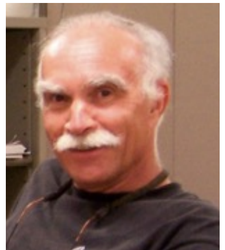
Reflections on the REST Architectural Style and “Principled Design of the Modern Web Architecture”

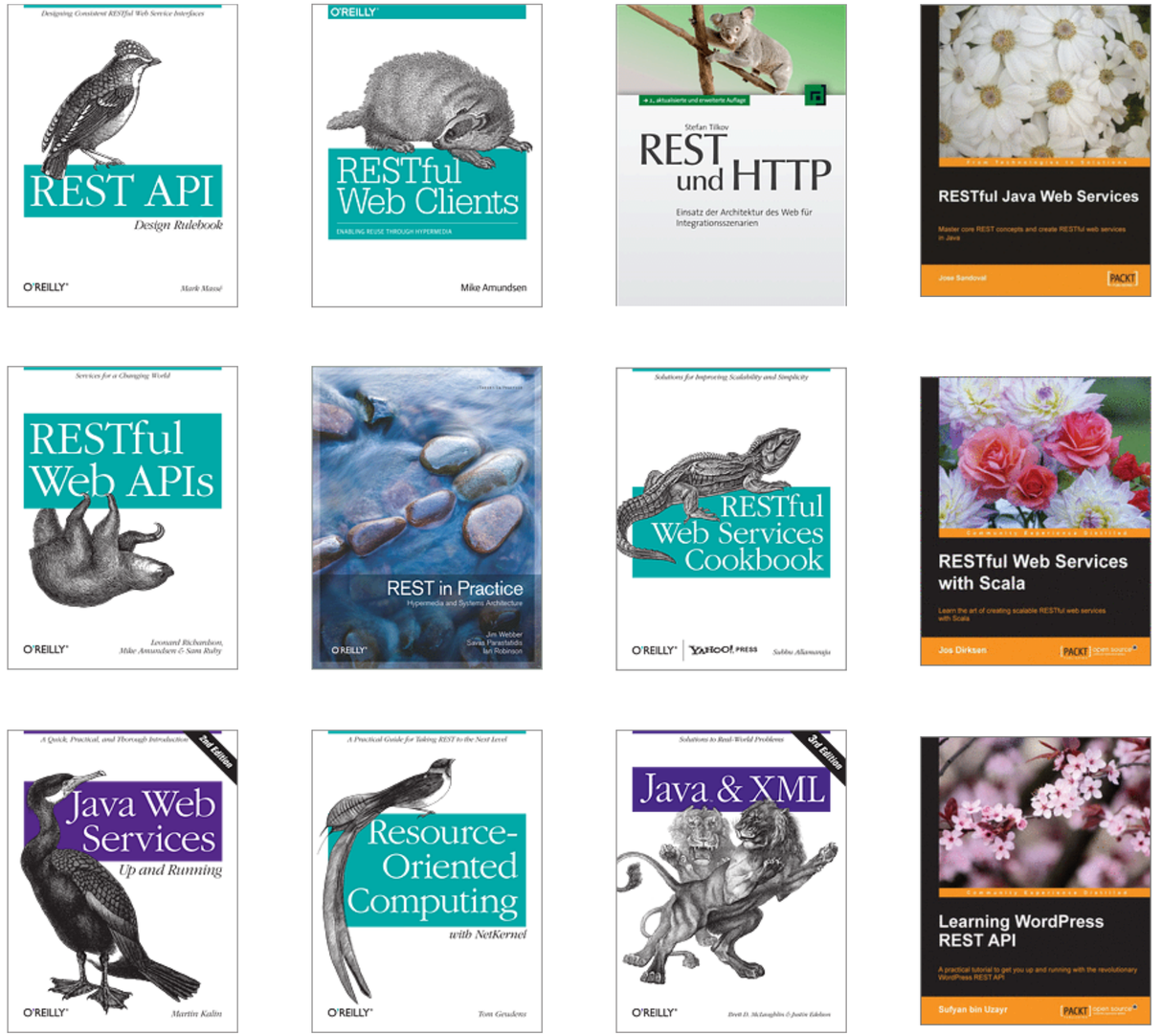
Roy T. Fielding	Adobe
Richard N. Taylor	University of California, Irvine
Justin R. Erenkrantz	Bloomberg
Michael M. Gorlick	University of California, Irvine
Jim Whitehead	University of California, Santa Cruz
Rohit Khare	Google
Peyman Oreizy	Dynamic Variable LLC



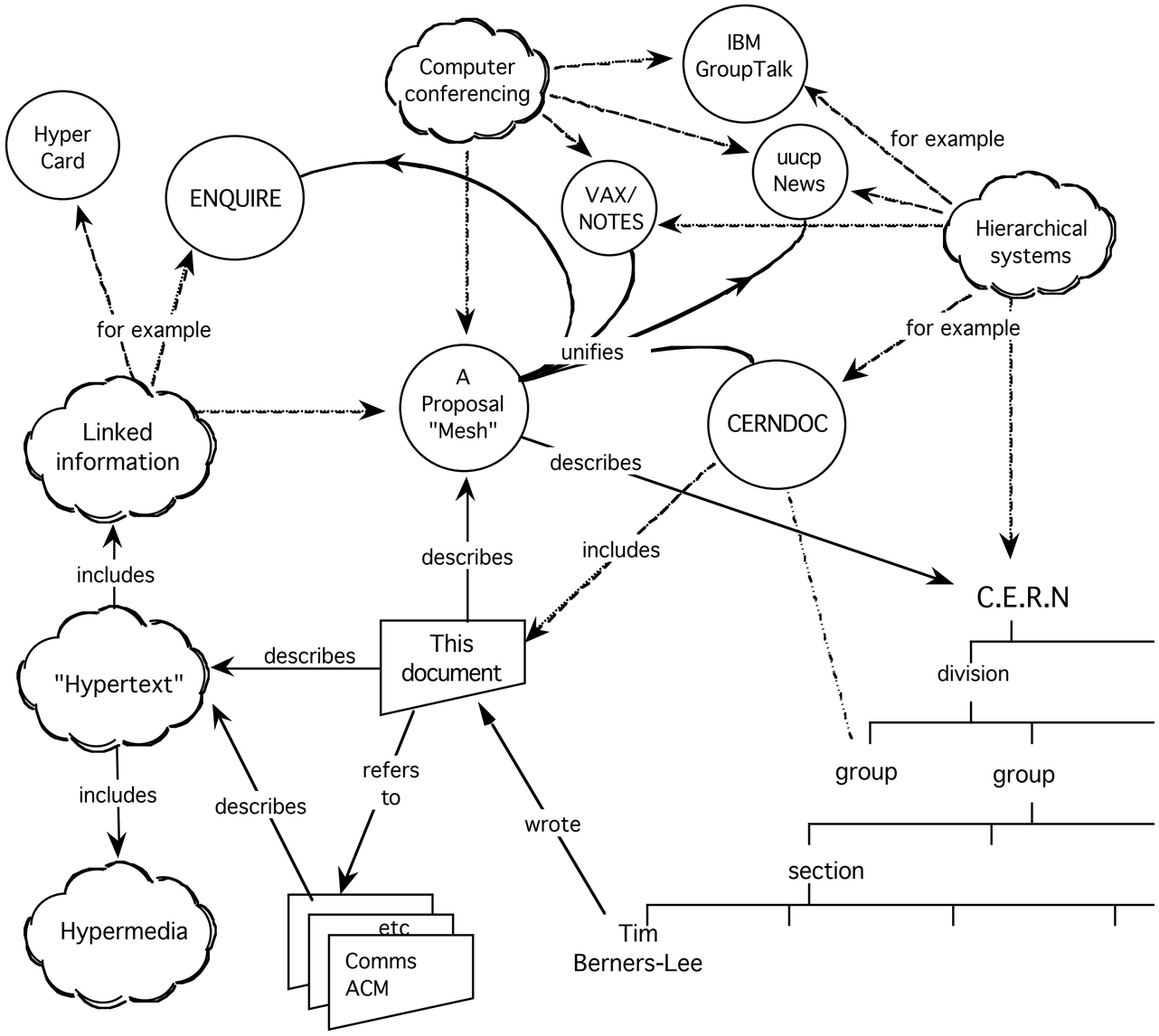




Outline

1. The Story of REST
   - Early history of the Web
   - What REST is (and is not)
   - Contemporary influences
2. Work inspired by REST
   - Decentralization
   - Generalization
   - Secure computation
3. Reflections on REST
   - Investing in entrepreneurial students
   - Role of Software Engineering research

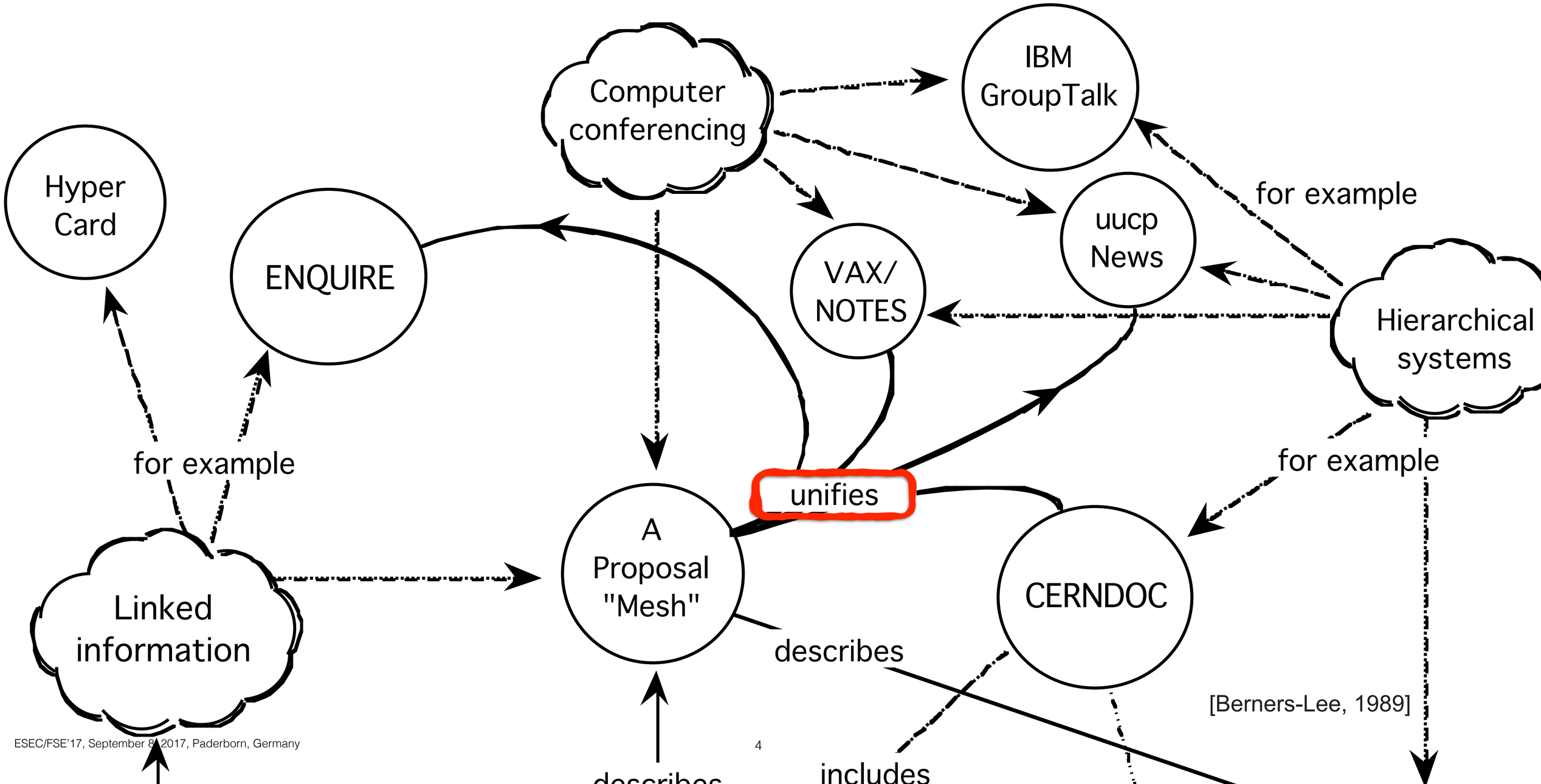


Original proposal for the World Wide Web



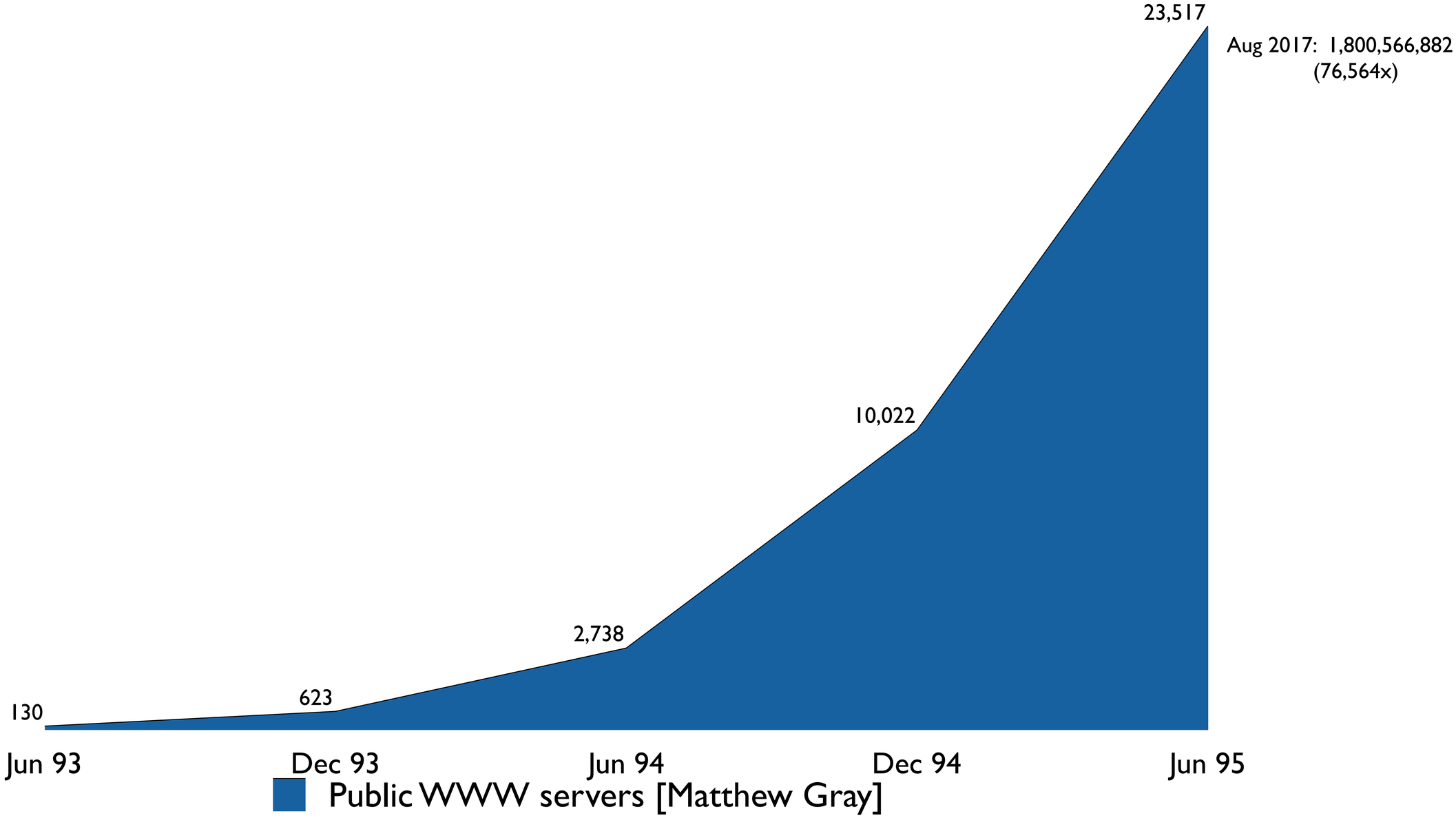
[Berners-Lee, 1989]

The Web is an application integration system

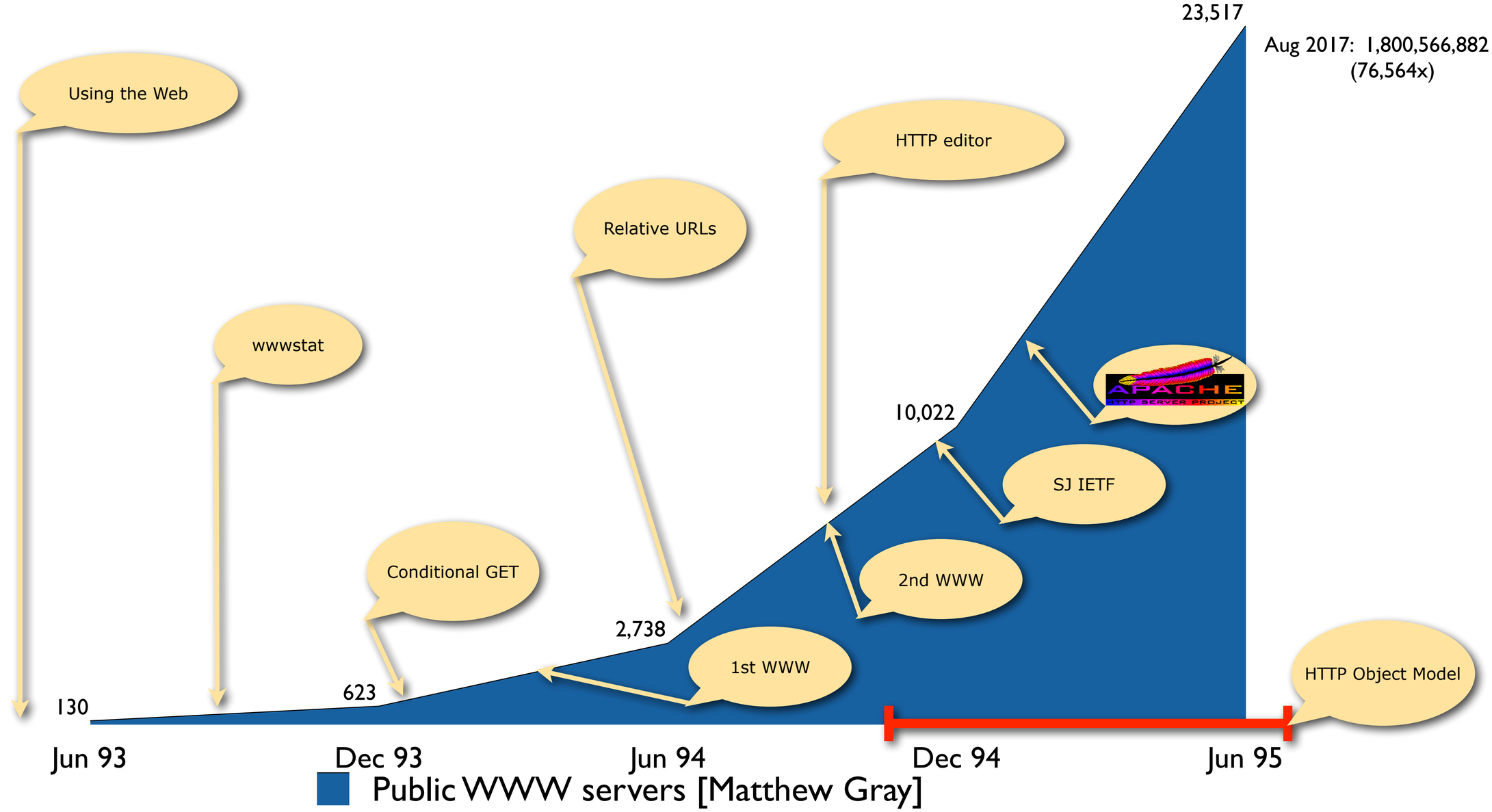


[Berners-Lee, 1989]

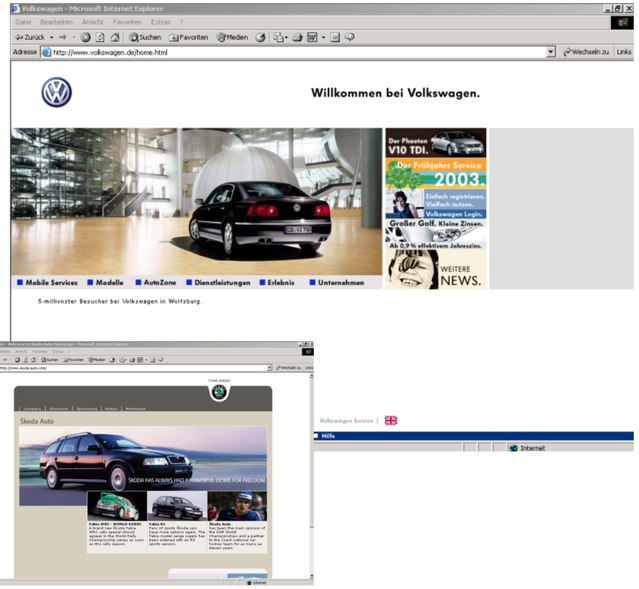
A bit of context



A bit of context

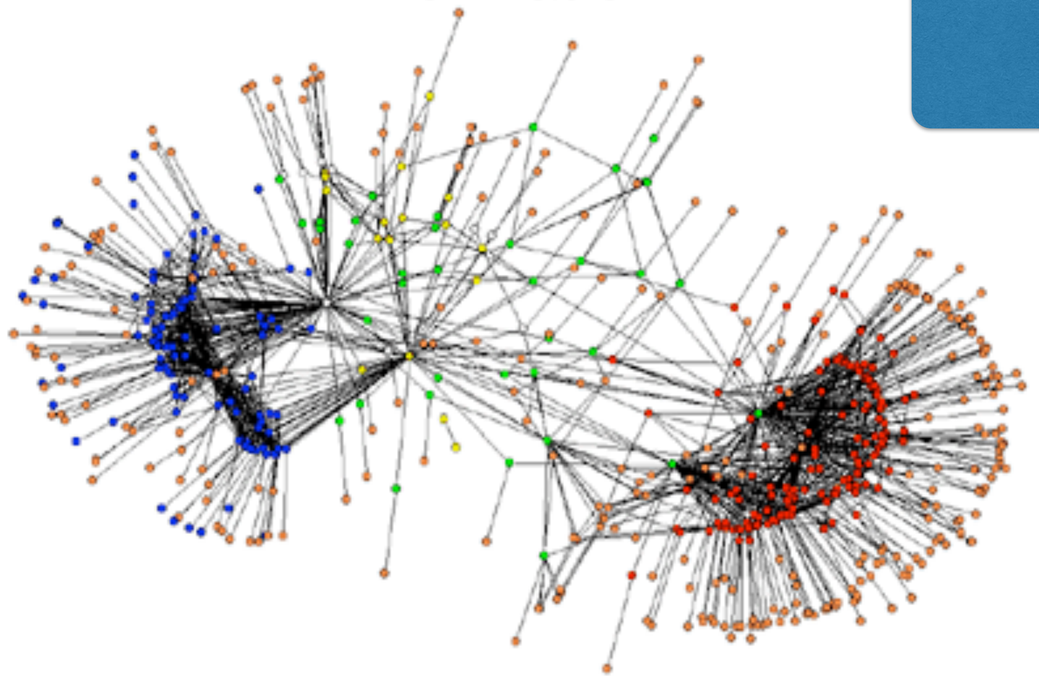


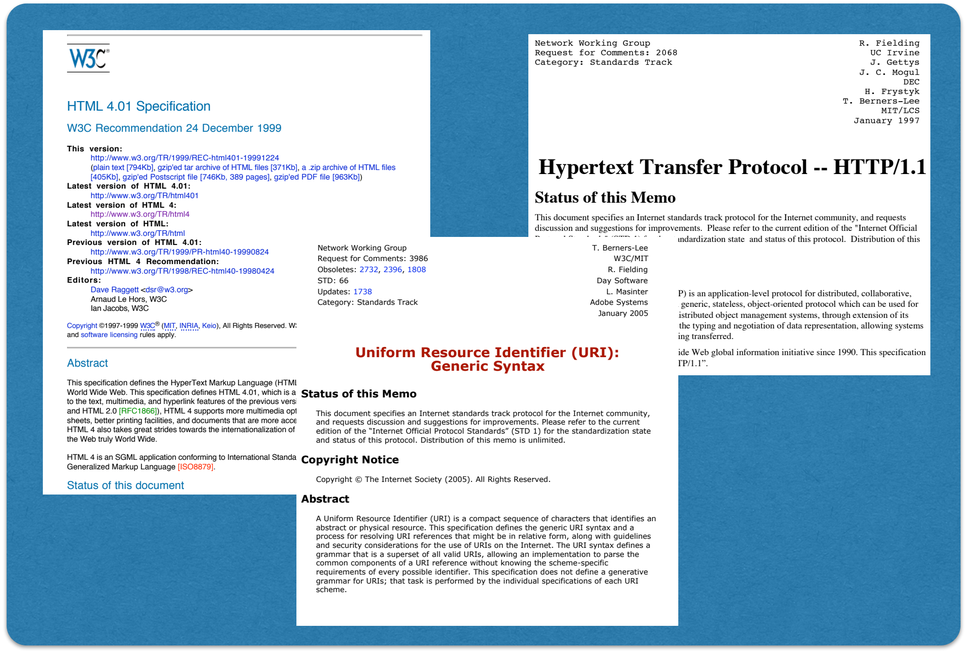
Three (very different) perspectives of the Web



Browsers

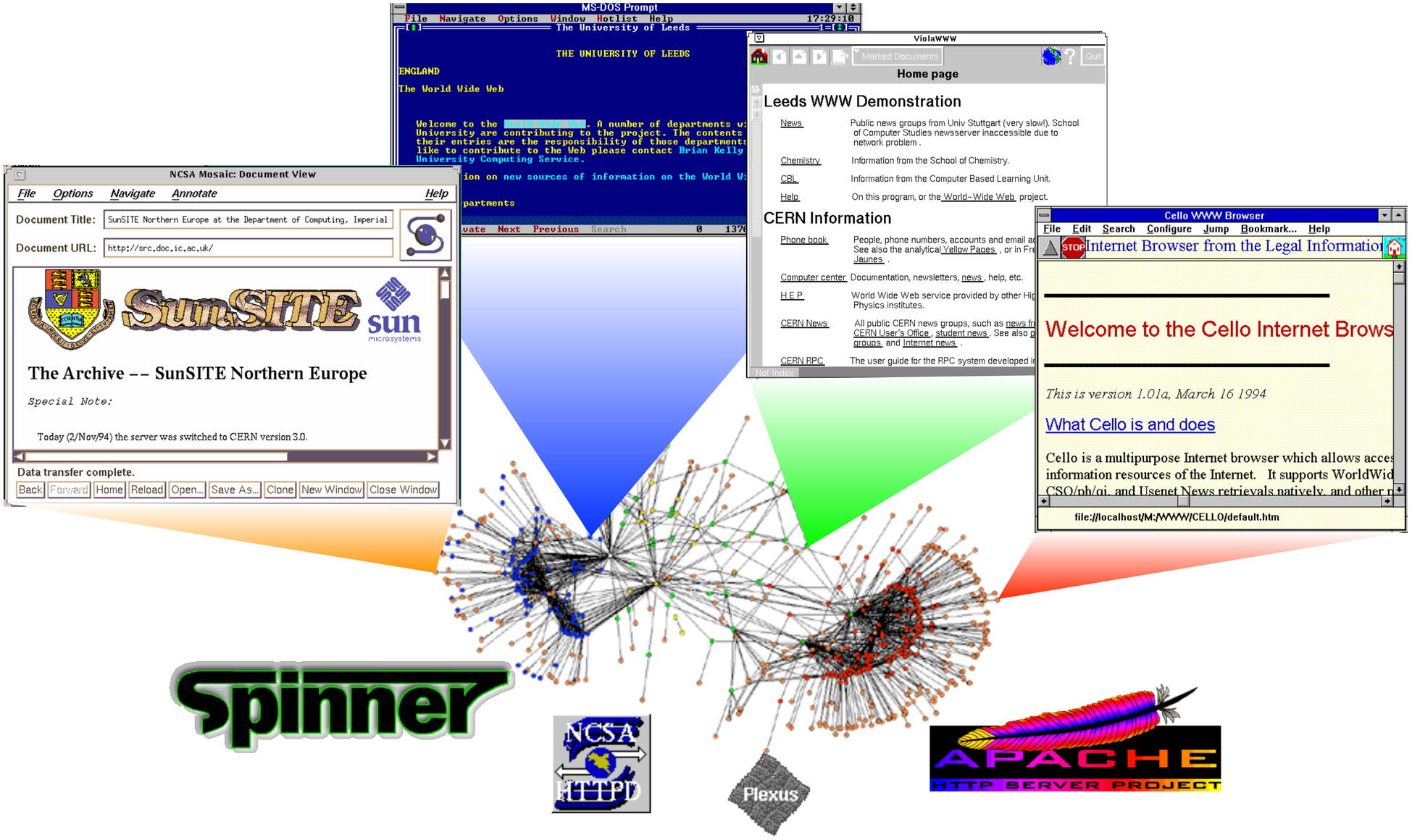
Information



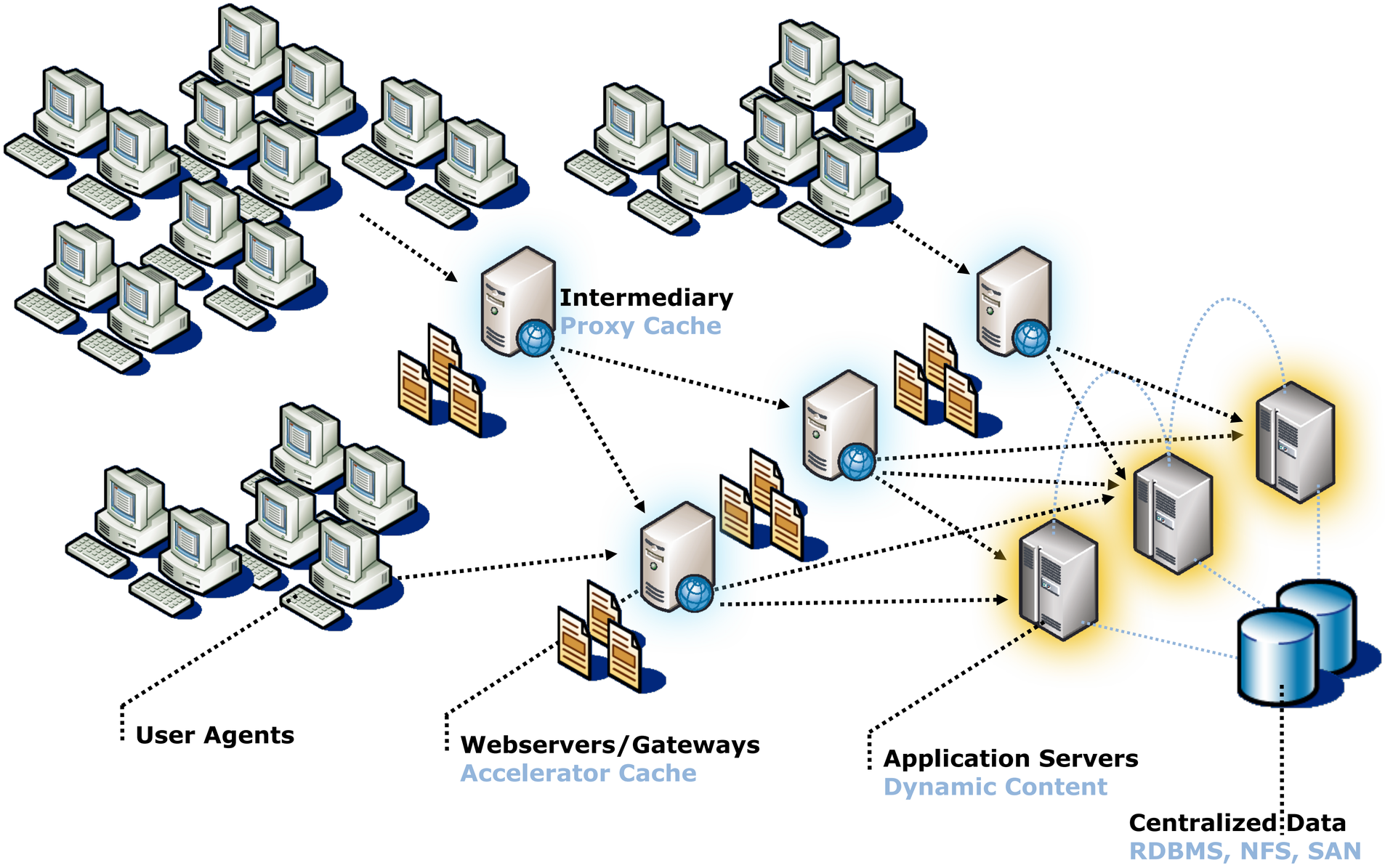


Protocols

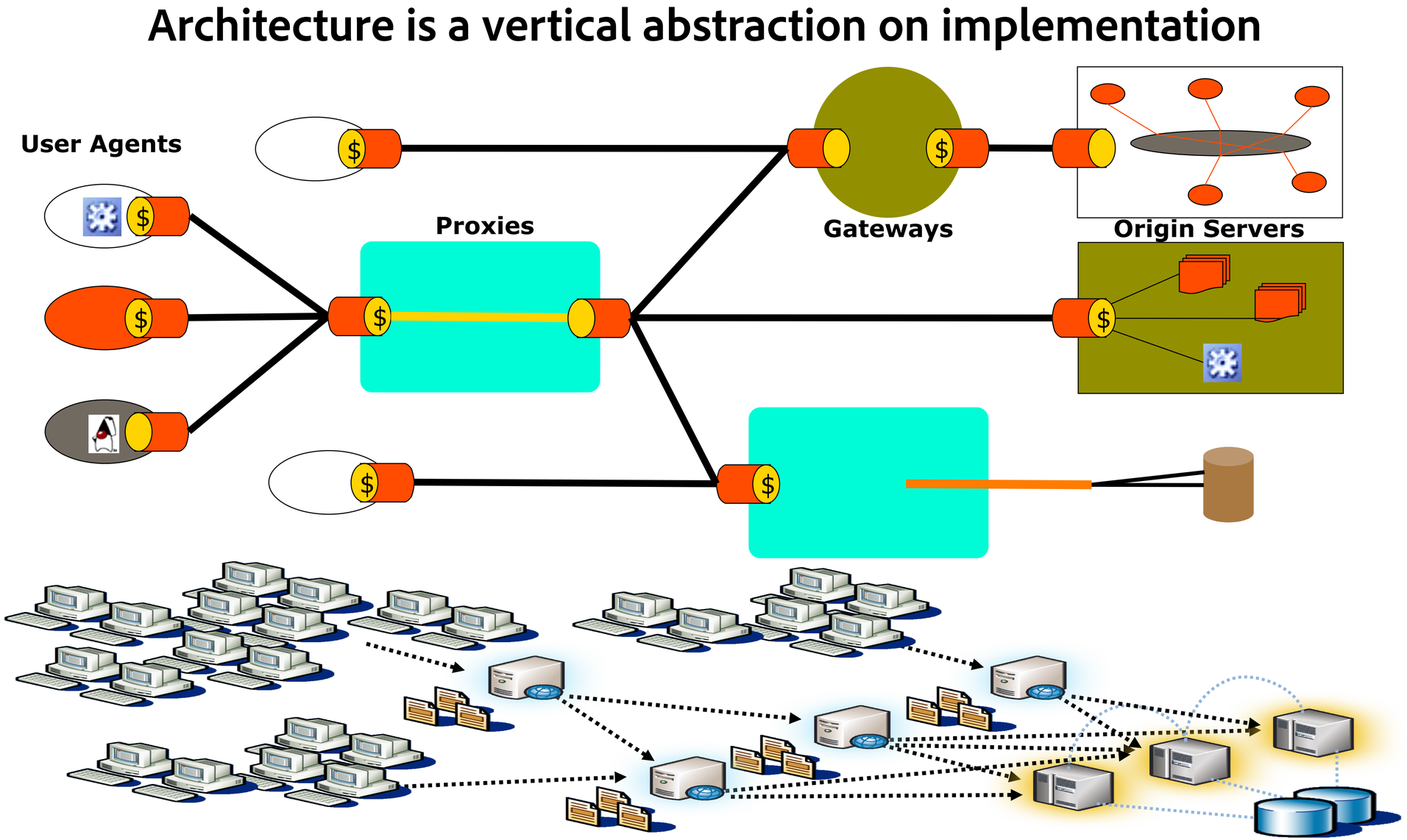
Web Implementation (user view)



Web Implementation (origin view)

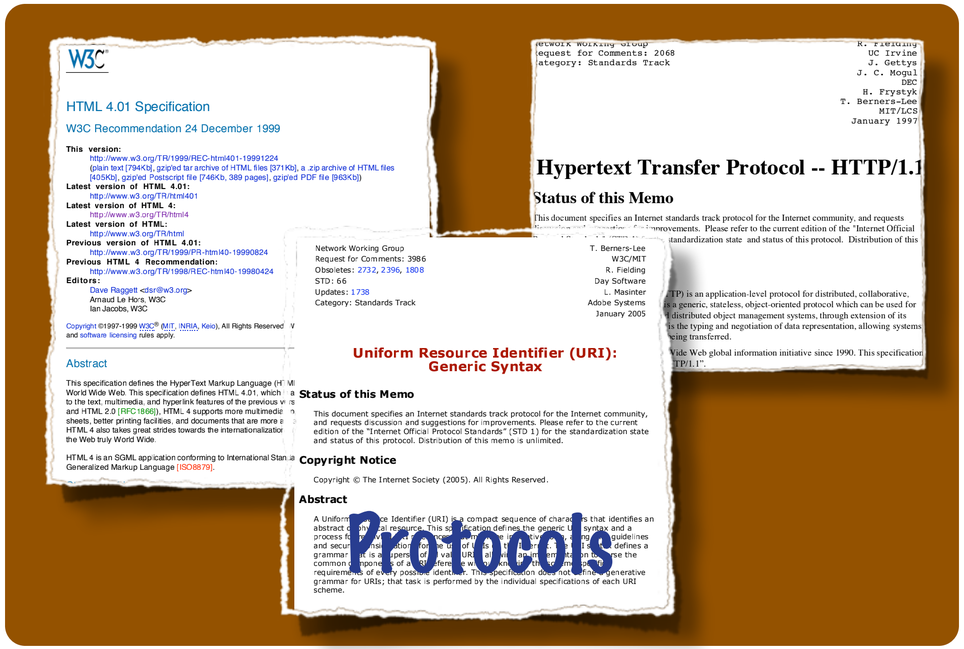


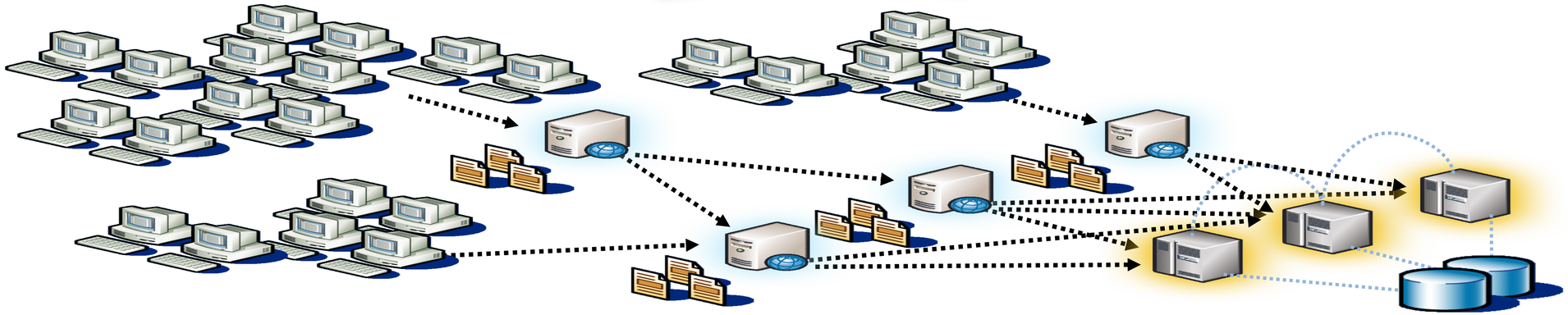
Web Architecture



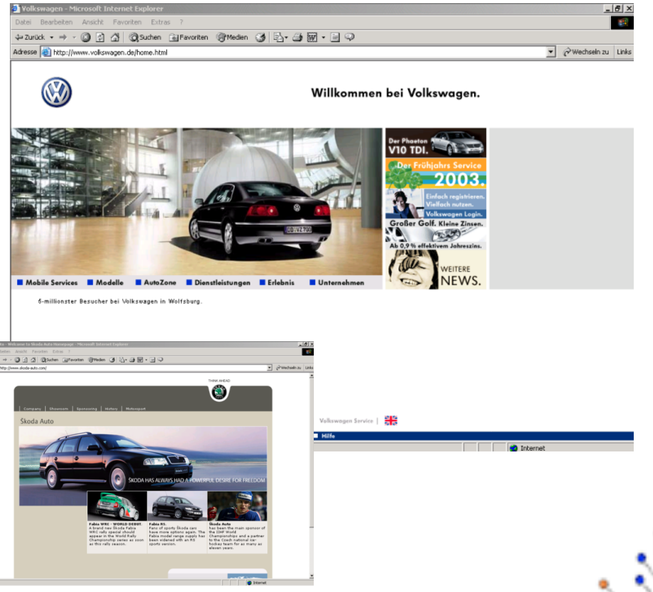
Web Architecture

Web protocols define that vertical abstraction on implementation



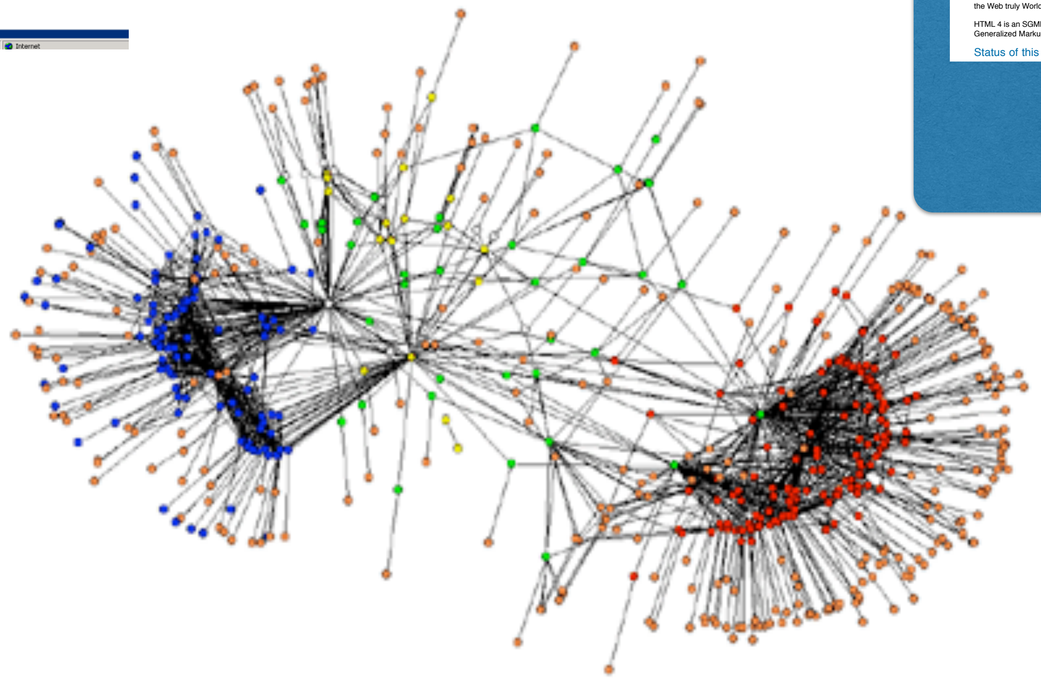


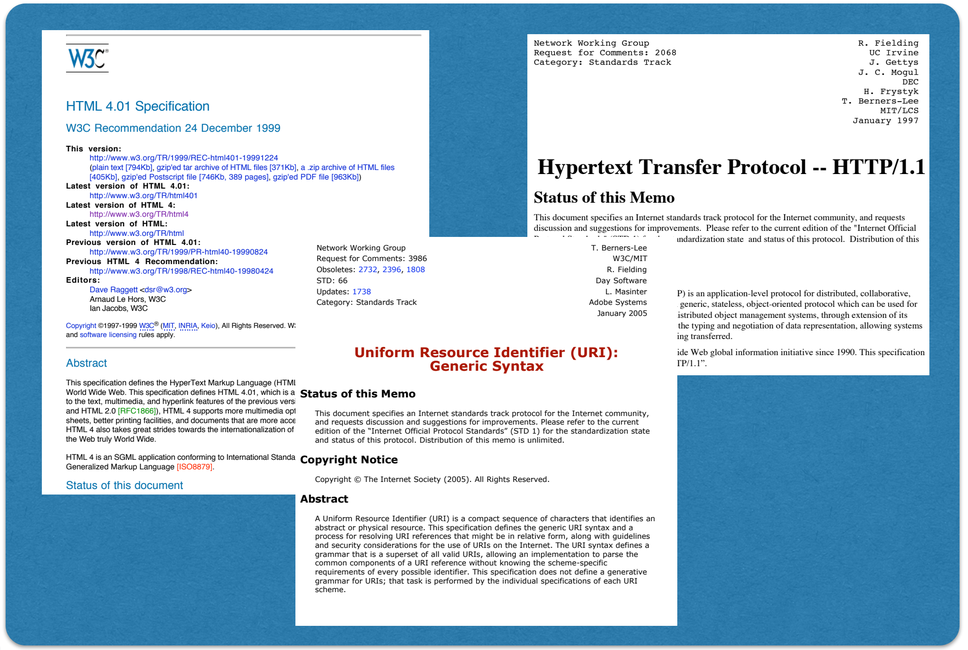
So, which one is the Web?



Browsers

Information





Protocols

So, which one is the Web?

All of them!

The Web is a World-Wide System:
constantly running,
always changing,
anarchically accessed, and
independently deployed.

Outline

1. The Story of REST
   - Early history of the Web
   - What REST is (and is not)
   - Contemporary influences
2. Work inspired by REST
   - Decentralization
   - Generalization
   - Secure computation
3. Reflections on REST
   - Investing in entrepreneurial students
   - Role of Software Engineering research

Roy T. Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. Dissertation. University of California, Irvine, California, USA. http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

Roy T. Fielding and Richard N. Taylor. 2000. Principled Design of the Modern Web Architecture. In *Proceedings of the 22nd Int'l Conference on Software Engineering*. IEEE, Limerick, Ireland, 407–416.

Roy T. Fielding and Richard N. Taylor. 2002. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology* 2, 2 (May 2002), 115–150.

Why talk about my definition of REST?

Because

REST

has become a

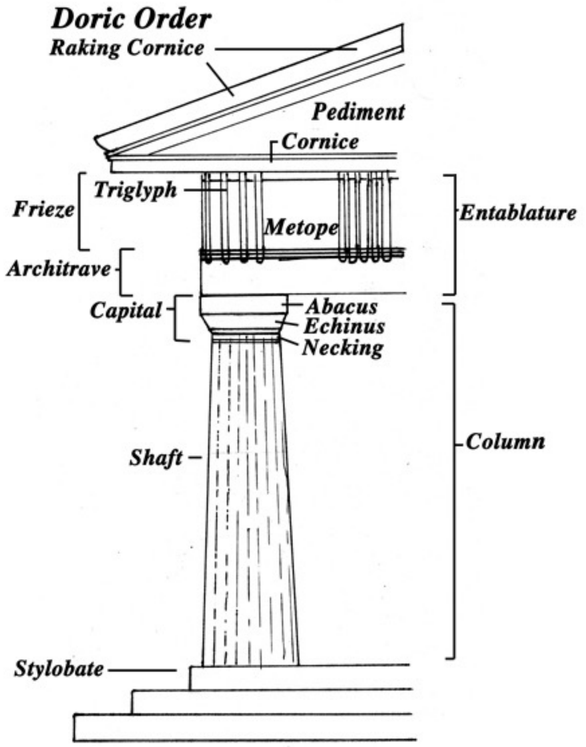
BUZZWORD

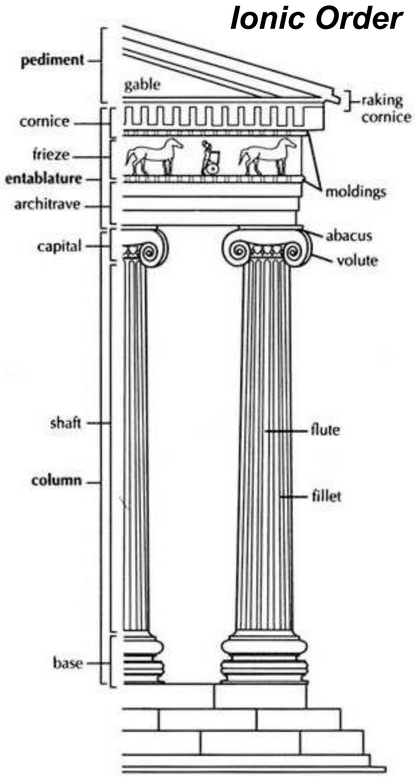
There's nothing particularly wrong with that…
unless you happen to be me…
or working with me

Architectural Styles

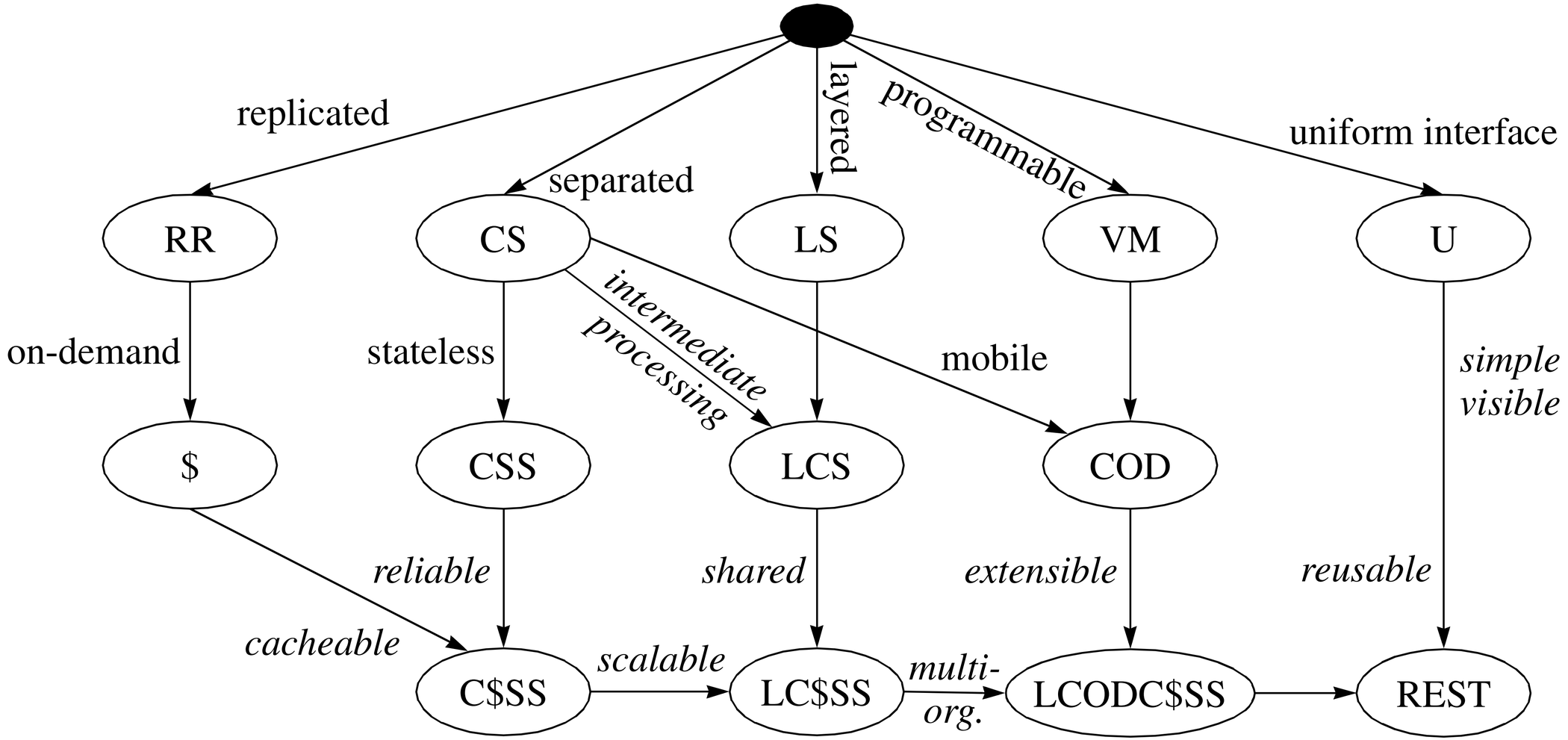
- A **horizontal abstraction** across multiple architectures (vertical abstractions)
  - names a repeated architectural pattern
  - defined by its design constraints
  - chosen for the properties they induce

- **REST is an architectural style**
  - for network-based applications
  - to induce a specific set of architectural properties
  - that were desired for the World Wide Web

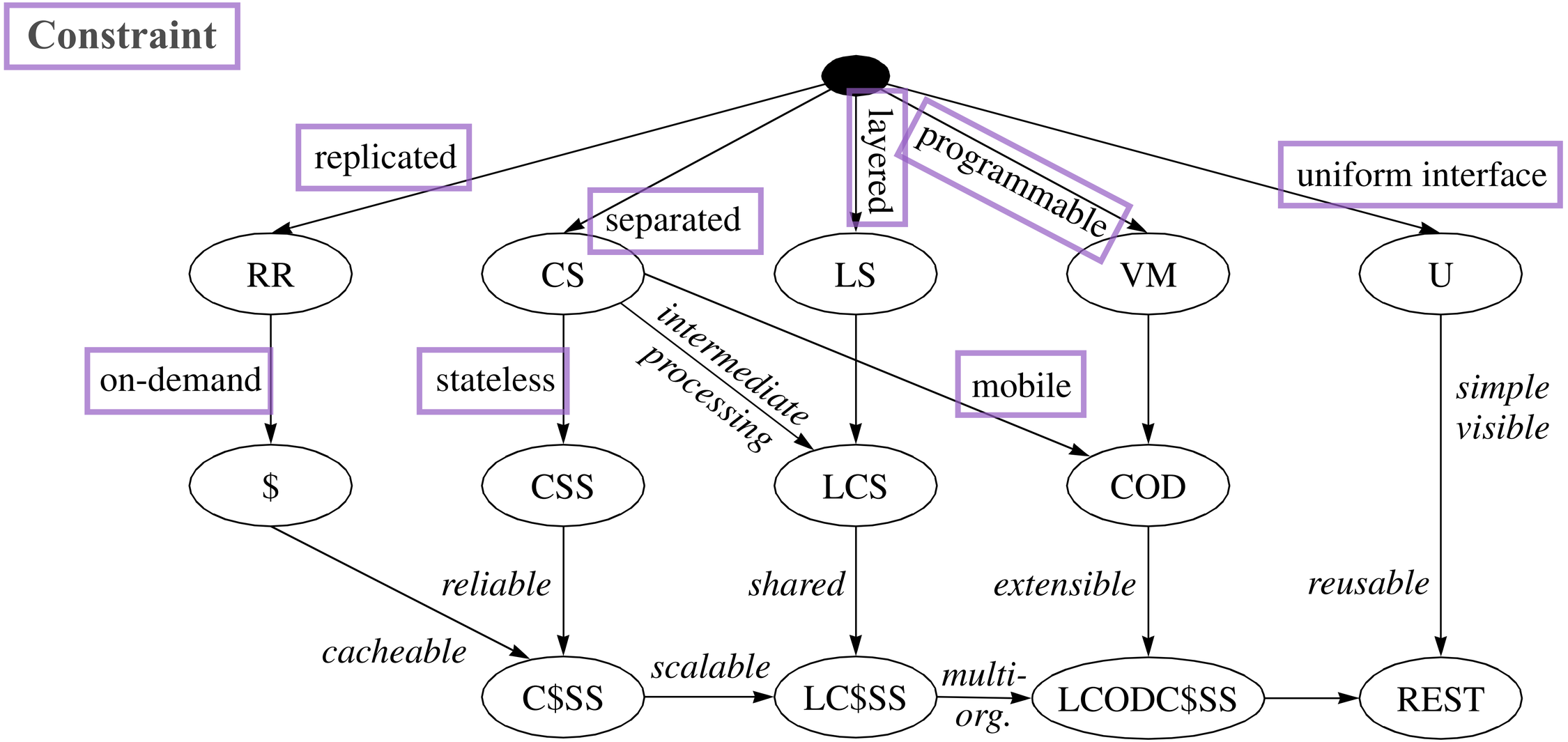




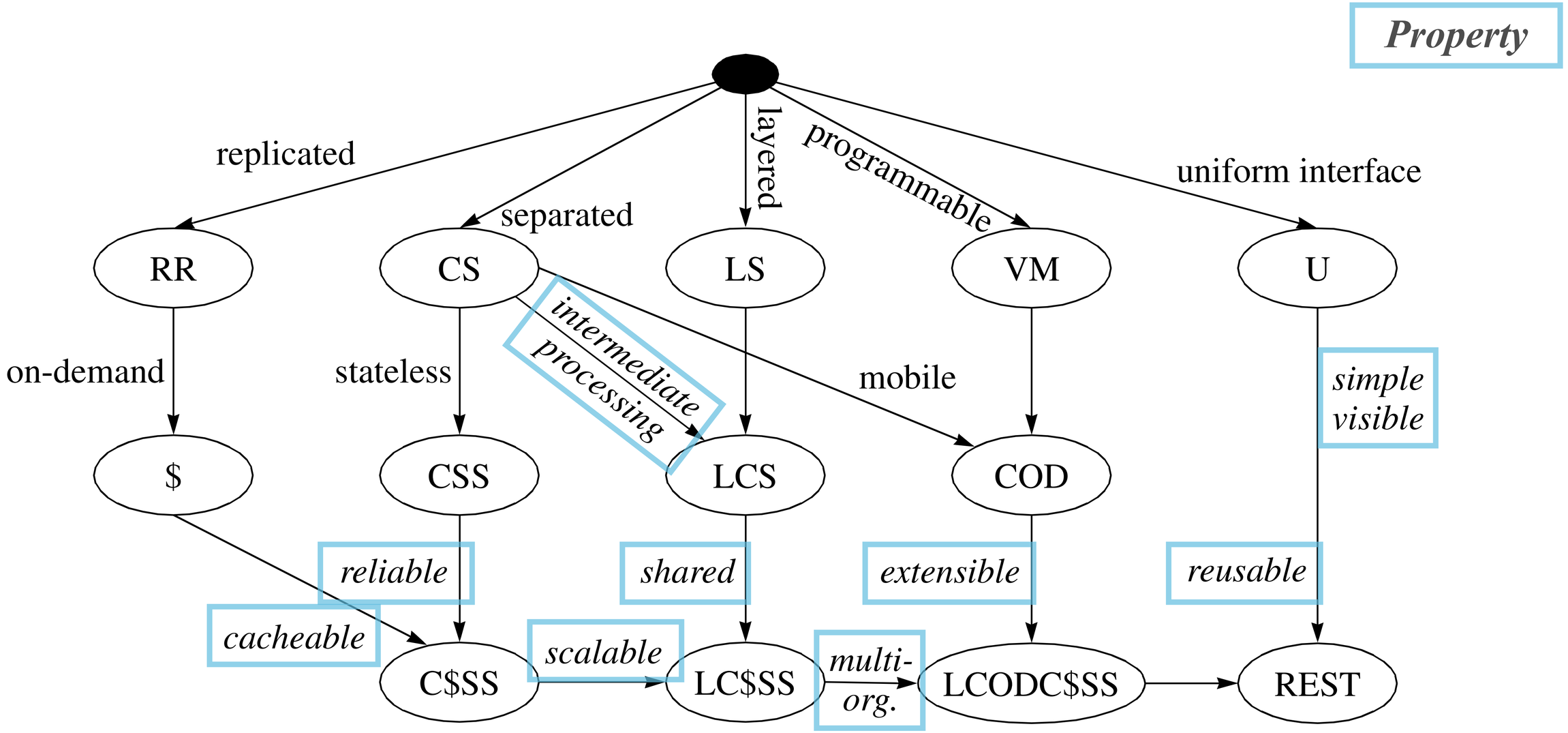
REST is an accumulation of design constraints

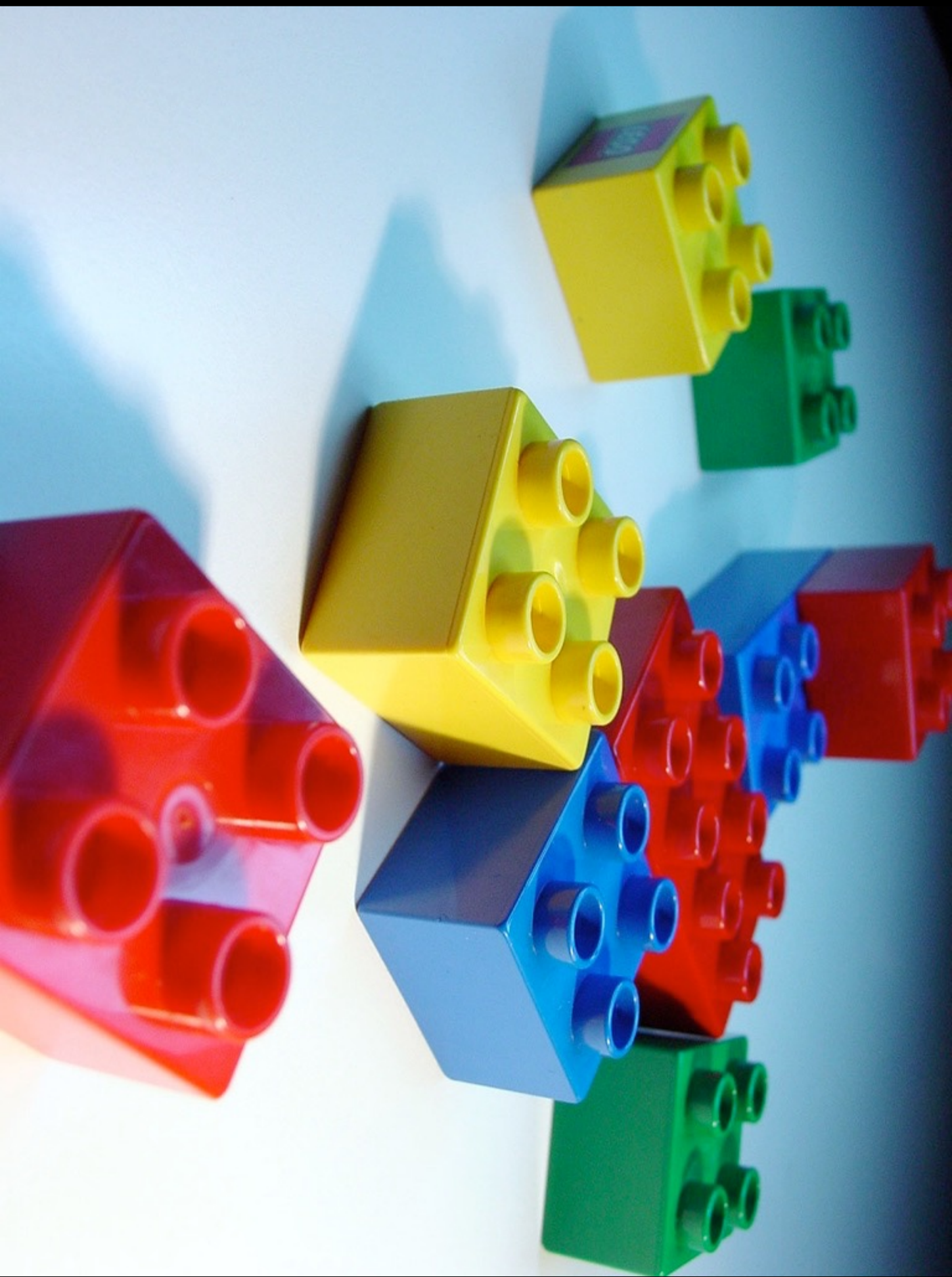


REST is an accumulation of design constraints



REST is an accumulation of design constraints





[photo by dhester: http://mrg.bz/xVLmr1]

[photo by EmmiP: http://mrg.bz/P7BJRi]

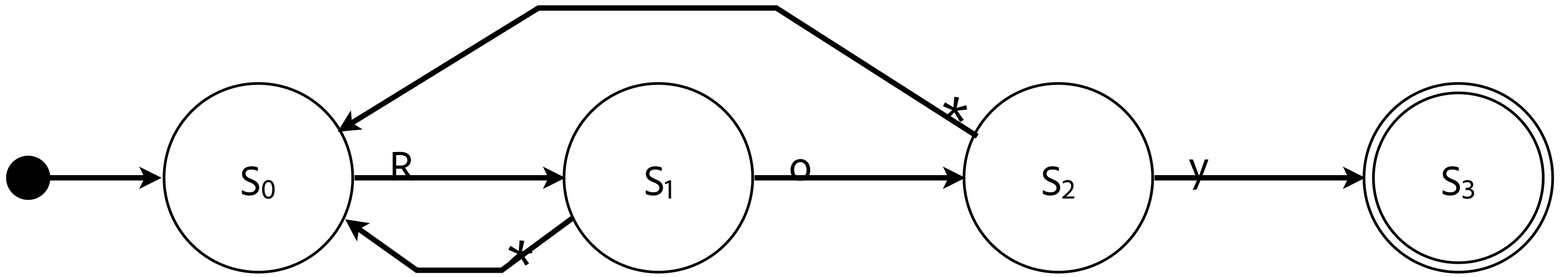
[photo by rupertjefferies: http://mrg.bz/Y9XThf]

REST's Five Uniform Interface Constraints

1. All important resources are identified by one resource identifier mechanism
   - induces simple, visible, reusable, stateless communication
2. Access methods have the same semantics for all resources
   - induces visible, scalable, available through layered system, cacheable, and shared caches
3. Resources are manipulated through the exchange of representations
   - induces simple, visible, reusable, cacheable, and evolvable (information hiding)
4. Representations are exchanged via self-descriptive messages
   - induces visible, scalable, available through layered system, cacheable, and shared caches
   - induces evolvable via extensible communication
5. Hypertext as the engine of application state
   - induces simple, visible, reusable, and cacheable through data-oriented integration
   - induces evolvable (loose coupling) via late binding of application transitions

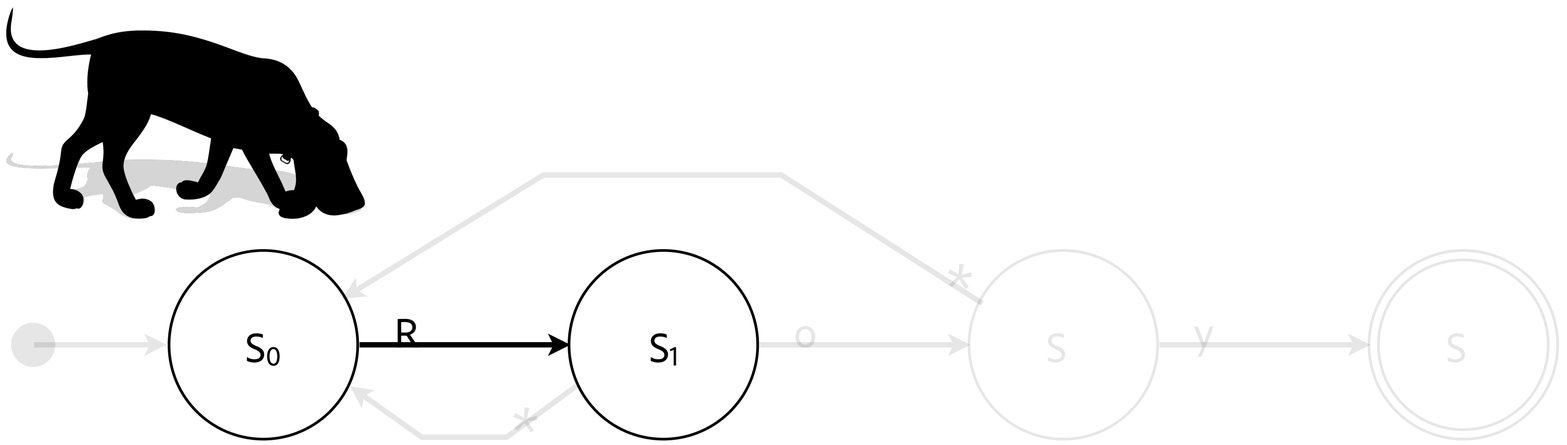
Deep dive into the hypermedia constraint

Hypertext as the Engine of Application State



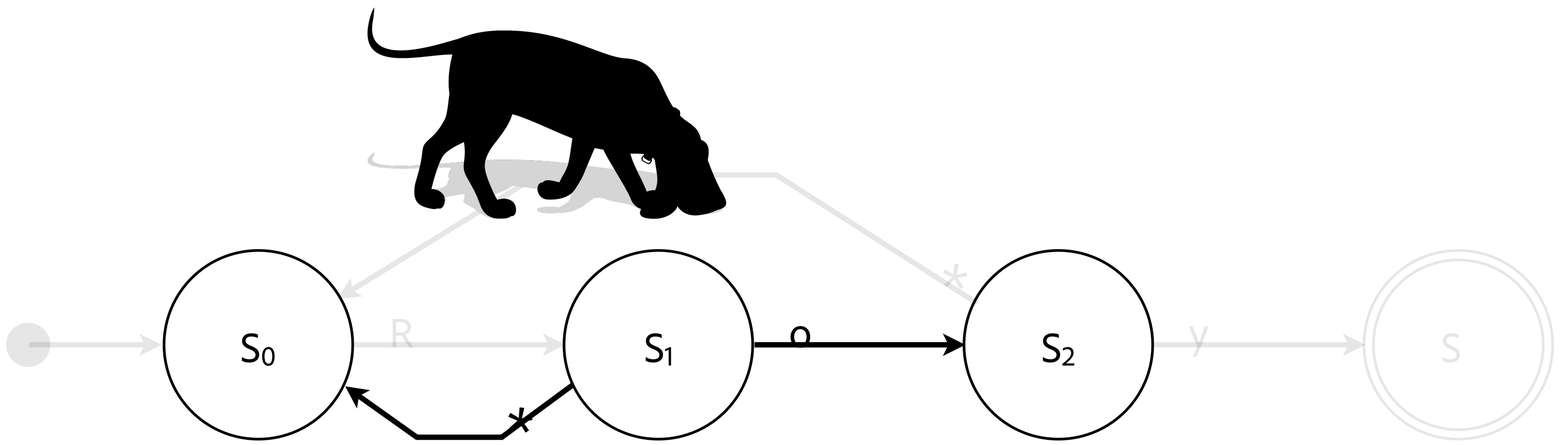
each state can be dynamic
each transition can be redirected

The client only needs to know one state and its transitions!



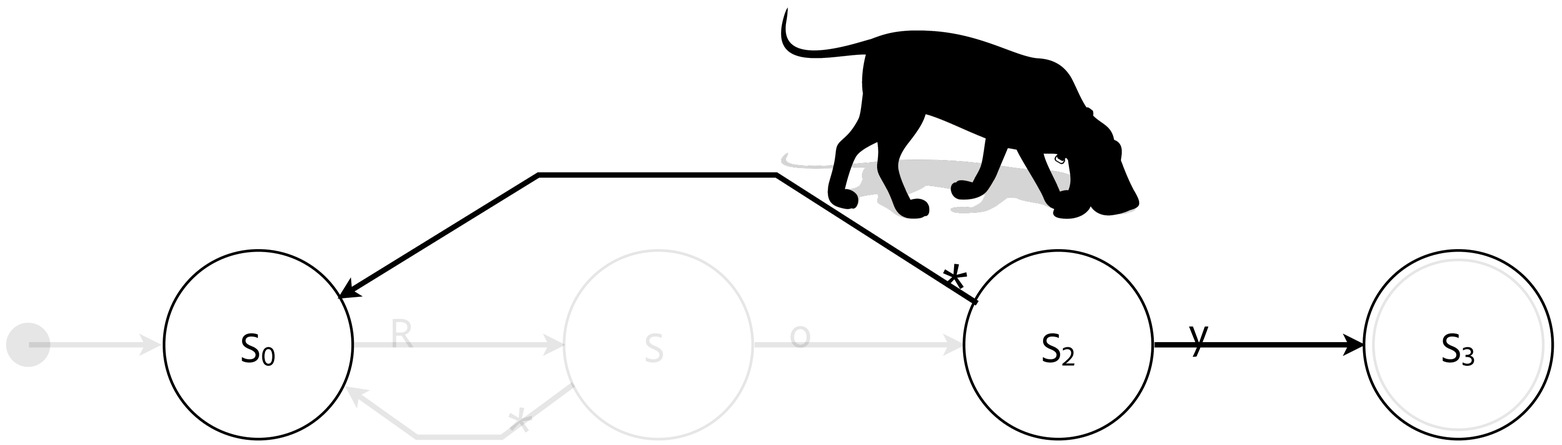
Follow Your Nose

The client only needs to know one state and its transitions!



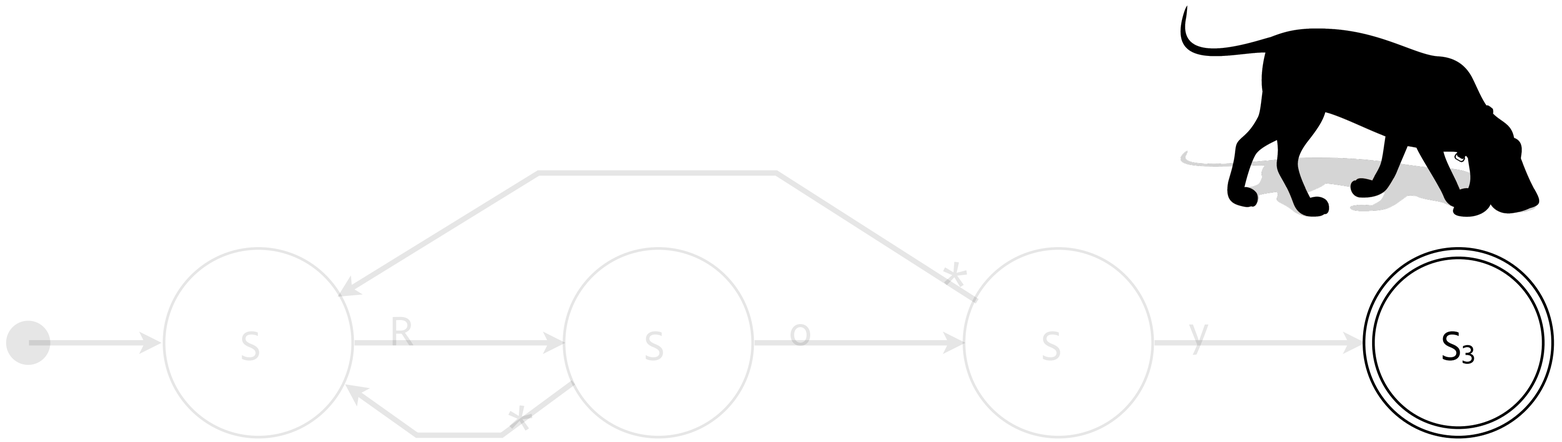
Follow Your Nose

The client only needs to know one state and its transitions!



Follow Your Nose

The client only needs to know one state and its transitions!



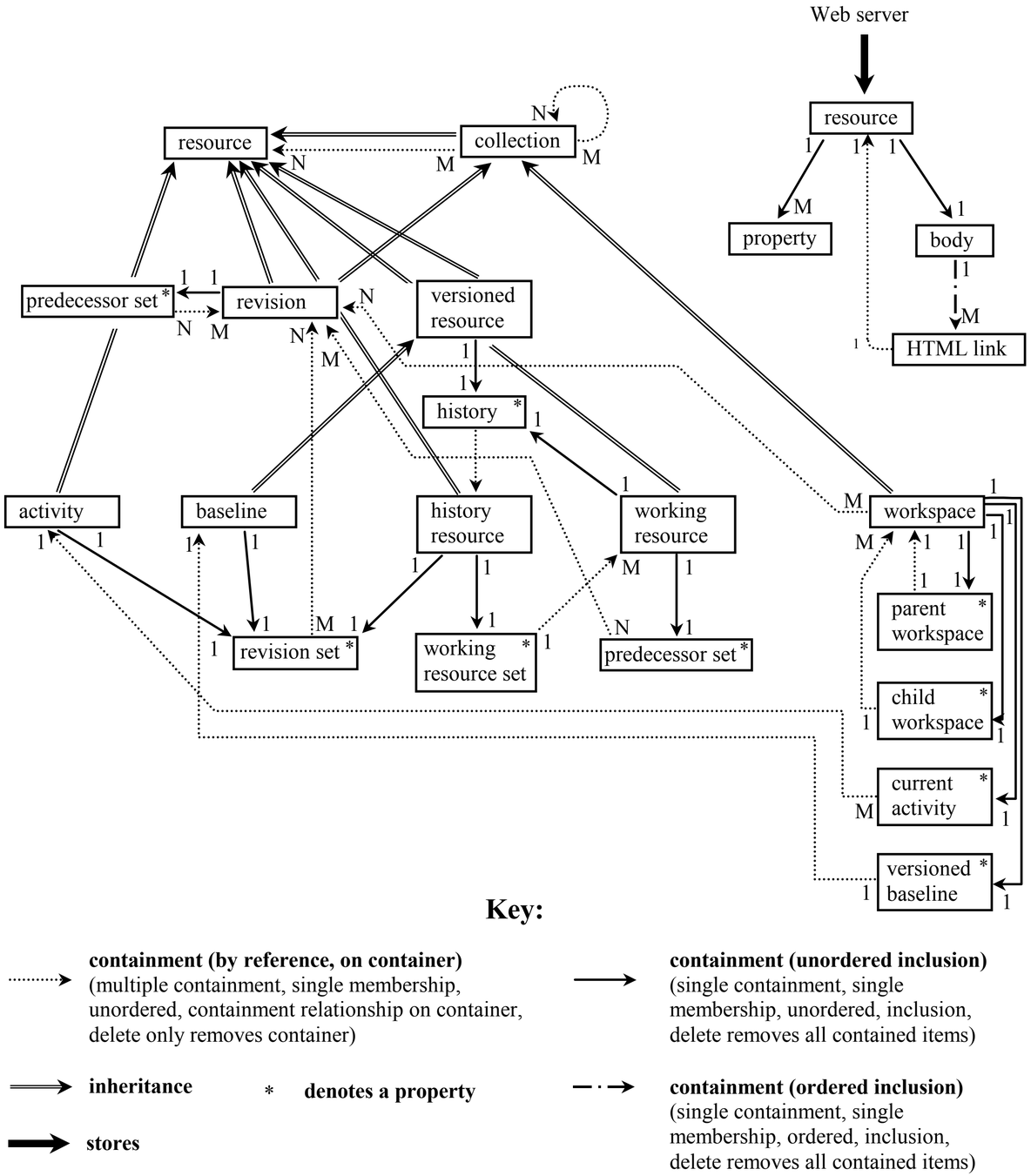
Follow Your Nose

Outline

1. The Story of REST
   - Early history of the Web
   - What REST is (and is not)
   - Contemporary influences
2. Work inspired by REST
   - Decentralization
   - Generalization
   - Secure computation
3. Reflections on REST
   - Investing in entrepreneurial students
   - Role of Software Engineering research

WebDAV

- Distributed Authoring and Versioning
  - Returning to the Web's roots
  - Resources vs Representations vs Metadata
  - Stateless Interaction vs Session Locks
- Overwhelmed by commercial demands
  - XML, Locks, remote data model
  - Moved away from REST constraints, but helped enlighten and refine them



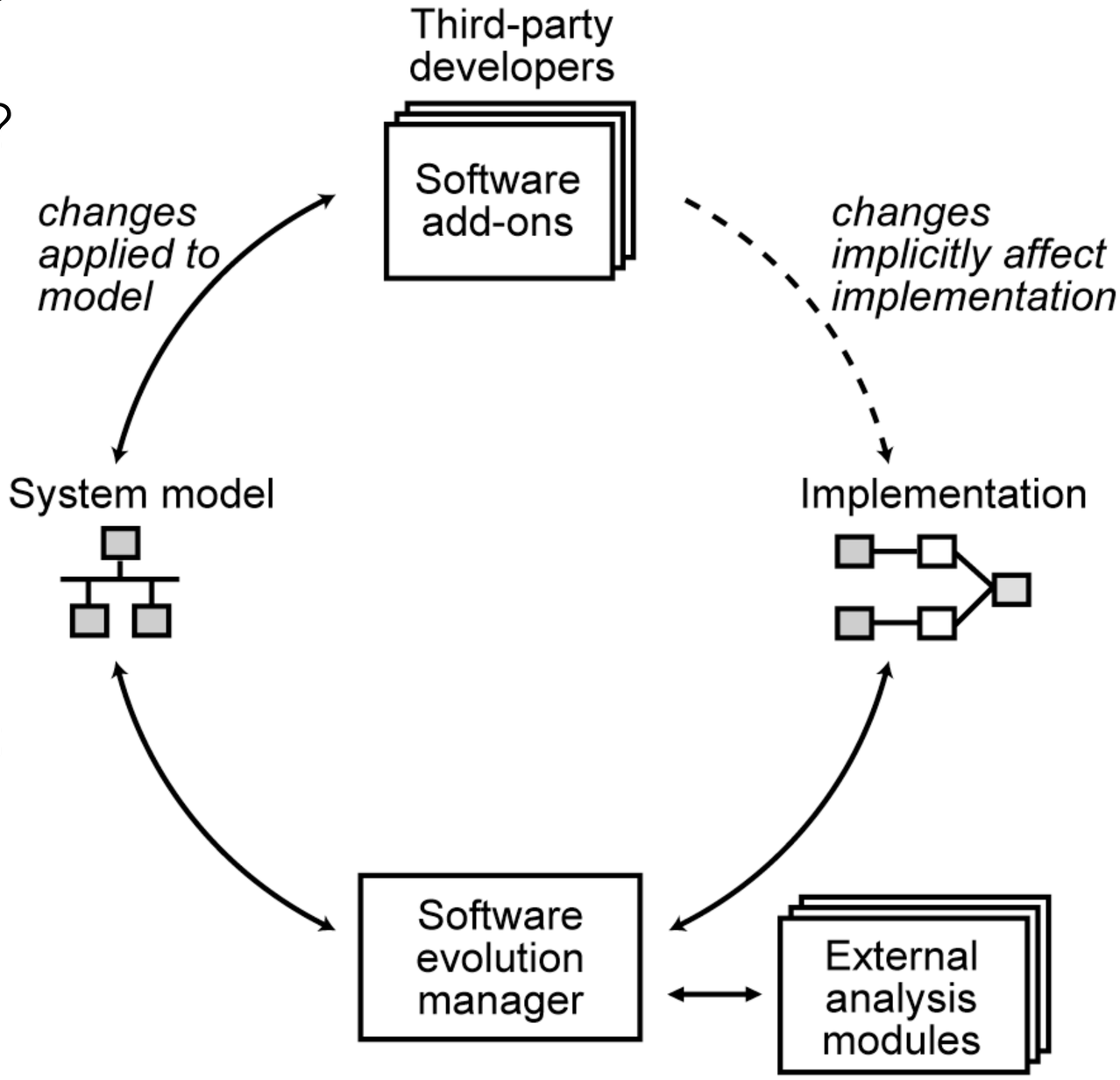
E. James Whitehead, Jr. "World Wide Web Distributed Authoring and Versioning (WebDAV): An Introduction." *StandardView*, Vol. 5, No. 1, March 1997, pages 3-8.

Y. Goland, E. Whitehead, A. Faizi, S. Carter, D. Jensen, *HTTP Extensions for Distributed Authoring - WEBDAV.* Microsoft, U.C. Irvine, Netscape, Novell, Internet Proposed Standard RFC 2518. February, 1999.

E. James Whitehead, Jr. and Yaron Goland. 2004. The WebDAV Property Design. *Software, Practice and Experience* 34 (2004), 135–161.

Dynamic Software Architectures

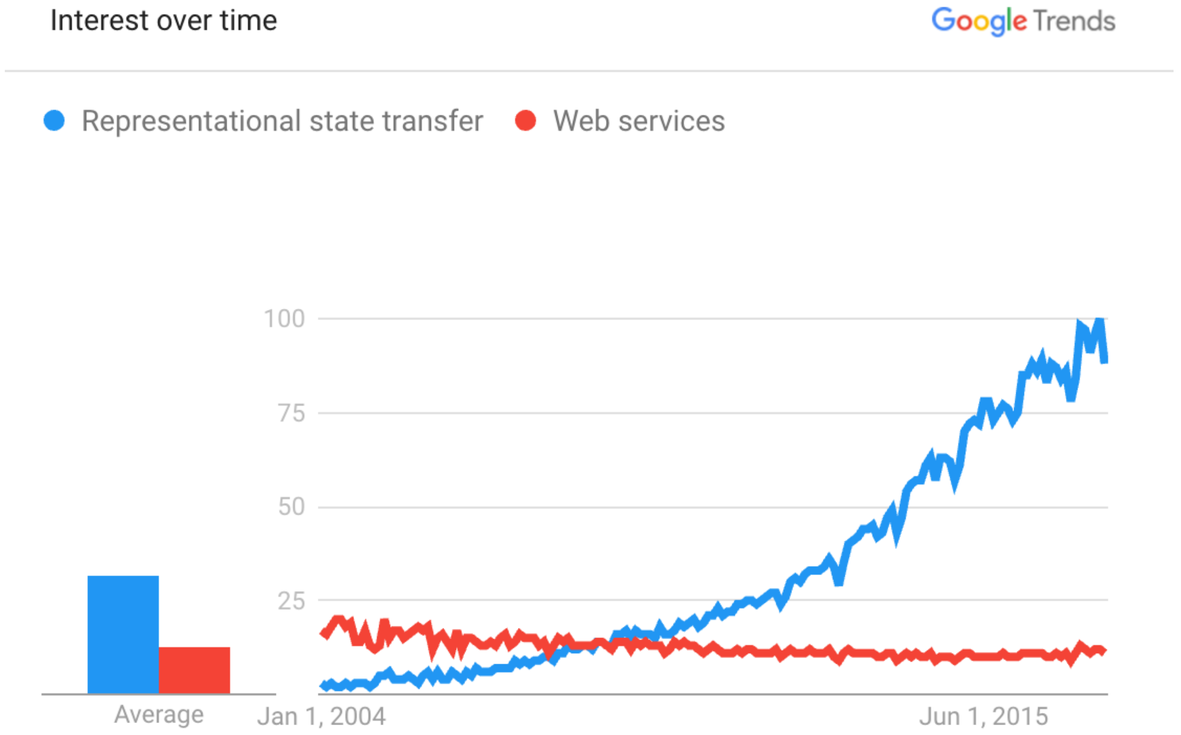
- How do you make adaptability easier?
  - Independent post-deployment evolvability
- Expose the application's architecture
  - Allow third-parties to evolve application by changing architecture
- Verify changes against semantic annotations on the system model
  - with assistance of external analysis modules
  - if change is okay, apply it to the implementation; else, take appropriate action; notify, prevent, …

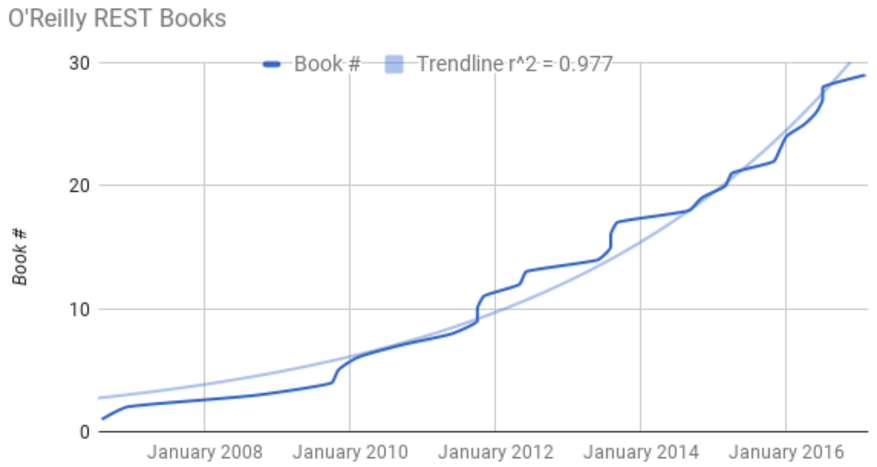


Peyman Oreizy, Nenad Medvidovic, and Richard N. Taylor. 2008 Runtime Software Adaptation: Framework, Approaches, and Styles. In *Companion of 30th International Conference on Software Engineering (ICSE Companion 2008)*. ACM, 899-910. (Most Influential Paper Award for ICSE 1998)

Outline

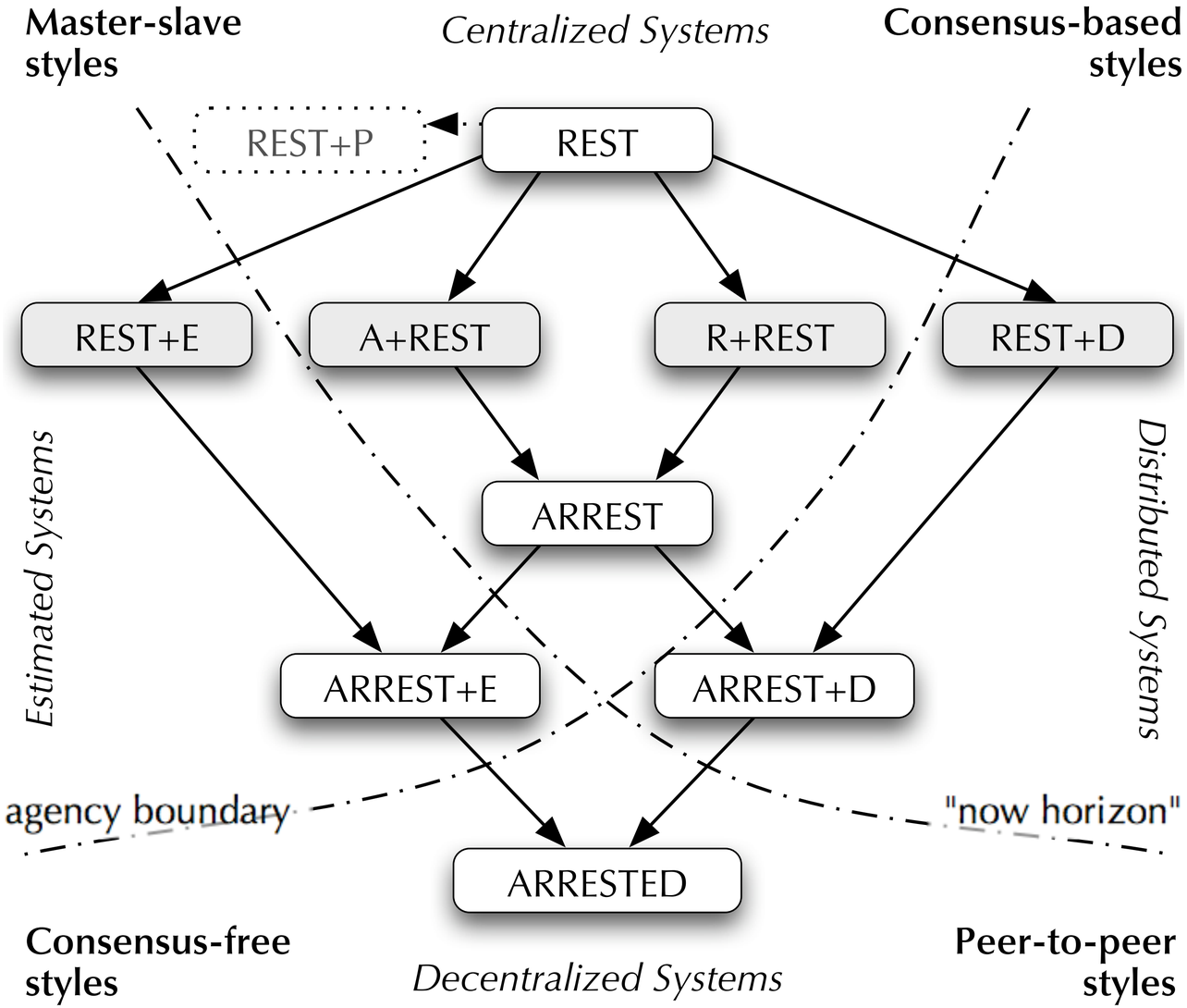
1. The Story of REST
   - Early history of the Web
   - What REST is (and is not)
   - Contemporary influences
2. Work inspired by REST
   - Decentralization
   - Generalization
   - Secure computation
3. Reflections on REST
   - Investing in entrepreneurial students
   - Role of Software Engineering research





New challenges for real-time network-based applications

- By 2000, there was a boom in real-time applications:
  - Push, Peer-to-Peer, Publish/ Subscribe, Instant Messaging, and Internet-Scale Event Notification…
- REST had gaps for real-time:
  - *One-shot:* no retry if response lost
  - *One-to-one:* no concurrent groups
  - *One-way:* no asynchronous links
- *Latency* & *Agency* concerns



Rohit Khare and Richard N. Taylor. 2004. Extending the REpresentational State Transfer Architectural Style for Decentralized Systems. In *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*. IEEE Computer Society, Edinburgh, Scotland, UK, 428–437. (Distinguished Paper award for ICSE 2004) http://www.ics.uci.edu/~rohit/ARRESTED-ICSE.pdf

Computation exchange: CREST

- Technical triad of the Web:
  1. URLs name information resources
  2. Metadata used for distinguishing representations
  3. HTTP defines exchanges between clients and servers
- What happens when you generalize?
  A. CURLs name *computation resources*
  B. *Metaprogamming* is used for examining and describing computations
  C. Asynchronous protocol for *peer-to-peer* exchanges
- CREST learned about deferring code from 'living labs' like Subversion
  - Analysis of the essential architectural decisions of the WWW, followed by generalization, opened up an entirely new space of decentralized, Internet-based applications based on computations as the fundamental concept.

Justin R. Erenkrantz, Michael M. Gorlick, Girish Suryanarayana, and Richard N. Taylor. 2007. From Representations to Computations: The Evolution of Web Architectures. In *ACM SIGSOFT Symposium on The Foundations of Software Engineering (FSE'07)*. 255–264.

CREST with security: COAST

- Capability based security model with computation exchange
- Exchange active computations among peers: Code + run-time state (reified as closures and continuations)
- Novel security mechanism: Capability URL (CURL)
  - Dictates where computations may go
  - Bounds what visiting computations can do
  - Limits resource consumption of computations
- Architectural style: ComputAtional State Transfer (COAST)
  - Build capability security into the architectural style
    - Functional capability: What can a visiting computation do?
    - Communication capability: With whom, when, and how often may that computation communicate?

Michael Martin Gorlick. 2016. *Computational State Transfer: An Architectural Style for Decentralized Systems*. Ph.D. Dissertation. University of California, Irvine.

Michael M. Gorlick, Kyle Strasser, and Richard N. Taylor. 2012. COAST: An Architectural Style for Decentralized On-Demand Tailored Services. In *Proceedings of 2012 Joint Working Conference on Software Architecture & 6th European Conference on Software Architecture (WICSA/ECSA'12)*. 71–80.

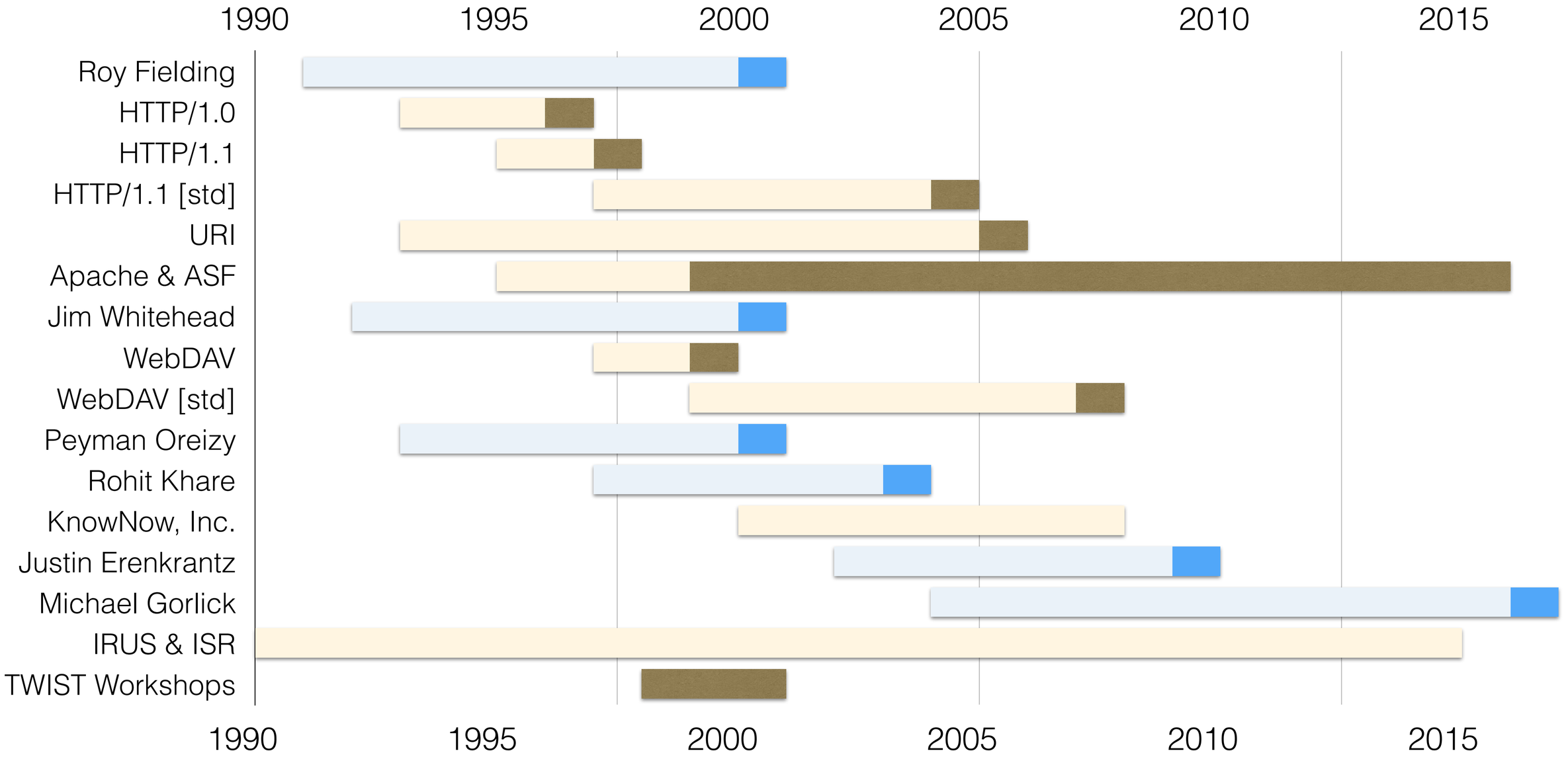
COAST: ComputAtional State Transfer

- An Architectural Style for the Idiom of Computation Exchange
  1. Service: All services are computations whose sole means of interaction is the asynchronous messaging of values, closures, continuations and binding environments
  2. Execution: Each computation executes within the confines of some execution site ⟨E, B⟩ where E is an execution engine and B is a binding environment
  3. Messaging: Computation x may deliver a message to computation y only if x holds a CURL u and y has the authority to read a message via u
     - A Capability URL (CURL) conveys the authority to communicate and is a tamper-proof cryptographic structure that can not be forged or guessed.
  4. Interpretation: The interpretation of a message delivered to computation y via CURL u of y is u-dependent
  5. The Service and Messaging rules confer communication by introduction:
     - Computation x can message computation y only if x has been introduced to y beforehand
  6. Ab initio, endowment, Messaging, or Execution

Outline

1. The Story of REST
   - Early history of the Web
   - What REST is (and is not)
   - Contemporary influences
2. Work inspired by REST
   - Decentralization
   - Generalization
   - Secure computation
3. Reflections on REST
   - Investing in entrepreneurial students
   - Role of Software Engineering research

Investing in entrepreneurial students, over long periods…



The Role of Software Engineering research

- This **is** Software Engineering research
  - It is about Design
  - It is fundamentally software architecture
  - It is based on reflection
  - This is how styles....good styles... get developed:
    - Experience, evaluation, reflection. Wash Rinse Repeat.
- The research environment was “unusual”
  - DARPA funding with a long leash (Thanks Bill and John!)
  - Lots of travel. Lots
  - A (mostly) accommodating university process
    - 9 years to Ph.D after B.S.
  - A highly interactive research community: UCI’s PhD students, W3C, IETF, and close industry links
- Contrast with today's environment…

The Paper

- The first version of what eventually became *Principled Design of the Modern Web Architecture* was submitted to FSE99 a year earlier.
- It was **rejected**, with reviewer comments including “Over all, the originality of the paper is quite low. There is only little to learn from it.” and “- the web is old technolgoy [sic] now. - lots of jargon make the paper difficult to understand. ... - I can't find a novel lessons [sic] for software engineers in this paper.”
- The ICSE 2000 paper had:
  - no surveys, no statistical analyses, and essentially no evaluation section. It merely stated:
    - “The REST architectural style has been validated through six years of development of the HTTP/1.0 and HTTP/1.1 standards, elaboration of the URI and relative URL standards, and successful deployment of several dozen independently developed, commercial-grade software systems within the modern Web architecture.''
- That work, in its multiple forms, has now been cited over 8000 times…

Acknowledgments

- Funding
  - DARPA: Bill Scherlis and John Salasin; National Science Foundation; ISR's corporate sponsors
- The Web
  - Tim Berners-Lee, Henrik Frystyk Nielsen, Dan Connolly, Dave Raggett, and Larry Masinter
- REST advocates
  - Mark Baker, Paul Prescod, Mike Amundsen, Leonard Richardson, Sam Ruby, and Aaron Swartz
- Colleagues
  - Mark Ackerman, Ken Anderson, Greg Bolcer, Eric Dashofy, Nenad Medvidovic, Kari Nies, Jie Ren, Jason Robbins, David Rosenblum, and Girish Suryanarayana.

- Thanks —
  - To the SIGSOFT community for the honor of this Impact Award.

Questions?