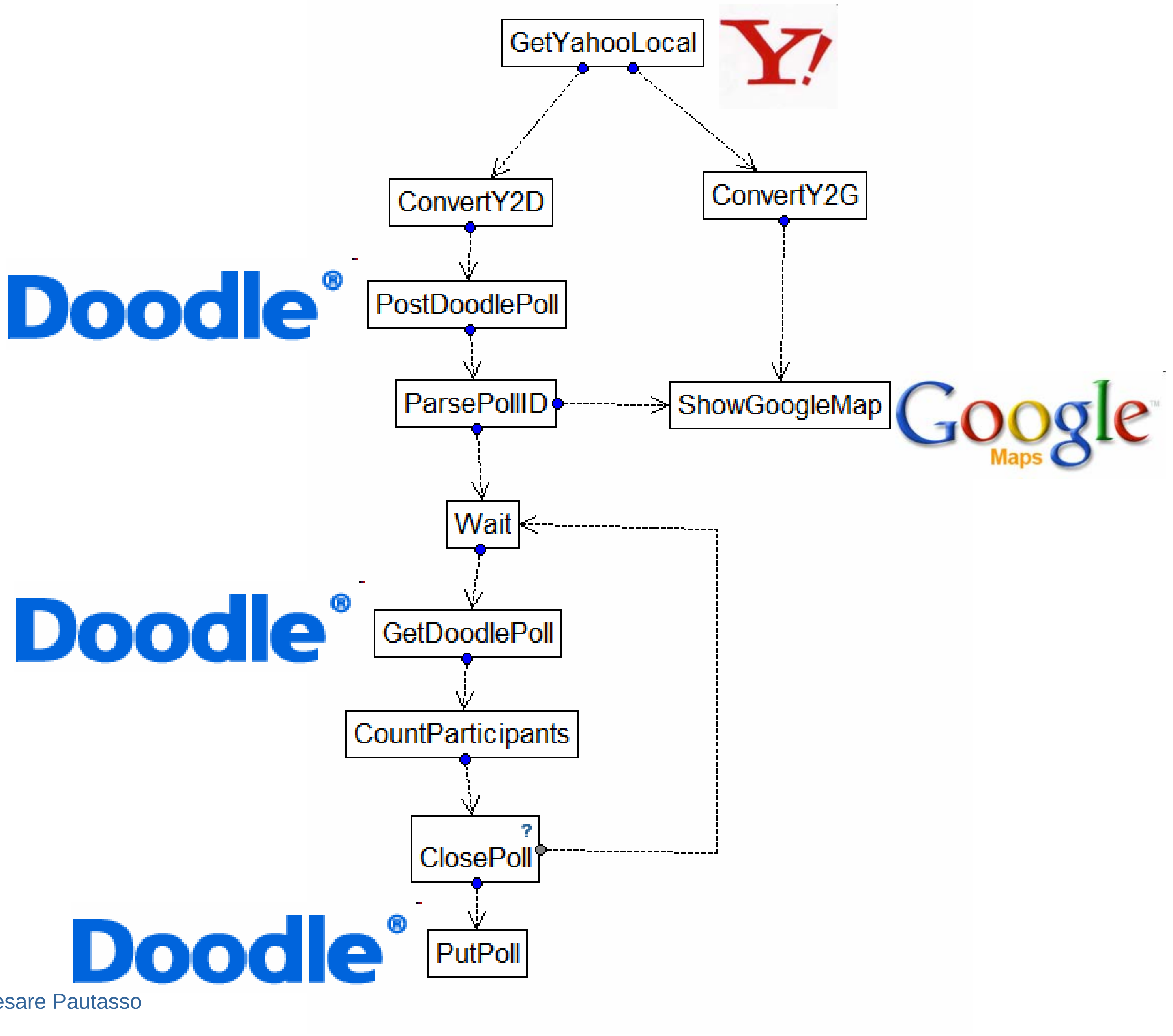


RESTful Workflows



Use the resource interface abstraction to publish the state of the workflow

DoodleMap as RESTful workflow

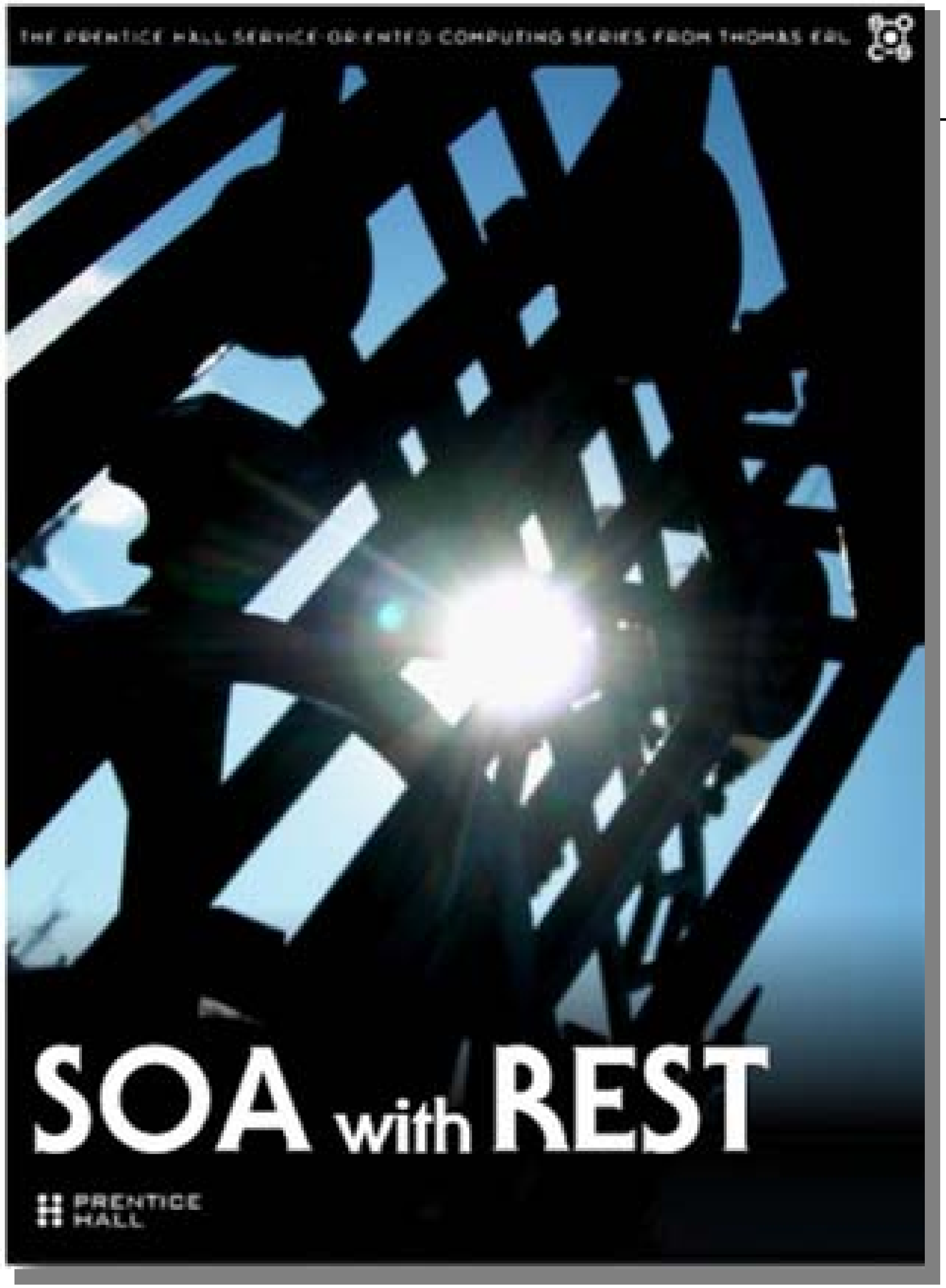


Conclusion

- Applying the SOA composition principle to REST gives interesting results
- Thanks to hyperlinks, REST brings a new (more dynamic and loosely coupled) twist to SOA composition
- Composing RESTful services helps to build mashups, but is different
- A RESTful API is the perfect abstraction for publishing the state of a workflow

References

- R. Fielding, Architectural Styles and the Design of Network-based Software Architectures, PhD Thesis, University of California, Irvine, 2000
- C. Pautasso, O. Zimmermann, F. 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
- C. Pautasso, BPEL for REST, Proc. of the 7th International Conference on Business Process Management (BPM 2008), Milano, Italy, September 2008
- C. Pautasso, Composing RESTful Services with JOpera, In: Proc. of the International Conference on Software Composition (SC2009), July 2009, Zurich, Switzerland.


THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL
SOA with REST
PRENTICE HALL

Raj Balasubramanian,
Benjamin Carlyle,
Thomas Erl,
Cesare Pautasso,

SOA with REST,

Prentice Hall,
to appear in 2010