Università della Svizzera italiana

Facoltà di scienze informatiche

REST vs. SOAP: Making the Right Architectural Decision

Cesare Pautasso

Faculty of Informatics

University of Lugano (USI), Switzerland

http://www.pautasso.info

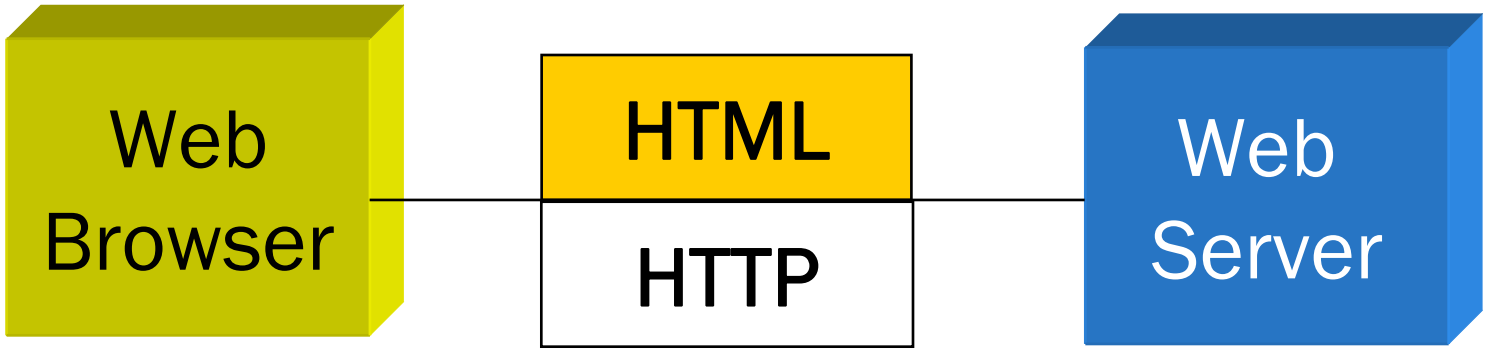
©2008 Cesare Pautasso

Agenda

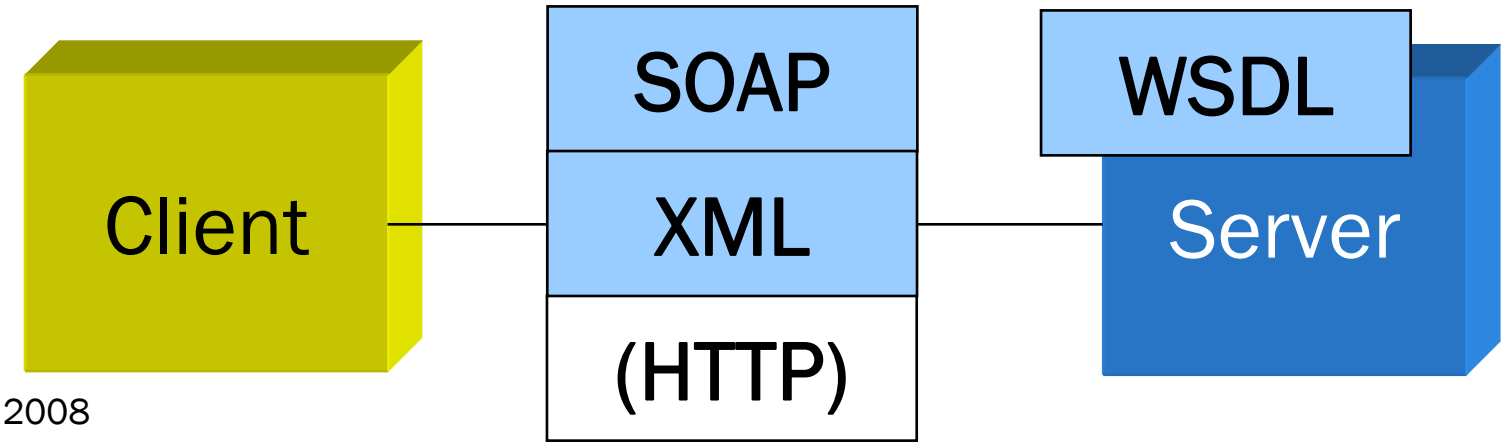
1. Motivation: A short history of Web Services
2. Comparing REST vs. SOAP/WS-*
3. Architectural Decision Modeling
4. Conceptual Comparison
5. Technology Comparison
6. How to measure the “complexity” of WS-* or the “simplicity” of REST?
7. Conclusion: Making the right Architectural Decision

©2008 Cesare Pautasso

Web Sites (1992)

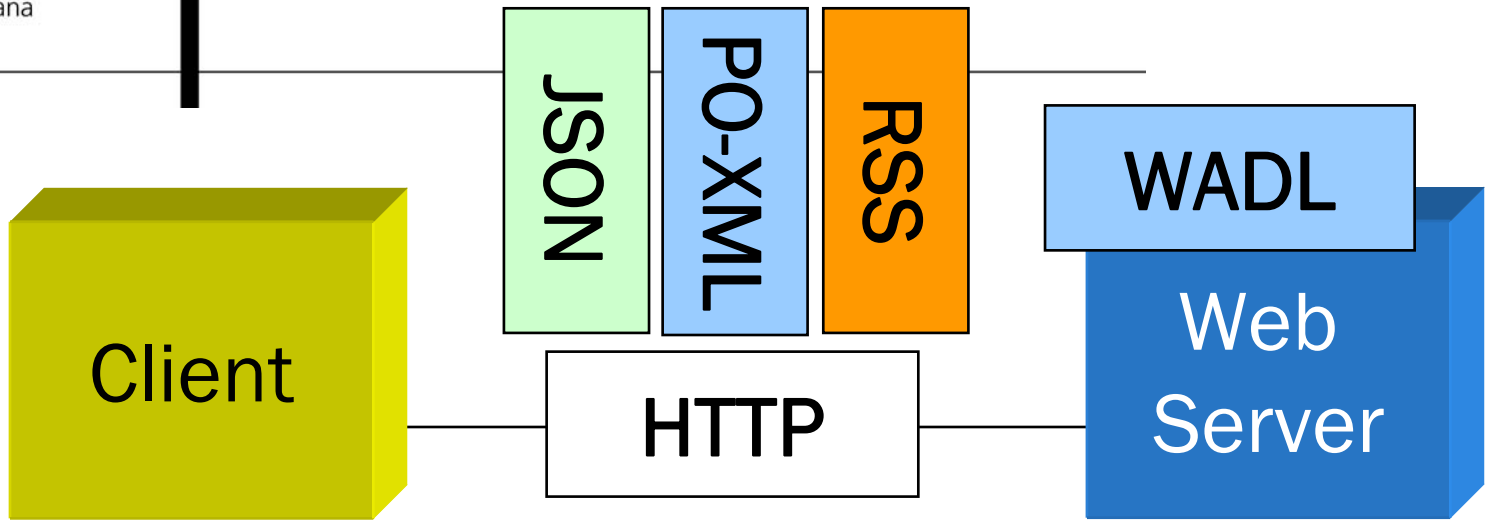


WS-* Web Services (2000)

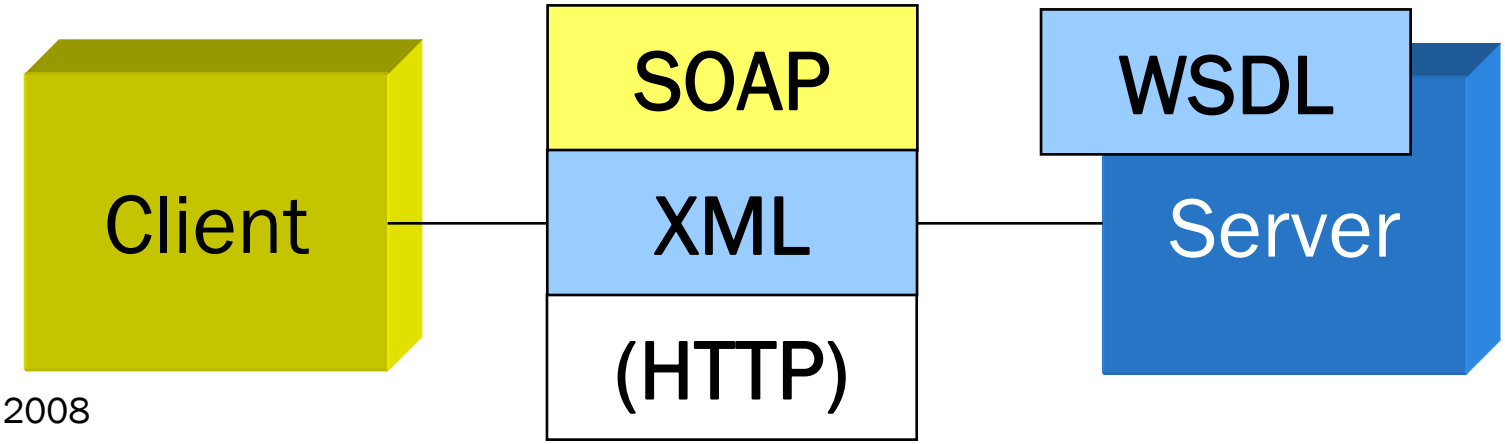


©2008 Cesare Pautasso

RESTful Web Services (2006)

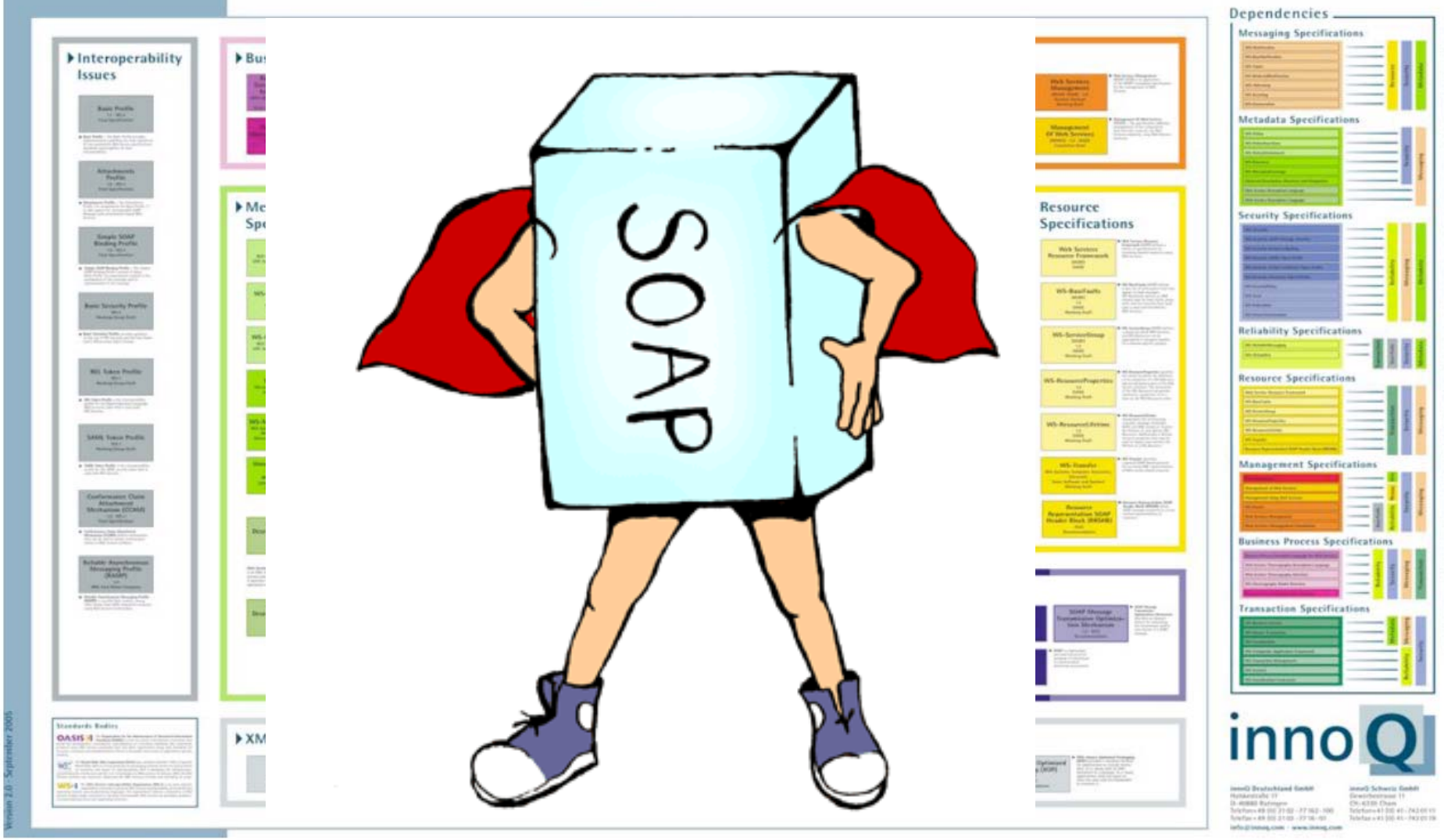


WS-* Web Services (2000)



©2008 Cesare Pautasso

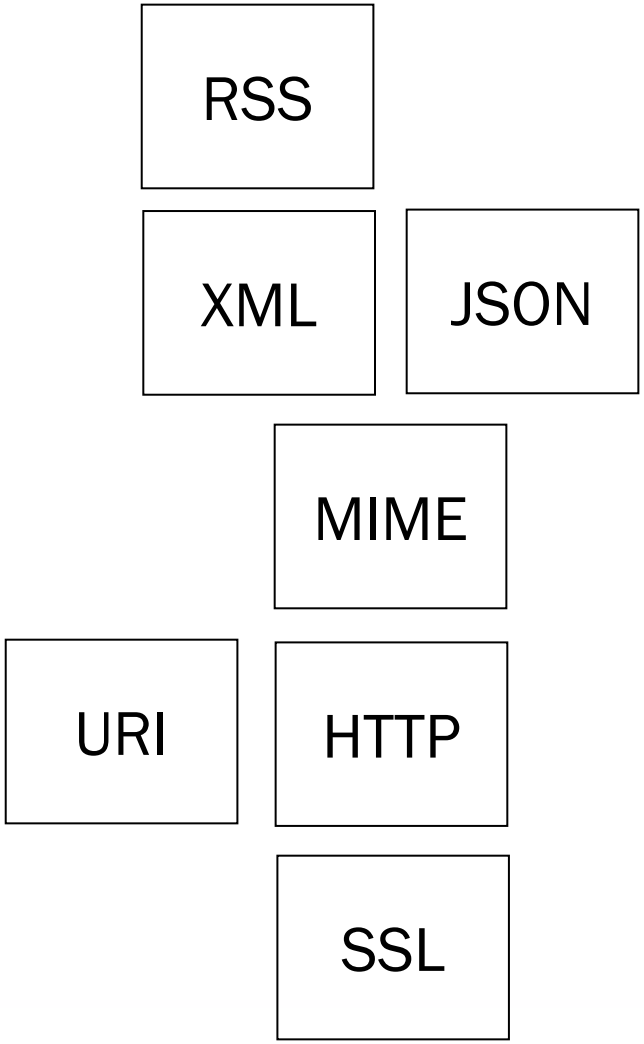
Web Services Standards



©2008 Cesare Pautasso

Web Services Standards

RESTful



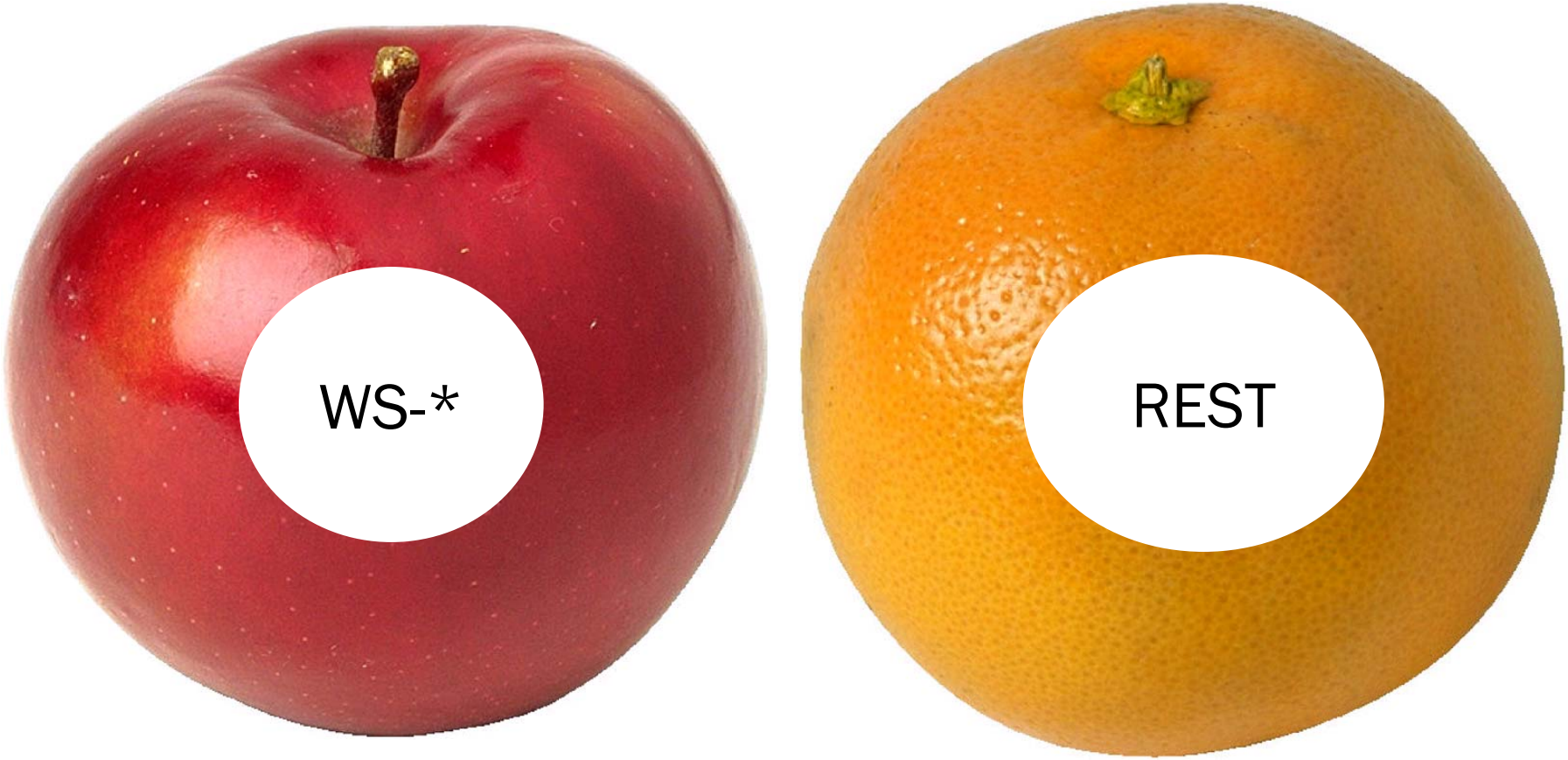
©2008 Cesare Pautasso

Is REST being used?

Slide from Paul Downey, BT

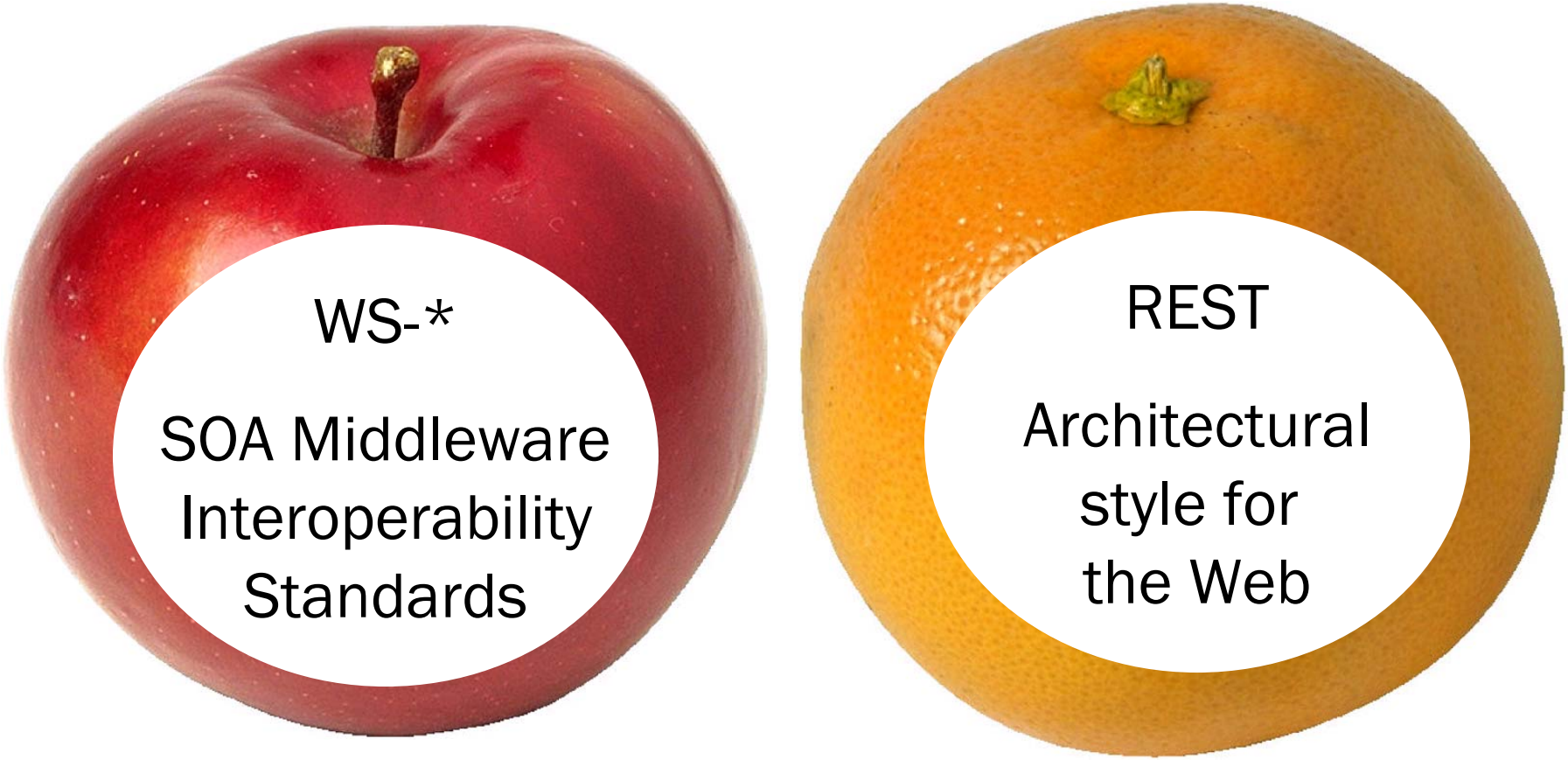
©2008 Cesare Pautasso

Can we really compare WS-* vs. REST?



©2008 Cesare Pautasso

Can we really compare WS-* vs. REST?

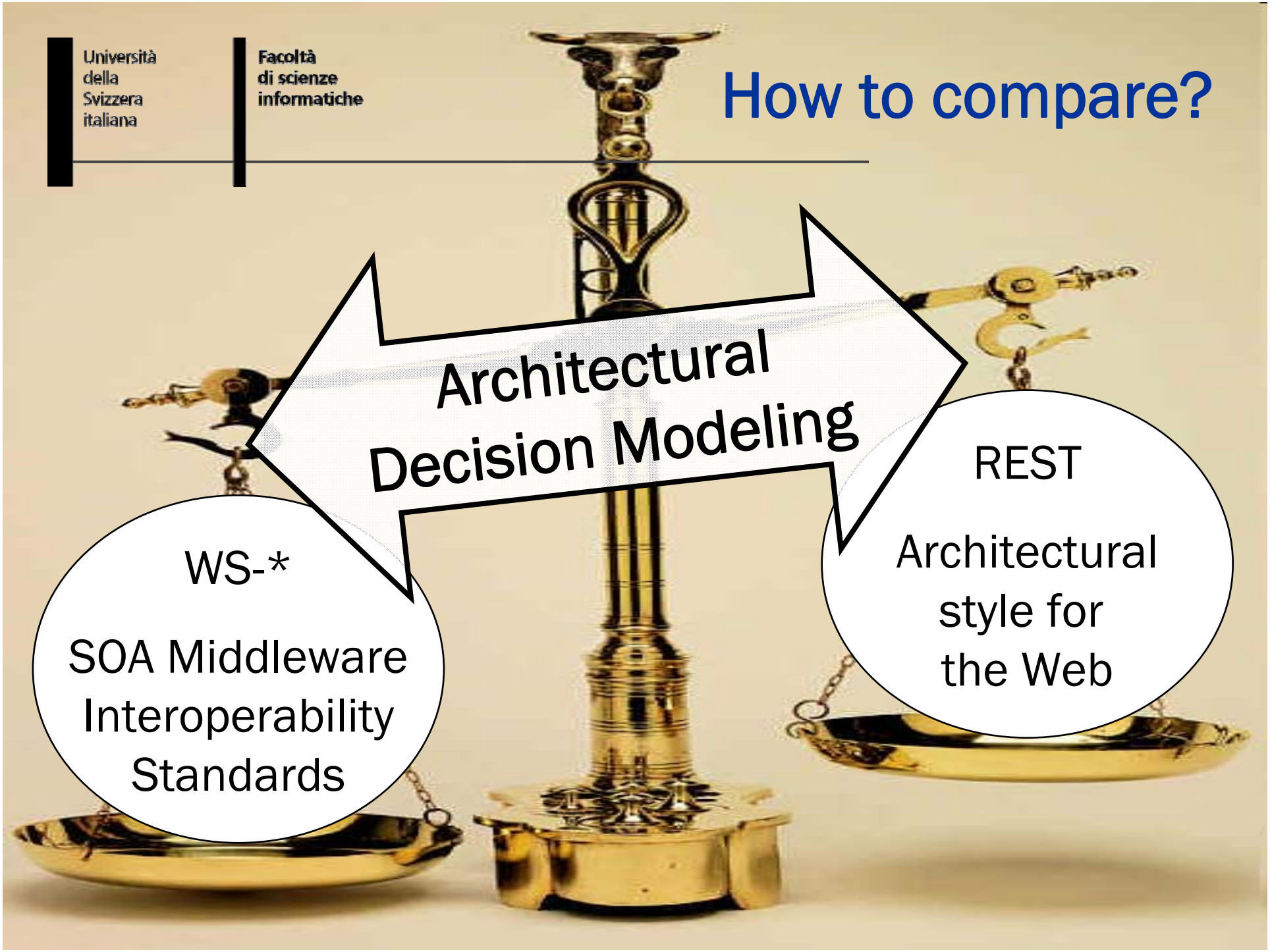


©2008 Cesare Pautasso

Università della Svizzera italiana

Facoltà di scienze informatiche

How to compare?



Architectural Decisions

- Architectural decisions capture the main design issues and the rationale behind a chosen technical solution
- The choice between REST vs. WS-* is an important architectural decision for integration projects
- Architectural decisions affect one another

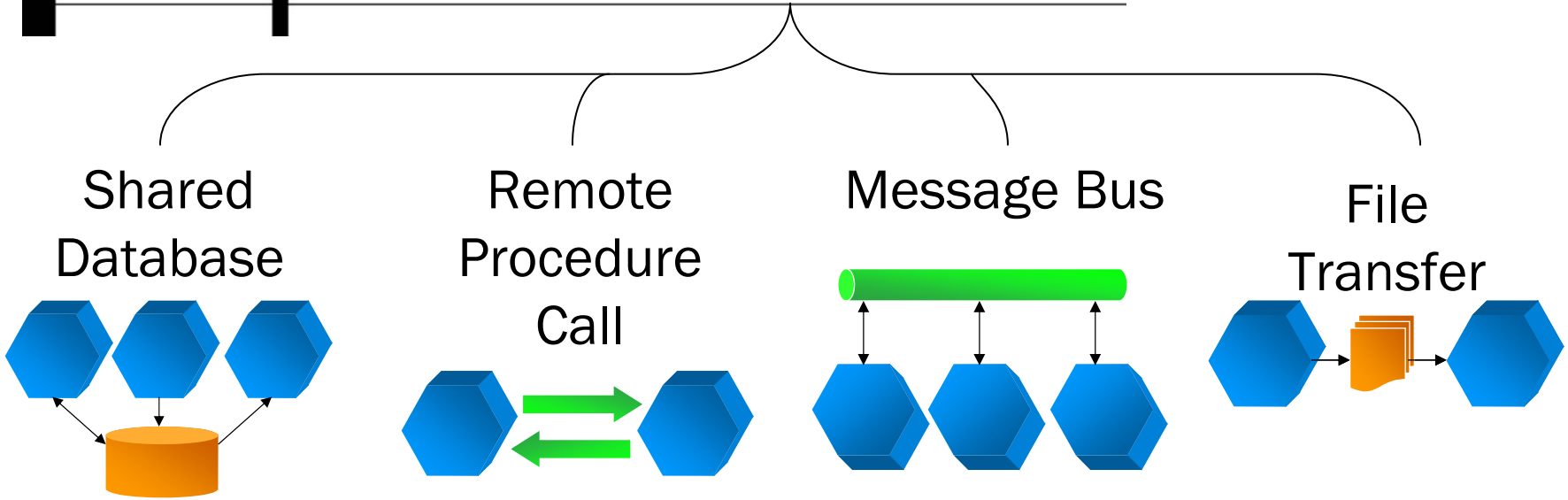
Architectural Decision:
Communication Protocol

Architecture Alternatives:
1. TCP
2. SMTP
3. HTTP
4. MQ
5. BEEP
6. CORBA IIOP
7. …

Rationale

©2008 Cesare Pautasso

Application Integration Styles

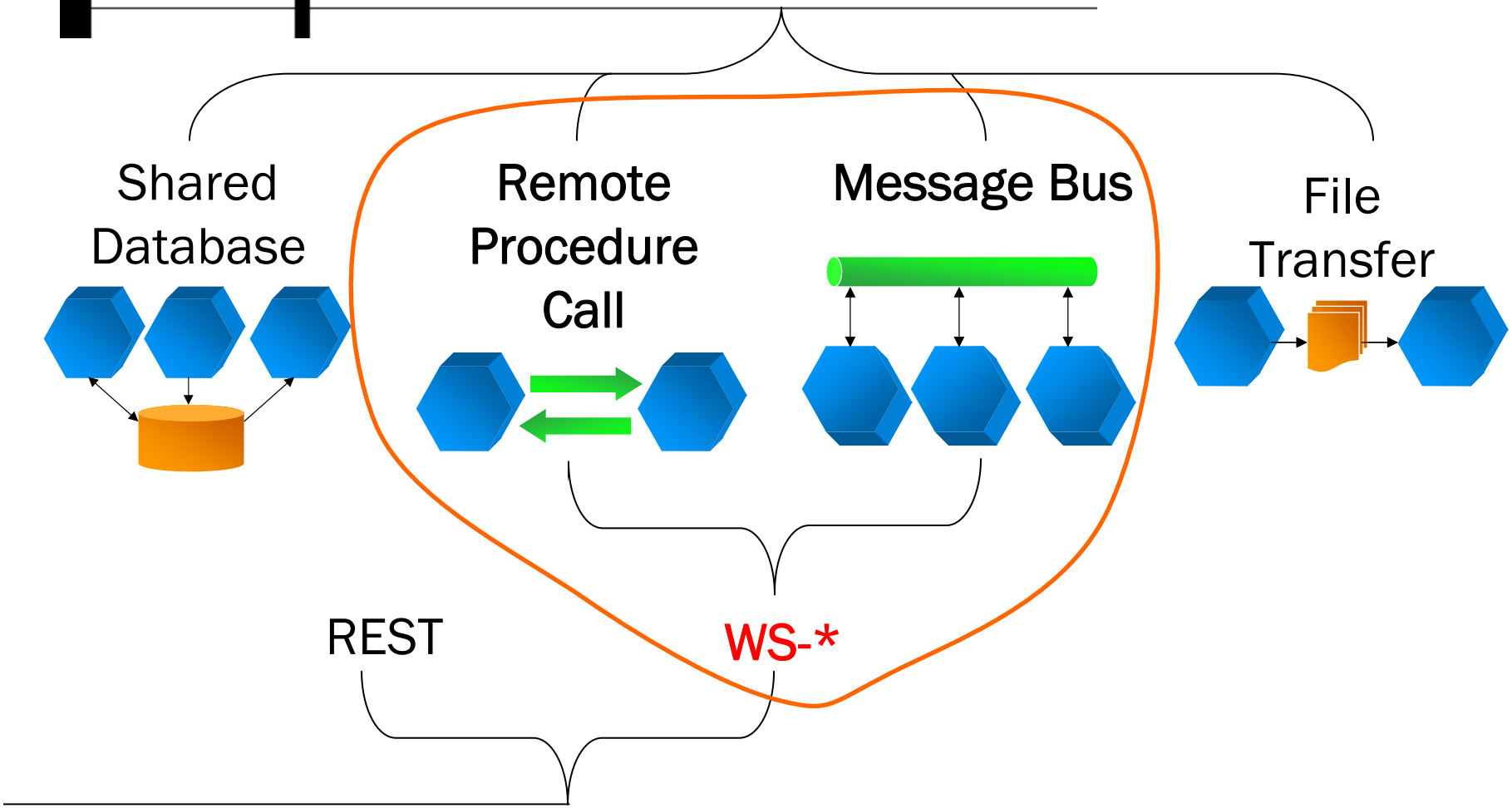


REST WS-*

Integration Technology Platform

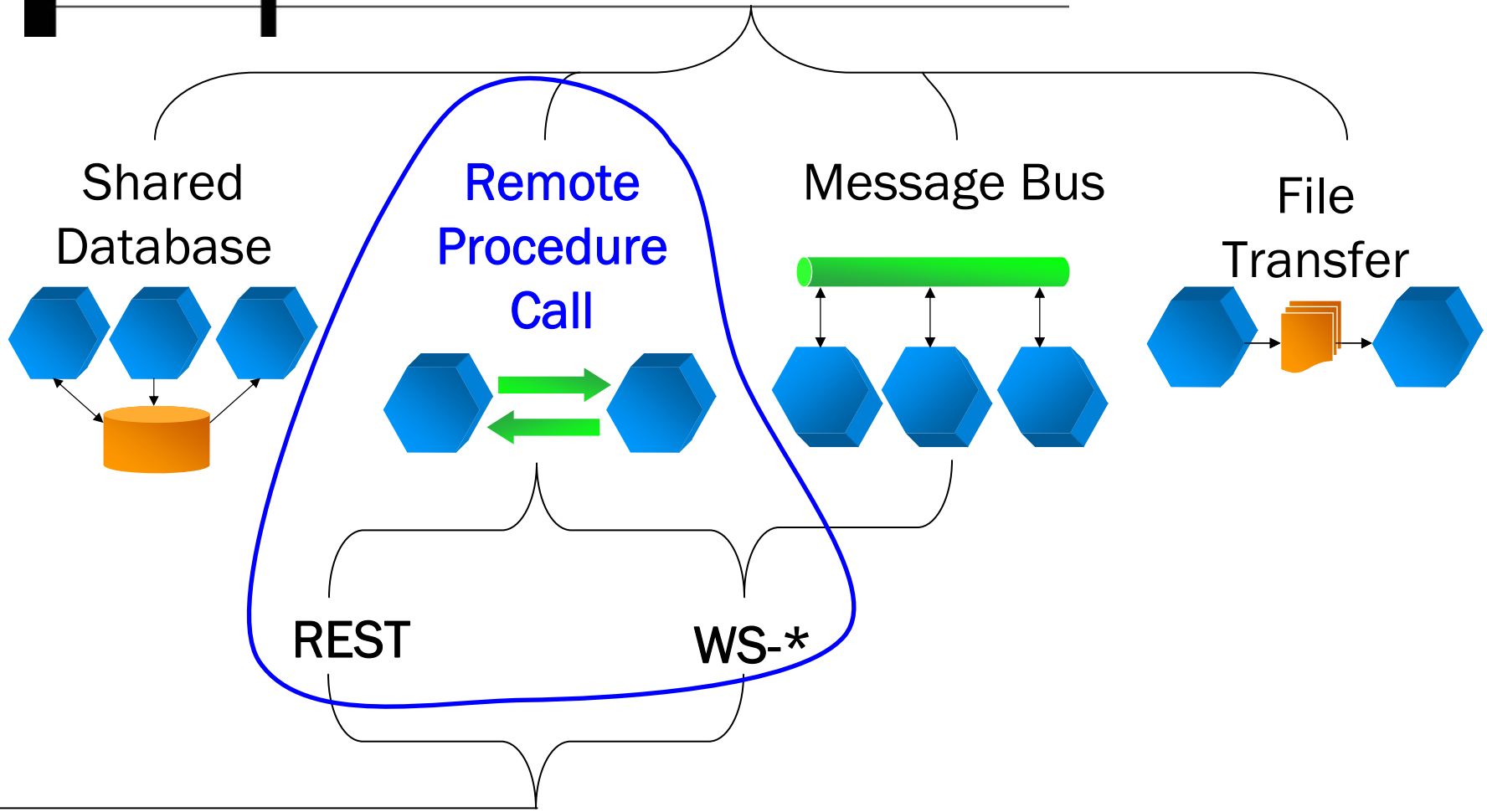
©2008 Cesare Pautasso

Related Decisions (WS-*)



©2008 Cesare Pautasso

Related Decisions (RPC)



©2008 Cesare Pautasso

Università della Svizzera italiana

Facoltà di scienze informatiche

Decision Space Overview

Architectural Decision and AAs	REST	WS-*
Integration Style	1 AA	2 AAs
Shared Database		
File Transfer		
Remote Procedure Call	✓	✓
Messaging		✓
Contract Design	1 AA	2 AAs
Contract-first		✓
Contract-last		✓
Contract-less	✓	
Resource Identification	1 AA	n/a
Do-it-yourself	✓	
URI Design	2 AA	n/a
"Nice" URI scheme	✓	
No URI scheme	✓	
Resource Interaction Semantics	2 AAs	n/a
Lo-REST (`POST`, `GET` only)	✓	
Hi-REST (4 verbs)	✓	
Resource Relationships	1 AA	n/a
Do-it-yourself	✓	
Data Representation/Modeling	1 AA	1 AA
XML Schema	(✓)[a]	✓
Do-it-yourself	✓	
Message Exchange Patterns	1 AA	2 AAs
Request-Response	✓	✓
One-Way		✓
Service Operations Enumeration	n/a	≥3 AAs
By functional domain		✓
By non-functional properties and QoS		✓
By organizational criterion (versioning)		✓
Total Number of Decisions, AAs	**8**, 10	**5**, ≥10

[a] Optional

Table 2: Conceptual Comparison Summary

Architectural Decision and AAs	REST	WS-*
Transport Protocol	1 AA	≥7 AAs
HTTP	✓	✓[a]
waka [13]	(✓)[b]	
TCP		✓
SMTP		✓
JMS		✓
MQ		✓
BEEP		✓
IIOP		✓
Payload Format	≥6 AAs	1 AA
XML (SOAP)	✓	✓
XML (POX)	✓	
XML (RSS)	✓	
JSON [10]	✓	
YAML	✓	
MIME	✓	
Service Identification	1 AA	2 AA
URI	✓	✓
WS-Addressing		✓
Service Description	3 AAs	2 AAs
Textual Documentation	✓	
XML Schema	(✓)[c]	✓
WSDL	✓[d]	✓
WADL [18]	✓	
Reliability	1 AA	4 AAs
HTTPR [38][e]	(✓)	(✓)
WS-Reliability		✓
WS-ReliableMessaging		✓
Native		✓
Do-it-yourself	✓	✓
Security	1 AA	2 AAs
HTTPS	✓	✓
WS-Security		✓

Transactions	1 AA	3 AAs
WS-AT, WS-BA		✓
WS-CAF		✓
Do-it-yourself	✓	✓
Service Composition	2 AAs	2 AAs
WS-BPEL		✓
Mashups	✓	
Do-it-yourself	✓	✓
Service Discovery	1 AAs	2 AAs
UDDI		✓
Do-it-yourself	✓	✓
Implementation Technology	many	many
...	✓	✓
Total Number of Decisions, AAs	**10**, ≥17	**10**, ≥25

[a] Limited to only the verb `POST`
[b] Still under development
[c] Optional
[d] WSDL 2.0
[e] Not standard

Table 3: Technology Comparison Summary

Architectural Principle and Aspects	REST	WS-*
Protocol Layering	yes	yes
HTTP as application-level protocol	✓	
HTTP as transport-level protocol		✓
Dealing with Heterogeneity	yes	yes
Browser Wars	✓	
Enterprise Computing Middleware		✓
Loose Coupling, aspects covered	yes, 2	yes, 3
Time/Availability		✓
Location (Dynamic Late Binding)	(✓)	✓
Service Evolution:		
Uniform Interface	✓	
XML Extensibility	✓	✓
Total Principles Supported	**3**	**3**

Table 1: Principles Comparison Summary

©2008 Cesare Pautasso

Università della Svizzera italiana

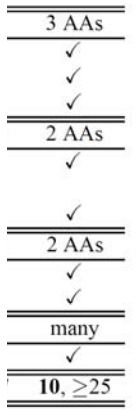
Facoltà di scienze informatiche

Decision Space Summary

21 Decisions and 64 alternatives

Classified by level of abstraction:

- 3 Architectural **Principles**
- 9 **Conceptual** Decisions
- 9 **Technology**-level Decisions



Decisions help us to **measure the complexity** implied by the choice of REST or WS-*

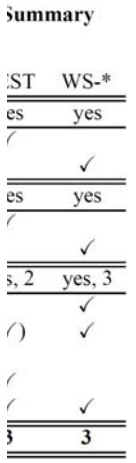


Table 2: Conceptual Comparison Summary

Table 1: Principles Comparison Summary

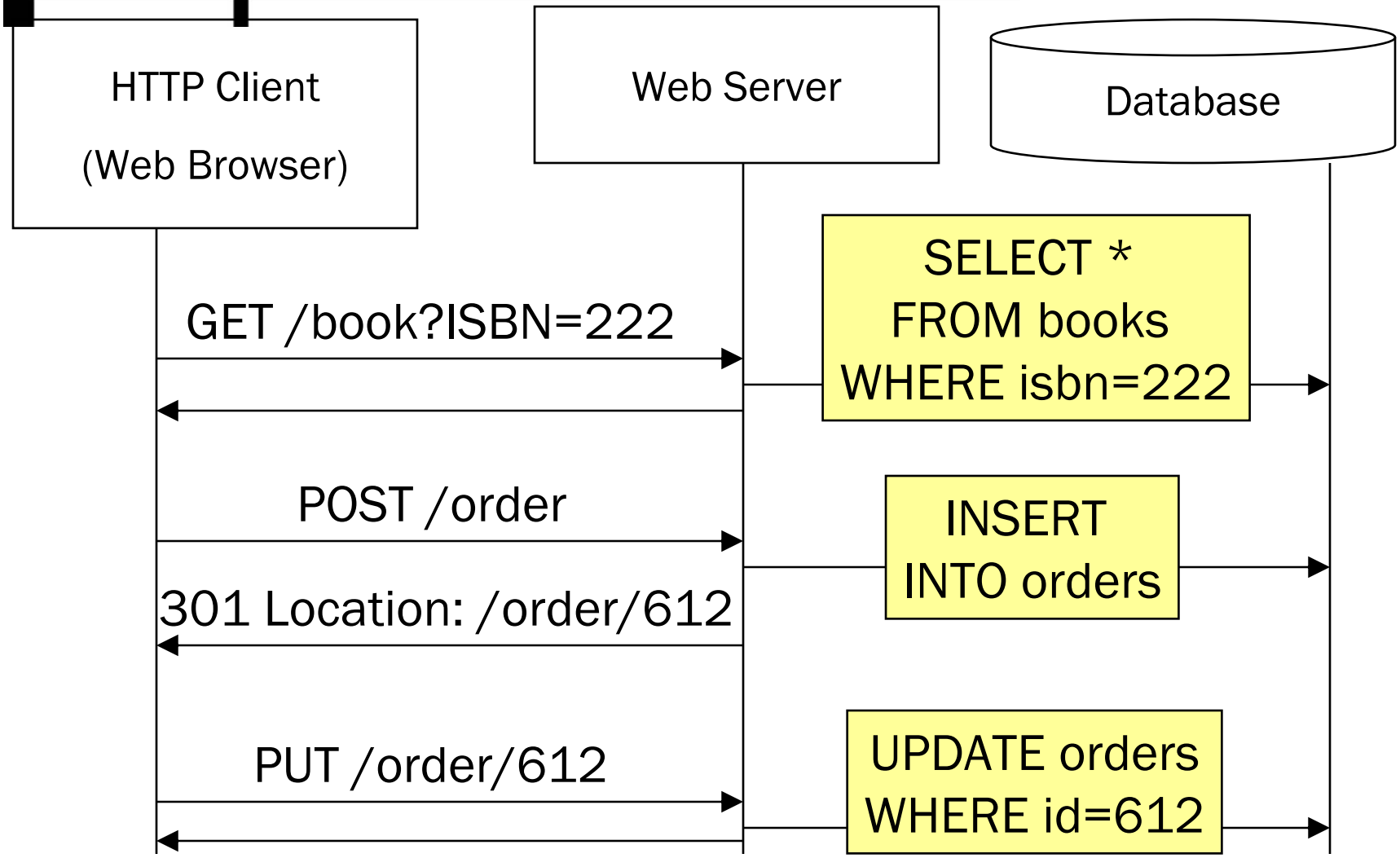
©2008 Cesare Pautasso

Architectural Principles

1. Protocol Layering
   - HTTP = Application-level Protocol (REST)
   - HTTP = Transport-level Protocol (WS-*)
2. Dealing with Heterogeneity
3. Loose Coupling

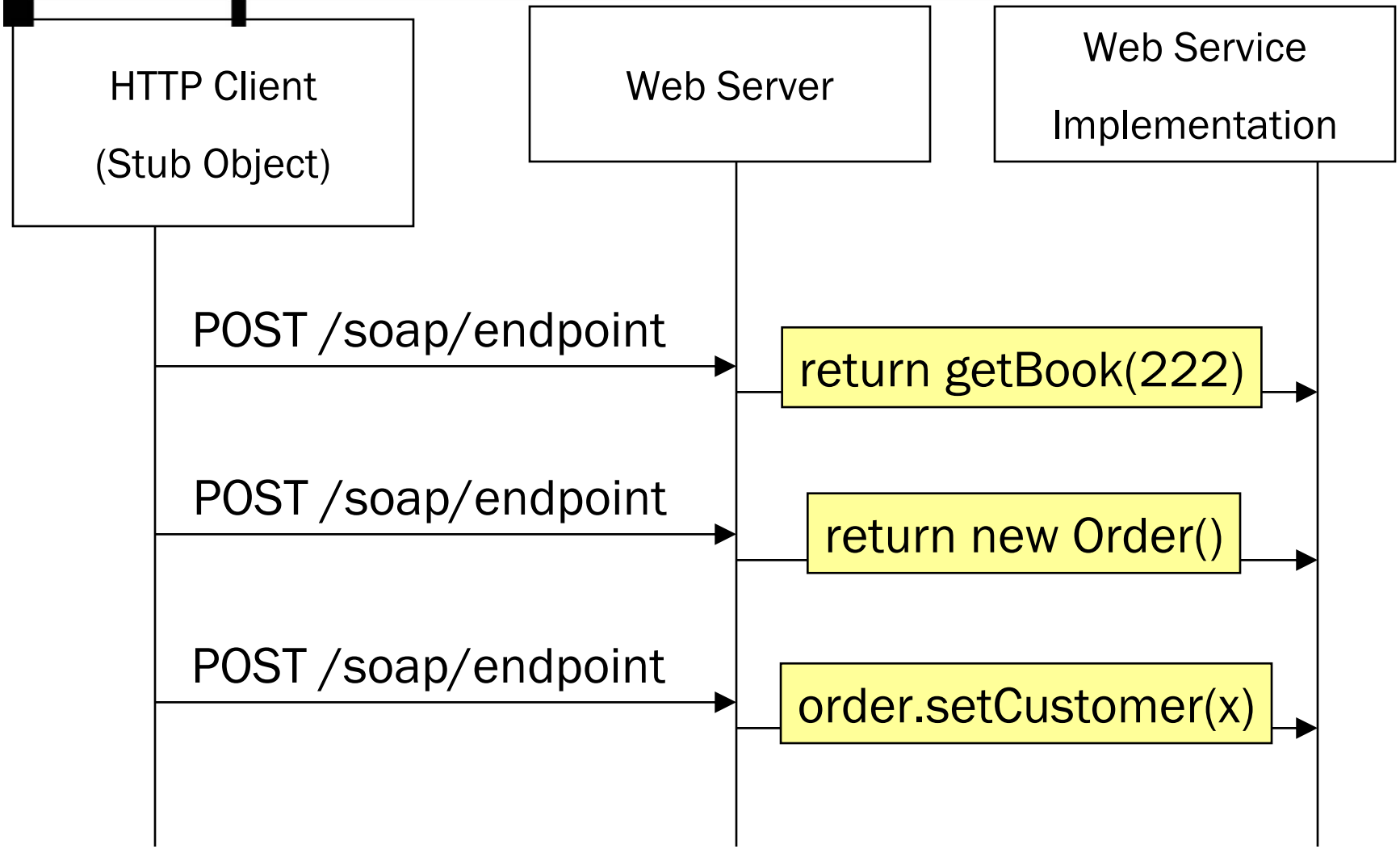
©2008 Cesare Pautasso

RESTful Web Service Example



©2008 Cesare Pautasso

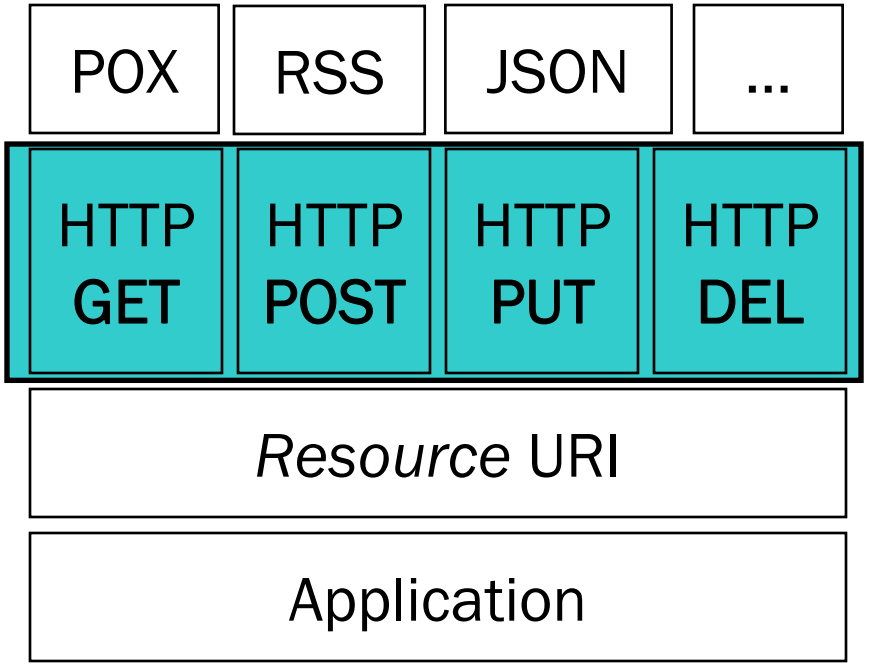
Big Web Service Example (from REST perspective)



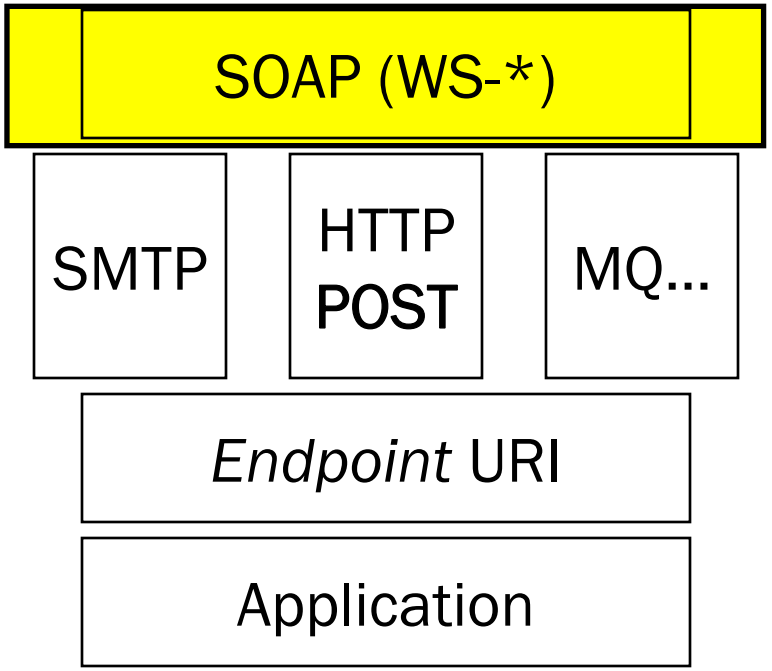
©2008 Cesare Pautasso

Protocol Layering

- "The Web is the universe of globally accessible information" (Tim Berners Lee)
  - Applications should publish their data on the Web (through URI)



- "The Web is the universal (tunneling) transport for messages"
  - Applications get a chance to interact but they remain "outside of the Web"

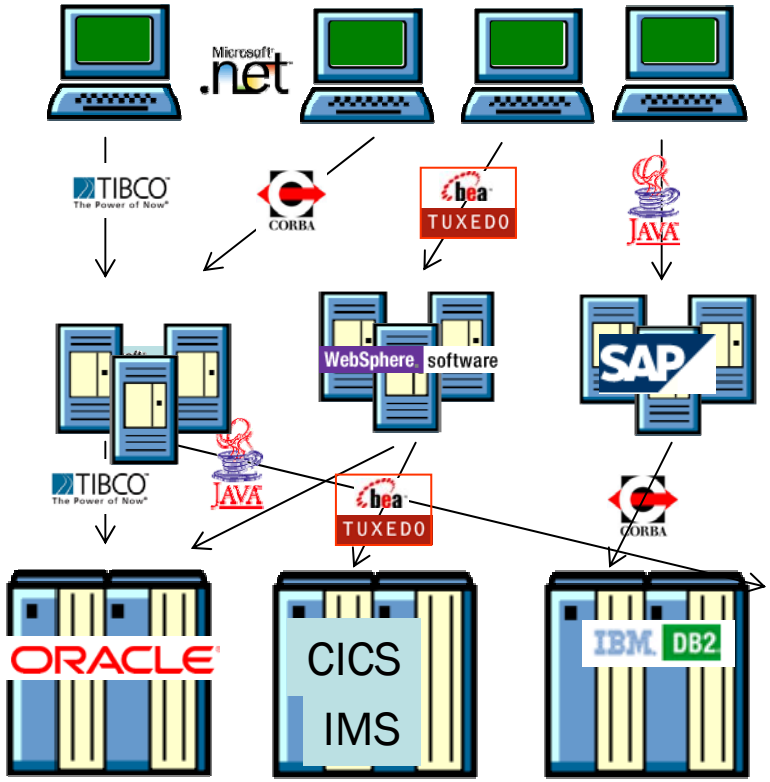


©2008 Cesare Pautasso

Dealing with Heterogeneity

- Web Applications

- Enterprise Computing



Picture from Eric Newcomer, IONA

©2008 Cesare Pautasso

Conceptual Comparison

Architectural Decision and AAs	REST	WS-*
Integration Style	1 AA	2 AAs
Shared Database		
File Transfer		
Remote Procedure Call	✓	✓
Messaging		✓
Contract Design	1 AA	2 AAs
Contract-first		✓
Contract-last		✓
Contract-less		
Resource Identification	AA	n/a
Do-it-yourself	✓	
URI Design	2 AA	n/a

Note: this table will scroll up during the talk

Technology Comparison

Architectural Decision and AAs	REST	WS-*
Transport Protocol	1 AA	≥7 AAs
HTTP	✓	✓[a]
waka [13]	(✓)[b]	
TCP		✓
SMTP		✓
JMS		✓
MQ		✓
BEEP		✓
IIOP		✓
Payload Format	≥6 AAs	1 AA
XML (SOAP)	✓	✓
XML (POX)	✓	
XML (RSS)	✓	

Note: this table will scroll up during the talk

Measuring Complexity

- Architectural Decisions give a **quantitative measure** of the complexity of an architectural design space:
  - Total number of decisions
  - For each decision, number of alternative options
  - For each alternative option, estimate the effort

	REST	WS-*
Decisions	17	14
Alternatives	27	35

Decisions with *1 or more* alternative options

©2008 Cesare Pautasso

Measuring Complexity

	REST	WS-*
Decisions	5	12
Alternatives	16	32

Decisions with *more than 1* alternative options

	REST	WS-*
Decisions	17	14
Alternatives	27	35

Decisions with *1 or more* alternative options

©2008 Cesare Pautasso

Measuring Complexity

	REST	WS-*
Decisions	5	12
Alternatives	16	32

Decisions with *more than 1* alternative options

- URI Design
- Resource Interaction Semantics
- Payload Format
- Service Description
- Service Composition

©2008 Cesare Pautasso

Measuring Complexity

	REST	WS-*
Decisions	5	12
Alternatives	16	32

Decisions with *more than 1* alternative options

	REST	WS-*
Decisions	12	2

Decisions with *only 1* alternative option

©2008 Cesare Pautasso

Measuring Complexity

- Payload Format
- Data Representation Modeling

	REST	WS-*
Decisions	12	2

Decisions with *only 1* alternative option

©2008 Cesare Pautasso

Measuring Effort

	REST	WS-*
Do-it-yourself Alternatives	5	0

Decisions with **only** *do-it-yourself* alternatives

	REST	WS-*
Decisions	12	2

Decisions with *only 1* alternative option

©2008 Cesare Pautasso

Measuring Effort

	REST	WS-*
Do-it-yourself Alternatives	5	0

Decisions with **only** *do-it-yourself* alternatives

- Resource Identification
- Resource Relationship

- Reliability
- Transactions
- Service Discovery

©2008 Cesare Pautasso

Università della Svizzera italiana

Facoltà di scienze informatiche

Freedom of Choice
Freedom from Choice

October 7-8, 2008 — 1st International SOA SYMPOSIUM

Architectural Decision and AAs	REST	WS-*
Integration Style	1 AA	2 AAs
Shared Database		
File Transfer		
Remote Procedure Call	✓	✓
Messaging		✓
Contract Design	1 AA	2 AAs
Contract-first		✓
Contract-last		✓
Contract-less	✓	
Resource Identification	1 AA	n/a
Do-it-yourself	✓	
URI Design	2 AA	n/a
"Nice" URI scheme	✓	
No URI scheme	✓	
Resource Interaction Semantics	2 AAs	n/a
Lo-REST (`POST`, `GET` only)	✓	
Hi-REST (4 verbs)	✓	
Resource Relationships	1 AA	n/a
Do-it-yourself	✓	
Data Representation/Modeling	1 AA	1 AA
XML Schema	(✓)[a]	✓
Do-it-yourself	✓	
Message Exchange Patterns	1 AA	2 AAs
Request-Response	✓	✓
One-Way		✓
Service Operations Enumeration	n/a	≥3 AAs
By functional domain		✓
By non-functional properties and QoS		✓
By organizational criterion (versioning)		✓
Total Number of Decisions, AAs	**8**, 10	**5**, ≥10

[a] Optional

Table 2: Conceptual Comparison Summary

Architectural Decision and AAs	REST	WS-*
Transport Protocol	1 AA	≥7 AAs
HTTP	✓	✓[a]
waka [13]	(✓)[b]	
TCP		✓
SMTP		✓
JMS		✓
MQ		✓
BEEP		✓
IIOP		✓
Payload Format	≥6 AAs	1 AA
XML (SOAP)	✓	✓
XML (POX)	✓	
XML (RSS)	✓	
JSON [10]	✓	
YAML	✓	
MIME	✓	
Service Identification	1 AA	2 AA
URI	✓	✓
WS-Addressing		✓
Service Description	3 AAs	2 AAs
Textual Documentation	✓	
XML Schema	(✓)[c]	✓
WSDL	✓[d]	✓
WADL [18]	✓	
Reliability	1 AA	4 AAs
HTTPR [38][e]	(✓)	(✓)
WS-Reliability		✓
WS-ReliableMessaging		✓
Native		✓
Do-it-yourself	✓	✓
Security	1 AA	2 AAs
HTTPS	✓	✓
WS-Security		✓

Transactions	1 AA	3 AAs
WS-AT, WS-BA		✓
WS-CAF		✓
Do-it-yourself	✓	✓
Service Composition	2 AAs	2 AAs
WS-BPEL		✓
Mashups	✓	
Do-it-yourself	✓	✓
Service Discovery	1 AAs	2 AAs
UDDI		✓
Do-it-yourself	✓	✓
Implementation Technology	many	many
…	✓	✓
Total Number of Decisions, AAs	**10**, ≥17	**10**, ≥25

[a] Limited to only the verb `POST`
[b] Still under development
[c] Optional
[d] WSDL 2.0
[e] Not standard

Table 3: Technology Comparison Summary

Architectural Principle and Aspects	REST	WS-*
Protocol Layering	yes	yes
HTTP as application-level protocol	✓	
HTTP as transport-level protocol		✓
Dealing with Heterogeneity	yes	yes
Browser Wars	✓	
Enterprise Computing Middleware		✓
Loose Coupling, aspects covered	yes, 2	yes, 3
Time/Availability		✓
Location (Dynamic Late Binding)	(✓)	✓
Service Evolution:		
Uniform Interface	✓	
XML Extensibility	✓	✓
Total Principles Supported	3	3

Table 1: Principles Comparison Summary

©2008 Cesare Pautasso

Comparison Summary

- Architectural Decisions measure complexity implied by alternative technologies

- **REST simplicity** = freedom from choice
  - 5 decisions require to choose among 16 alternatives
  - 12 decisions are already taken (but 5 are *do-it-yourself*)

- **WS-* complexity** = freedom of choice
  - 12 decisions require to choose among 32 alternatives
  - 2 decisions are already taken (SOAP, WSDL+XSD)

©2008 Cesare Pautasso

Conclusion

- You should focus on whatever solution gets the job done and try to **avoid being religious** about any specific architectures or technologies.
- WS-* has strengths and weaknesses and will be highly suitable to some applications and positively terrible for others. Likewise with REST.
- The decision of which to use depends entirely on the application requirements and constraints.
- We hope this comparison will help you make the right choice.

©2008 Cesare Pautasso

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, BPEL for REST, Proc. of the 6th International Conference on Business Process Management (BPM 2008), Milan, Italy, September 2008.

- 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 29-30, 2004.

©2008 Cesare Pautasso

Università della Svizzera italiana

Facoltà di scienze informatiche

REST vs. SOAP: Making the Right Architectural Decision

Cesare Pautasso

Faculty of Informatics

University of Lugano (USI), Switzerland

http://www.pautasso.info

©2008 Cesare Pautasso

Università della Svizzera italiana

Facoltà di scienze informatiche

Backup Material on REST

Cesare Pautasso

Faculty of Informatics

University of Lugano (USI), Switzerland

http://www.pautasso.info

©2008 Cesare Pautasso

REST in one Slide

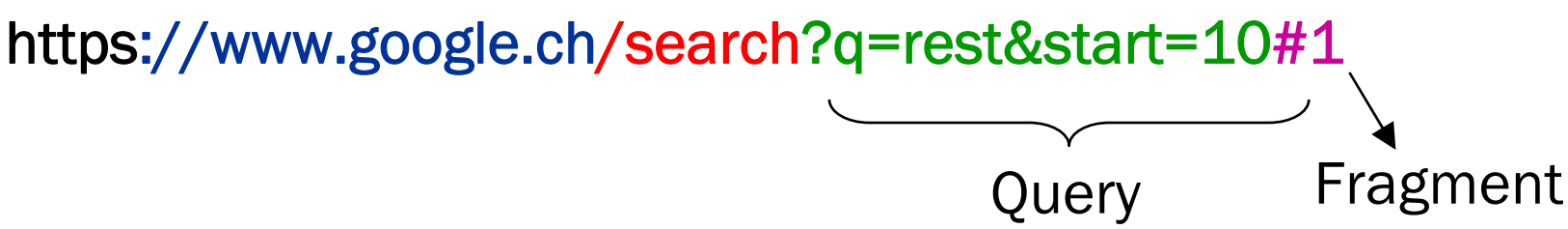
- REpresentational State Transfer defines the architectural style of the World Wide Web
- Its four principles can explain the success and the scalability of the HTTP protocol implementing them

1. **Resource Identification** through URI
2. **Uniform Interface** for all resources:
   - GET (Query the state, idempotent, can be cached)
   - POST (Update a resource or create child resource)
   - PUT (Transfer the state on existing/new resource)
   - DELETE (Delete a resource)
3. **"Self-Descriptive"** Message representations
4. **Hyperlinks** to define relationships between resources and valid state transitions of the service interaction

©2008 Cesare Pautasso

URI: Uniform Resource Identifier

- Internet Standard for resource naming and identification (originally from 1994, revised until 2005)
- Examples:



- REST advocates the use of "nice" URIs
- In most HTTP stacks URIs cannot have arbitrary length (4Kb)

©2008 Cesare Pautasso

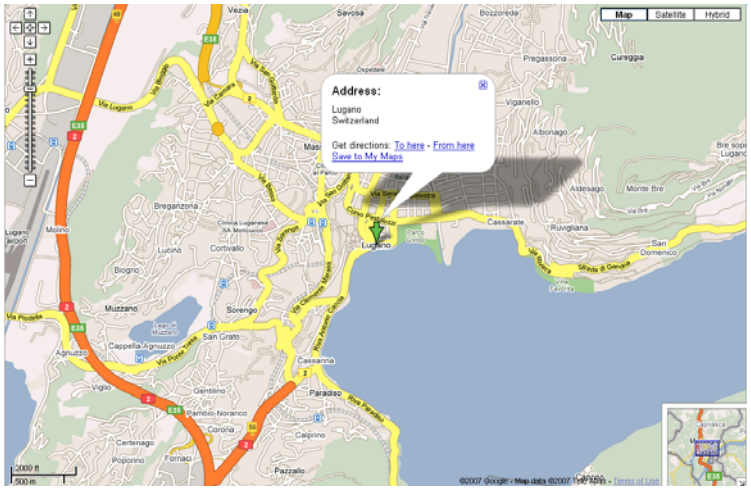
What is a “nice” URI?

Prefer Nouns to Verbs

Keep them Short

http://map.search.ch/lugano

http://maps.google.com/lugano



http://maps.google.com/maps?f=q&hl=en&q=lugano,+switzerland&layer=&ie=UTF8&z=12&om=1&iwloc=addr

©2008 Cesare Pautasso

Uniform Interface Principle (CRUD Example)

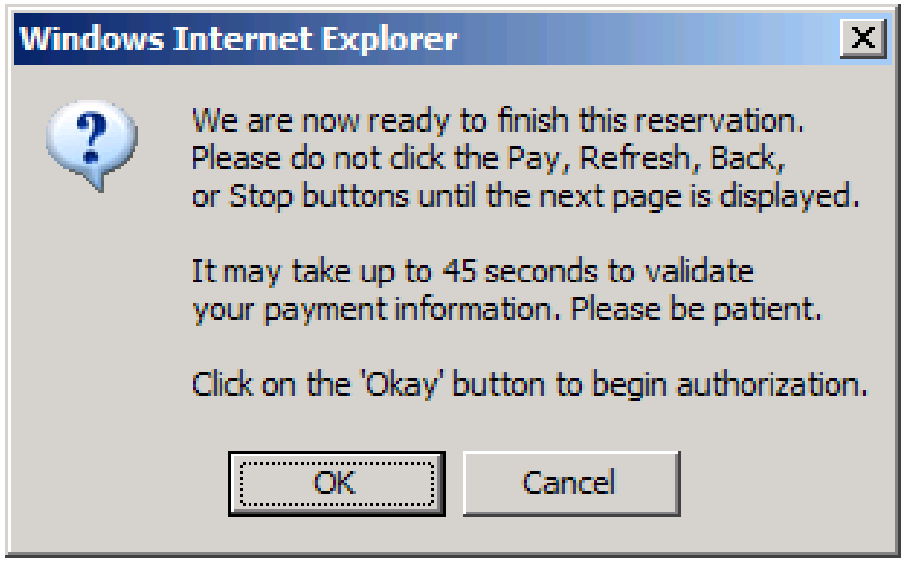
CRUD	REST	
CREATE	POST/PUT	Initialize the state of a new resource at the given URI
READ	GET	Retrieve the current state of the resource
UPDATE	PUT	Modify the state of a resource
DELETE	DELETE	Clear a resource, after the URI is no longer valid

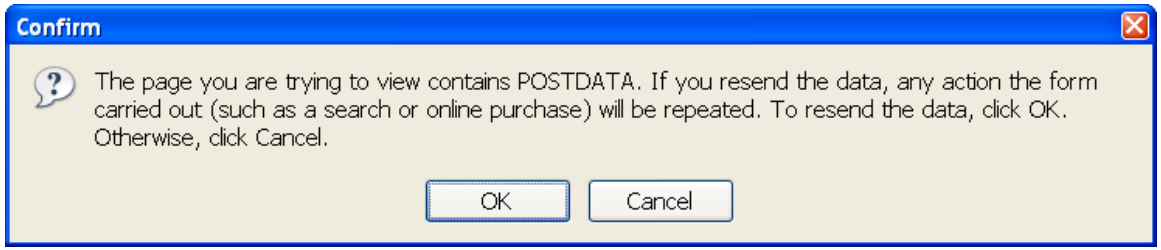
©2008 Cesare Pautasso

POST vs. GET

- GET is a **read-only** operation. It can be repeated without affecting the state of the resource (idem-potent)
- POST is a **read-write** operation and may change the state of the resource and provoke side effects on the server.

  Web browsers warn you when refreshing a page generated with POST





©2008 Cesare Pautasso

RESTful Web Services Design

1. Identify resources to be exposed as services (e.g., yearly risk report, book catalog, purchase order, open bugs)
2. Define "nice" URLs to address them
3. **Understand what it means to do a GET, POST, PUT, DELETE on a given resource URI**
4. Design and document resource representations (payload formats)
5. Model relationships (e.g., containment, reference, state transitions) between resources with hyperlinks that can be followed to get more details
6. Implement and deploy on Web server
7. Test with a Web browser

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

©2008 Cesare Pautasso

Università della Svizzera italiana | Facoltà di scienze informatiche

Resource Representation Formats: XML vs JSON

- XML
  - PO-XML
  - SOAP (WS-*)
  - RSS, ATOM
- Standard textual syntax for semi-structured data
- Many tools available: XML Schema, DOM, SAX, XPath, XSLT, XQuery
- Everyone can parse it (not necessarily understand it)
- Slow and Verbose

- JavaScript Object Notation (JSON)
- Wire format introduced for AJAX Web applications (Browser-Web Server communication)
- Textual syntax for serialization of non-recurrent data structures
- Supported in most languages (not only JavaScript)
- Not extensible (does not need to be)
- "JSON has become the X in Ajax"

©2008 Cesare Pautasso