

# Real-time Web 2.0: Evolution of Middleware for Grid-based Instruments and Sensors

Donald F. (Rick) McMullen

Marlon Pierce, Carol Deng, Kia Huffman

Knowledge Acquisition and Projection Lab & Community Grids Lab

Pervasive Technology Labs at Indiana University

[mcmullen@indiana.edu](mailto:mcmullen@indiana.edu)

Open Grid Forum 21, October 15, 2007, Seattle, WA



# Acknowledgements

- Co-authors: Marlon Pierce, Carol Deng, Kia Huffman
- National Science Foundation, through grants and cooperative agreements SCI 0330568, DBI 0446802, IIS 0513768, and IIS 0513687. The Data Capacitor is supported by the National Science Foundation under Grant No. CNS0521433.
- CIMA NSF CA **co-PIs** and project team members:
  - **John C. Huffman** (IU Chemistry), **Randall Bramley** (IU CS), **Ken Chiu** (SUNY Binghamton CS), Kianosh Huffman (IU Informatics), Gilead Kutnick (PTL@IU), Tharaka Devadithya (IU CS), Thomas Reichherzer (~GaTech), Charles Hart (IU), Siddika Chowdhury (Binghamton CS), Vishesh Panchal, Scott Dial (IU CS), Dr. Jaesoon An, Bill Tilghman, Lawrence Meehan, Carol Deng (PTL@IU), Alejandro Valerio (IU CS)
- Collaborators at the Community Grids Lab (PTL@IU), OGCE group, Advanced Photon Source, Argonne National Labs, Purdue University, University of Minnesota, Case Western, James Cook University, University of Sydney, Adelaide University, University of Queensland, University of Southampton, UK National Crystallographic Service.



# Roadmap

---

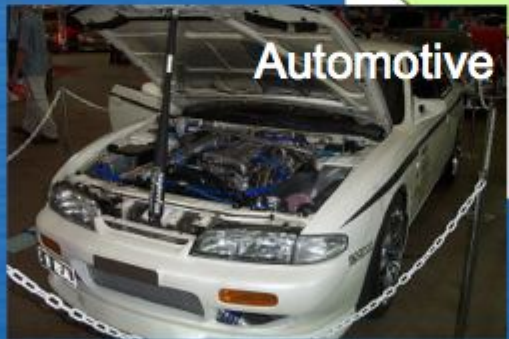
- 10Km view: real and near-real time access to sensors and instruments
- CIMA and SOA access to instruments
- Web 2.0
- Putting the pieces together: enabling the social construction of the Real World Web



Market logistics



Buildings: security, safety

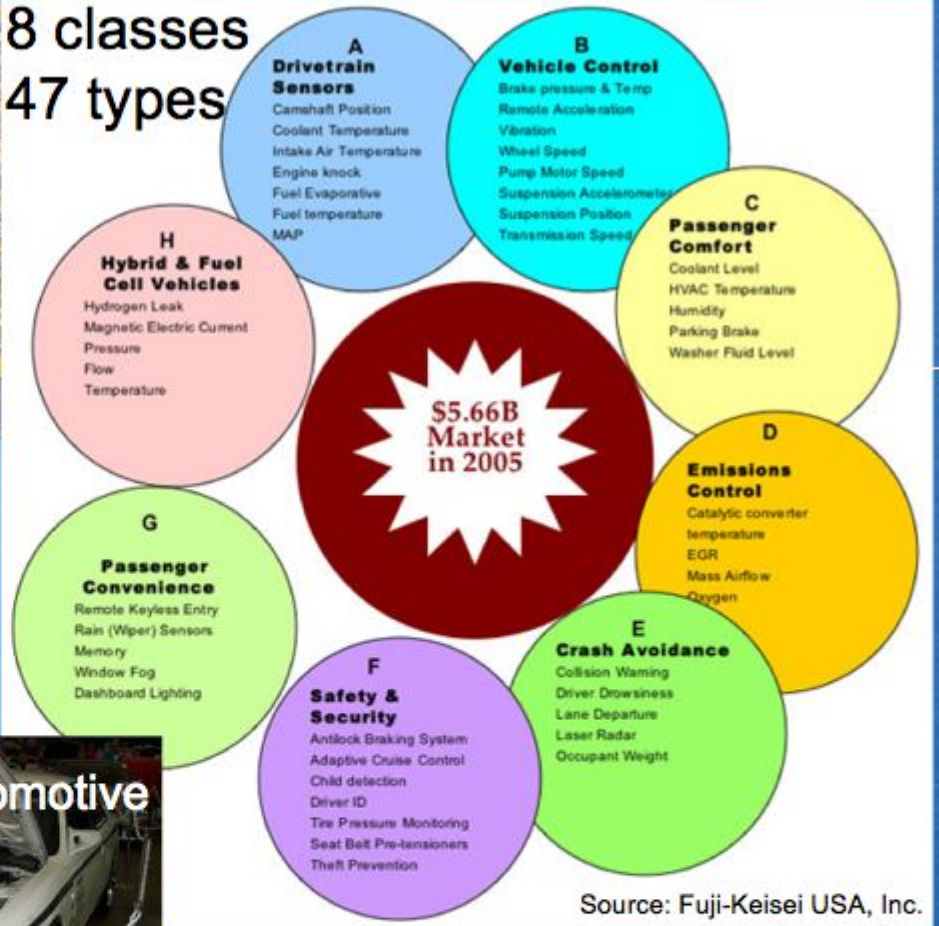


Automotive



Rural/Developing countries health care

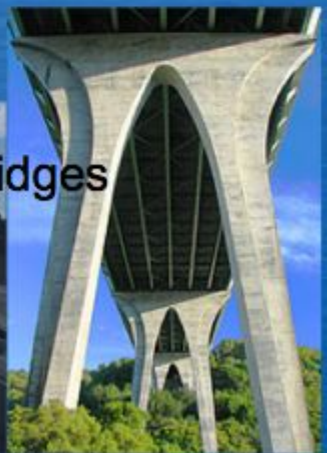
8 classes  
47 types



Source: Fuji-Keisei USA, Inc.



Roads and bridges



Sensors are everywhere...  
Time for the Real World Web

# Instruments and sensors for scientific discovery



Focus on scientific instruments and Grids that are driven by data from instruments and sensors





# Software systems for RT sensor and instrument access

---

- Device access
  - How do I talk to a real time data source?
  - What services are provided on behalf of the sensing hardware?
  - How are data transported to and from the device?
- Device characteristics and capabilities
  - The three Ws & H: Who, What, Where and How
- Integration frameworks and projects
  - Co-development of hardware and delivered services



# Potential Solutions

- Device access
  - IEEE 1451.1 - access to "smart" sensors using IP
  - Universal Plug and Play --> WS-Discovery
  - Antelope (Boulder Real-Time Systems)
  - NeesGrid Telecontrol Protocol
- Device capabilities
  - OpenGIS Consortium GML, sensorML - XML schema for sensor description
  - W3C Composite Capabilities/ Preferences Profile - ontology based RDF descriptions of mobile device capabilities
  - IEEE 1451.0 - layered specification for "universal" sensor access; .0 is Transducer Electronic Data Sheet (TEDS)
  - Monterey Bay Aquarium Research Institute "plug and work" instrument puck
- Integration
  - ORNL SensorNet
  - SDSC/CALIT<sup>2</sup> RoadNet (based on Antelope)
  - GRIDCC instrument grid architecture
  - MIT iLab project (education applications of remote access to instruments)
  - DARPA/DoD Sensor Web Enablement community
  - Event Heap (Kindberg, A. Fox, Johanson, Hanrahan, Winograd)



# Getting to the Real World Web

- Middleware that makes many kinds of instruments and sensors embedded in our world accessible as services in a **composable** way
- “Equal access” to real time data sources
- Easy to find data sources
- Easy to use a service
- Easy to connect data from the service to other services
- A Web of instruments and sensor services with similar interfaces
- Our approach is the Common Instrument Middleware Architecture (CIMA)

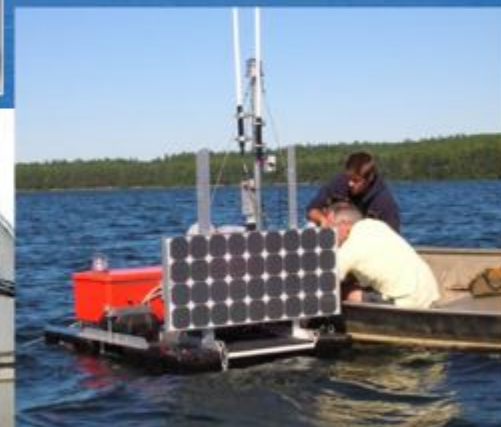
# CIMA Project Goals

Project supported by the NSF Middleware Initiative to:

- Integrate instruments and sensors as real-time data sources into grid computing environments through a Service Oriented Architecture
  - Improve accessibility and throughput in instrumentation investments
  - Promote sharing across institutions and disciplines
- Develop a methodology for describing instrument capabilities and functions and embedding these in the hardware to improve flexibility and lifetime of data acquisition and analysis applications
  - Automatic generation of basic metadata
  - Towards a Semantic Web for instruments and sensors
- Move production of metadata as close to instruments as possible and facilitate the automatic production of metadata
  - Improve data management, provenance and reuse

# CIMA Reference Implementation Applications


- **Synchrotron X-Ray crystallography**
  - Argonne APS ChemMatCARS & UNI/XOR
  - Lab systems through CrystalGrid (global network of crystallography labs)
- **Microscopy**
  - Data acquisition and mgmt.
- **TOF-MS**
  - Identification of proteins and other macromolecules
- **Robotic telescopes**
  - Remote observing for students and researchers
  - Queued and interactive operation
- **Sensor networks**
  - Ecological observation (LTER lake buoys, ocean platforms)
  - Low power wireless sensor network elements





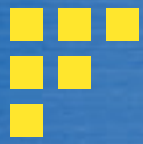
# A Dozen (or so) Requirements

- Self-describing instruments/sensors and data
- Functional transparency - what you see is what you get
- Adequate real-time performance (min. latency in control apps)
- Multiple modes of interaction (read/write, streaming, conditional response, sequenced programs)
- Resource oriented stateful services
- Reliable, available and serviceable
- Interoperable - basic interaction motifs are the same across instruments and data have a uniform “envelope”
- Efficient data transfer - support multiple (hierarchical) fieldbuses and interconnects
- Lightweight with respect to CPU and memory requirements
- Supports Intermediate processing on data streams
- Supports authorization, not just authentication
- Applicable to existing instruments and sensors

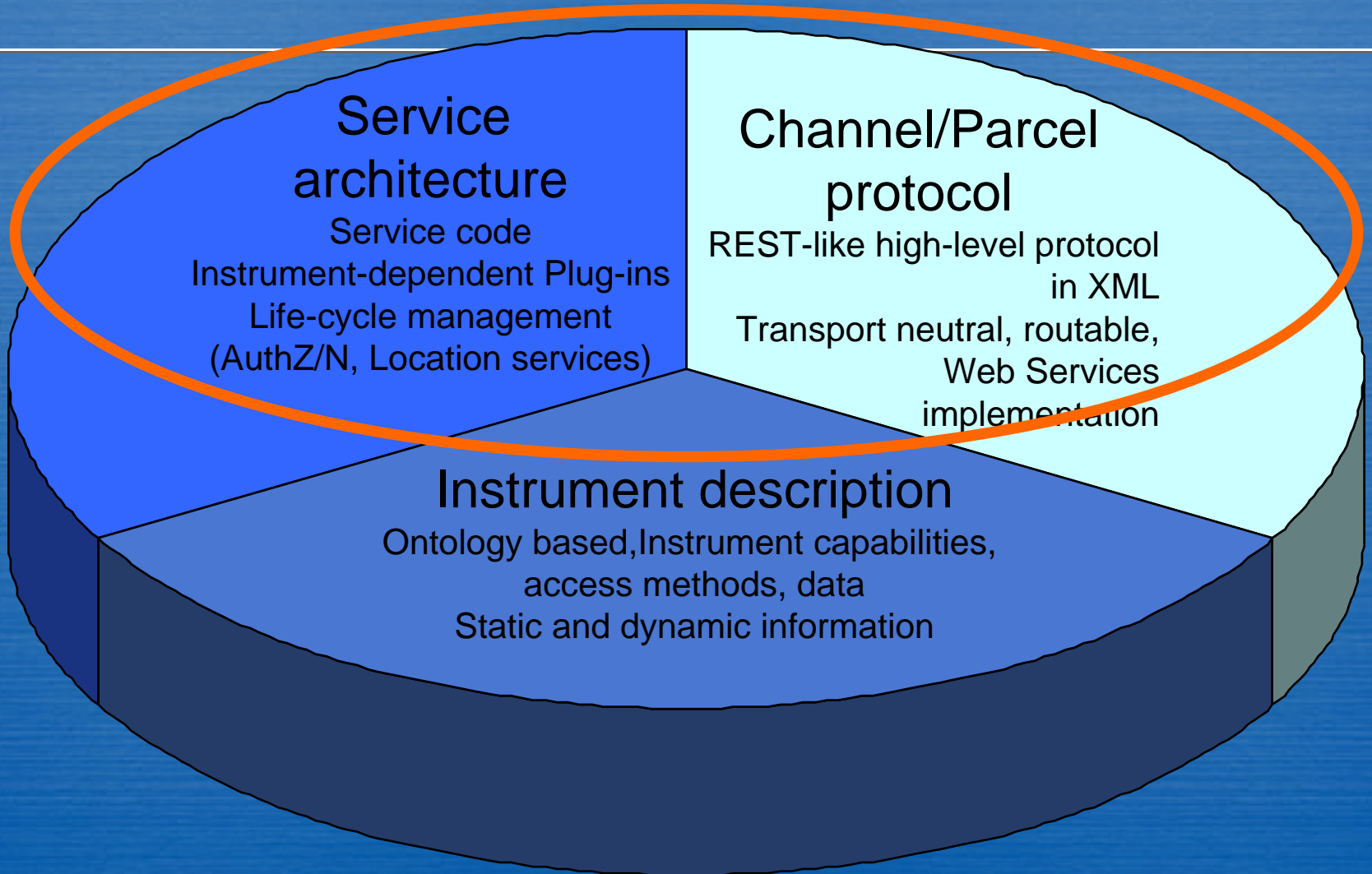


# Three Components of a middleware for instruments and sensors

- Reusable service architecture
  - Configured at run-time for specific instruments
- Transport independent protocol for interacting with an instrument's functions
  - Hierarchical interconnects (e.g. 802.11g->802.15.4 W-PAN)
- Representation for the functions and capabilities of an instrument
  - Capabilities described in a controlled vocabulary (ontology)
  - Decentralized process for extending capabilities profiles
- AND Standards used to interpret measured variables
  - Controlled vocabulary for observables, units, calibration, conversion, etc.



# CIMA Components

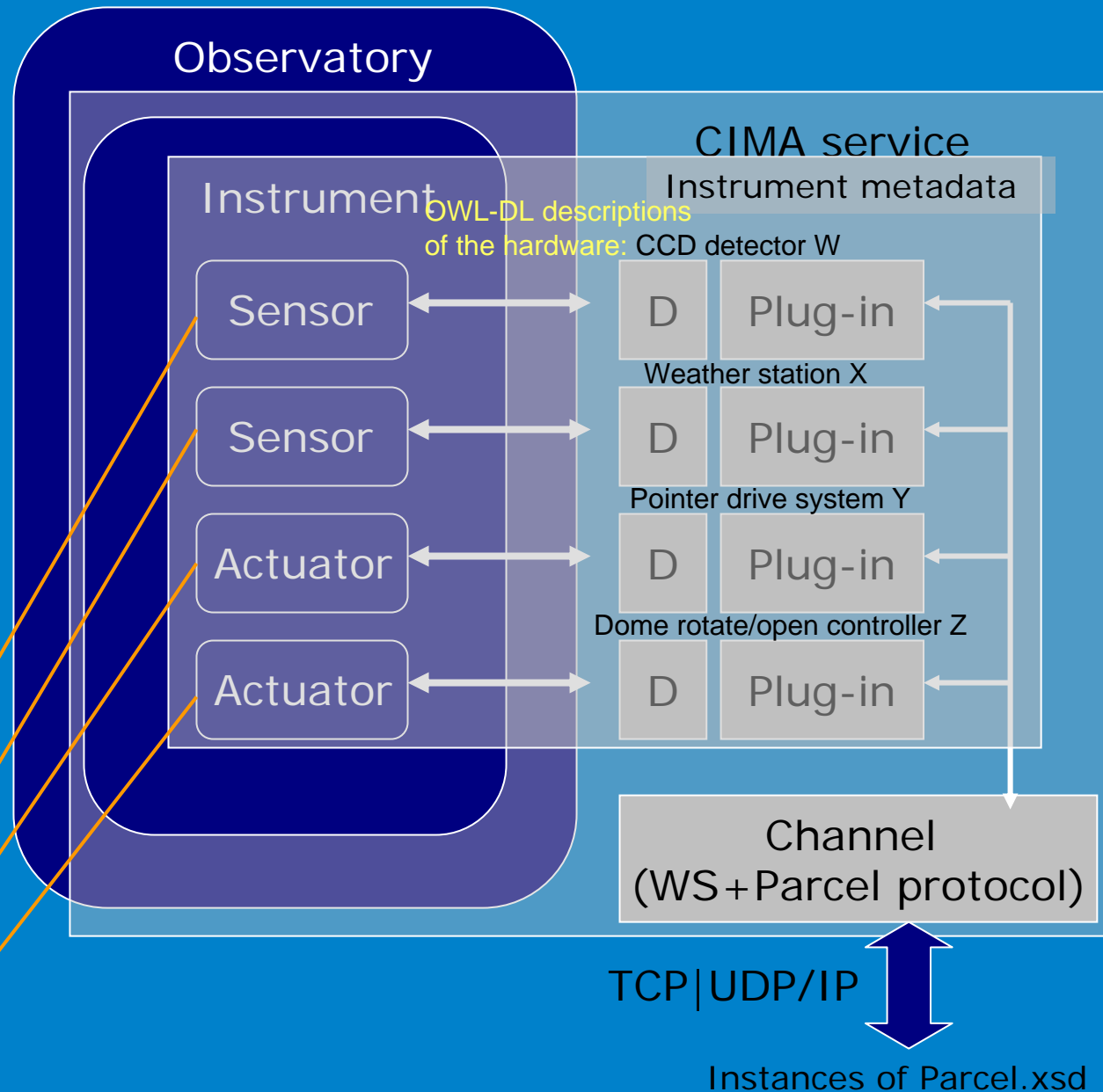


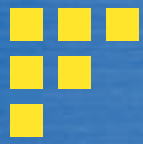
# Top-level CIMA Instrument Model



## Components

- Optics and imaging
- Interior and exterior conditions
- Positioning
- Dome control

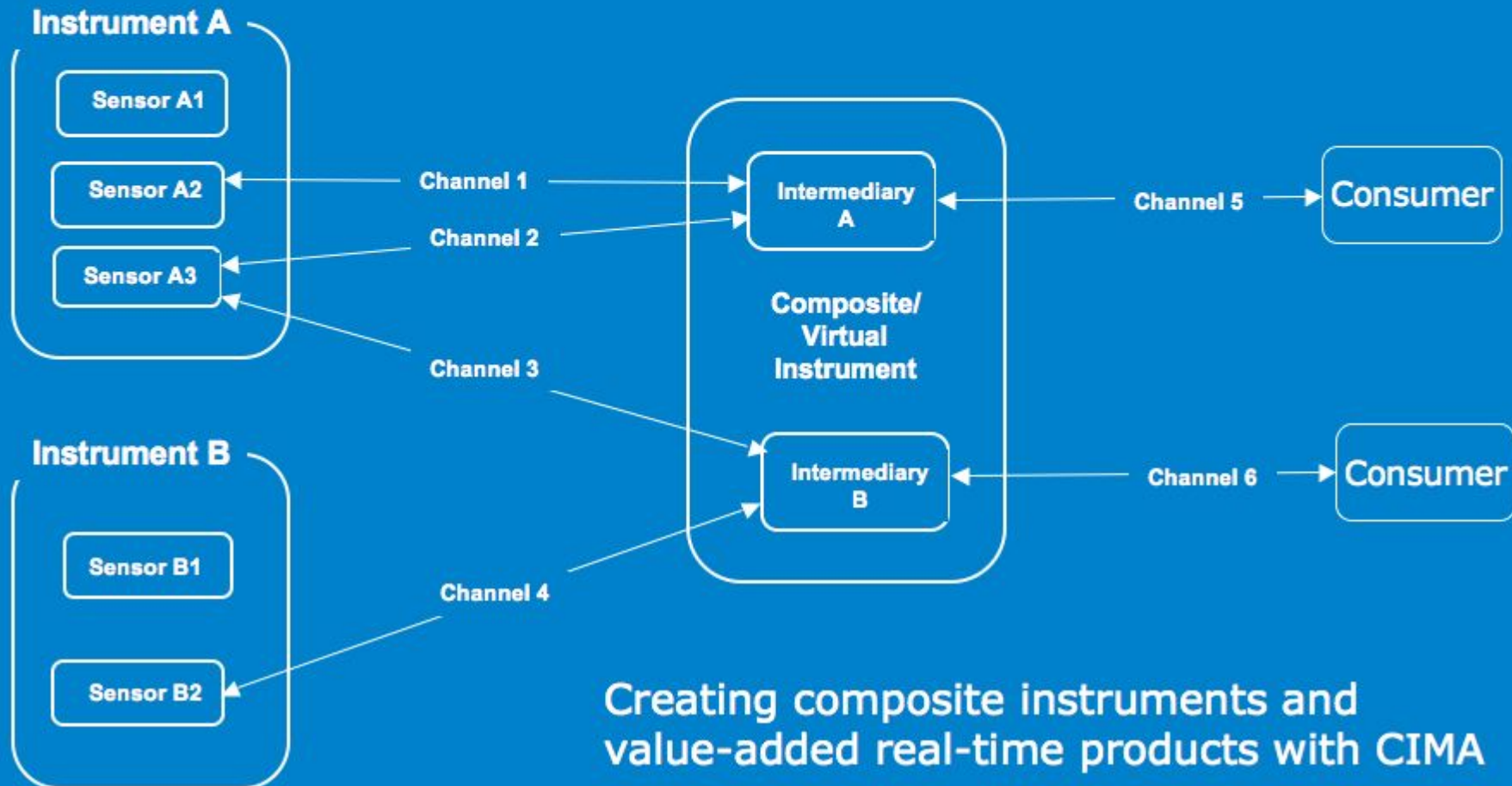


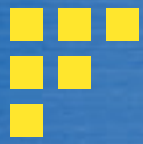


# CIMA Channel Protocol

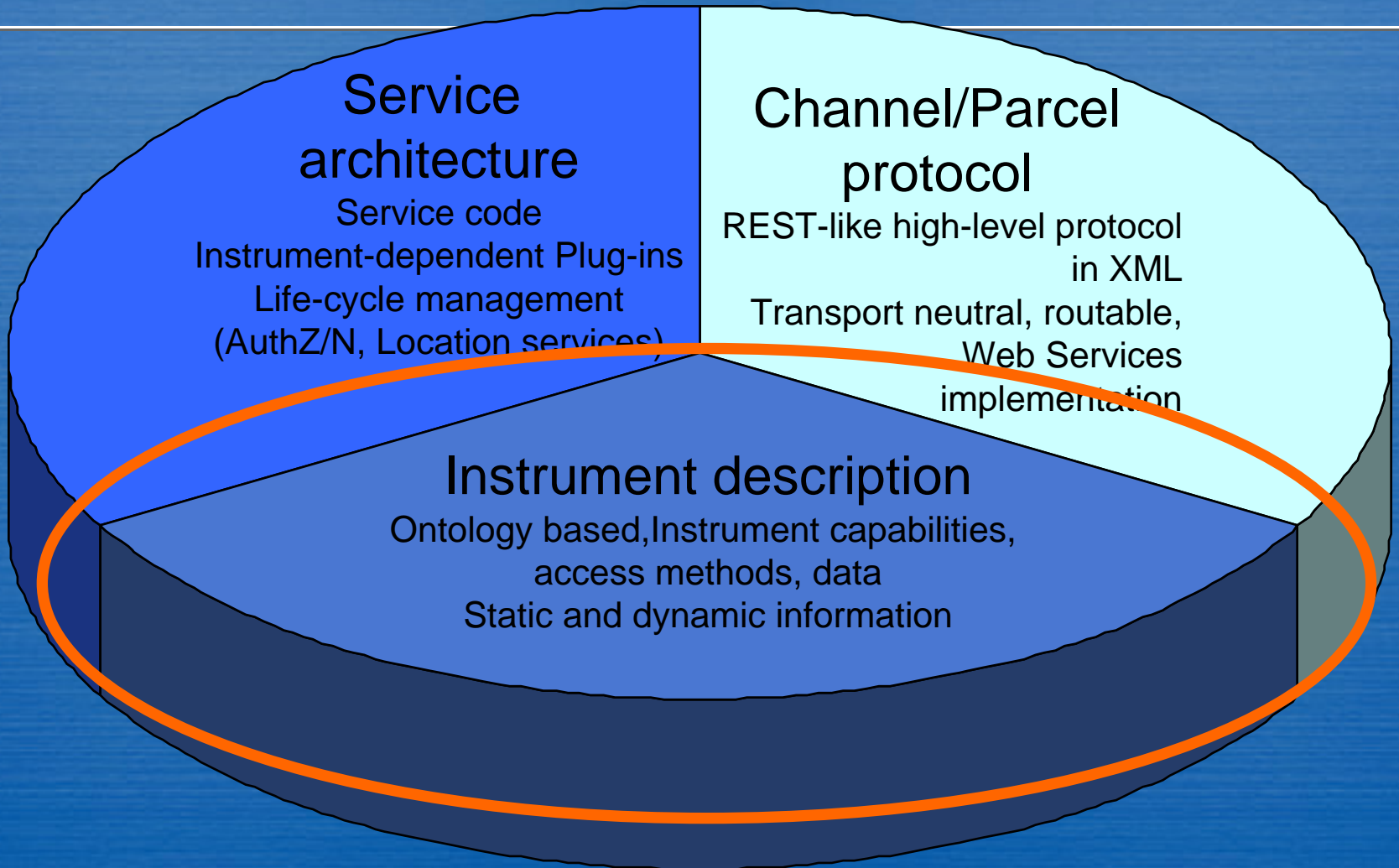
- Built on a **simple Web Services interface**: One endpoint at instrument, zero or one at client, both are String type, Doc/literal wrapping
- SOAP payload is a **Parcel XML document** or fragment; **REST-like protocol** using document oriented SOAP messages (*Parcels*): *session, describe, register, get, set, unregister*
- Additional **layered specifications** can be added as needed, e.g. WS-Security, WS-Addressing ...
- Protocol can be extended by versioning the parcel schema without modification to older clients or instrument services
- Clients can request data on multiple streams at different rates; One instrument service supports multiple clients

# CIMA in processing pipelines





# CIMA Components



# Instrument description and metadata

- Physical and logical attributes of the hardware and data produced by it
- Maps sensors and actuators defined in Plug-ins to the semantics of control functions and data outputs of the instrument
- Applications can query the instrument description to build an operational model of the instrument on the fly
- Description instance based on ontology, implemented as OWL-DL in RDF
- Community extensible in both ontology and instances (real net accessible hardware)



# Automation from the CIMA Ontology

---

- Generate data acquisition strategy and code
- Build user- and task-appropriate interfaces
- Automate production of high quality metadata
- Locate experimental resources based on high-level specification of goals
  - Towards a Semantic Web of instruments and sensors
  - “Interpreted” Real World Web

# Web 2.0



# Web 2.0 Characteristics

- Web 2.0 provides SOA-like functionality with an ideal of RESTful services
  - Emphasis on simple and transparent functionality: “Everything just does what it says”
  - Leverage HTTP as transport and error handler
  - Focus on “Resource Oriented Architecture” with operations on resources with URI handles
  - Low knowledge requirements, low entry barriers
  - Potential for rapid spiral development
- “Using the entire Internet as your API for new applications”
  - Many useful services, e.g. GoogleMaps; Amazon S3, EC, Queue Service, etc.
  - 523 APIs listed at [www.programmableweb.com/apis](http://www.programmableweb.com/apis) (10/10/2007)
- **Social networking**: sites, services and APIs for blogging, sharing images, video, etc.
- And it’s good fun!



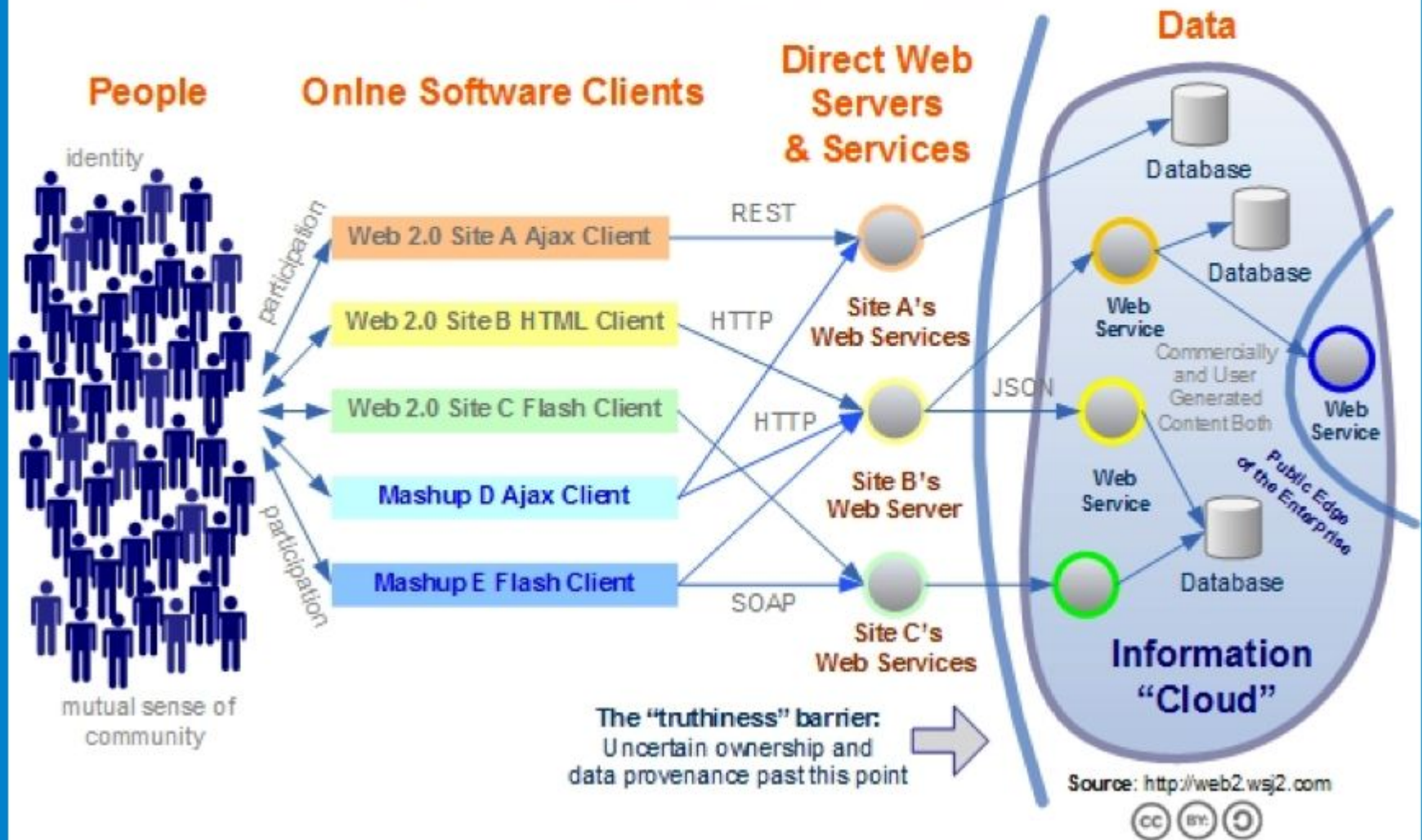
# REST

- Representational State Transfer\*
- Data and services are *Resources* addressable through handles (URIs)
- Clients and servers exchange operations on *representations* of the resources
- Emphasis on small number of operations on a wide variety of resources
- Core functionality from HTTP verbs GET, POST, PUT, HEAD, DELETE on a multiplicity of resources (URLs)

\* “Architectural Styles and the Design of Network-based Software Architectures,” Roy Fielding, Ph.D. Dissertation, UCI, 2000. Ch. 5.

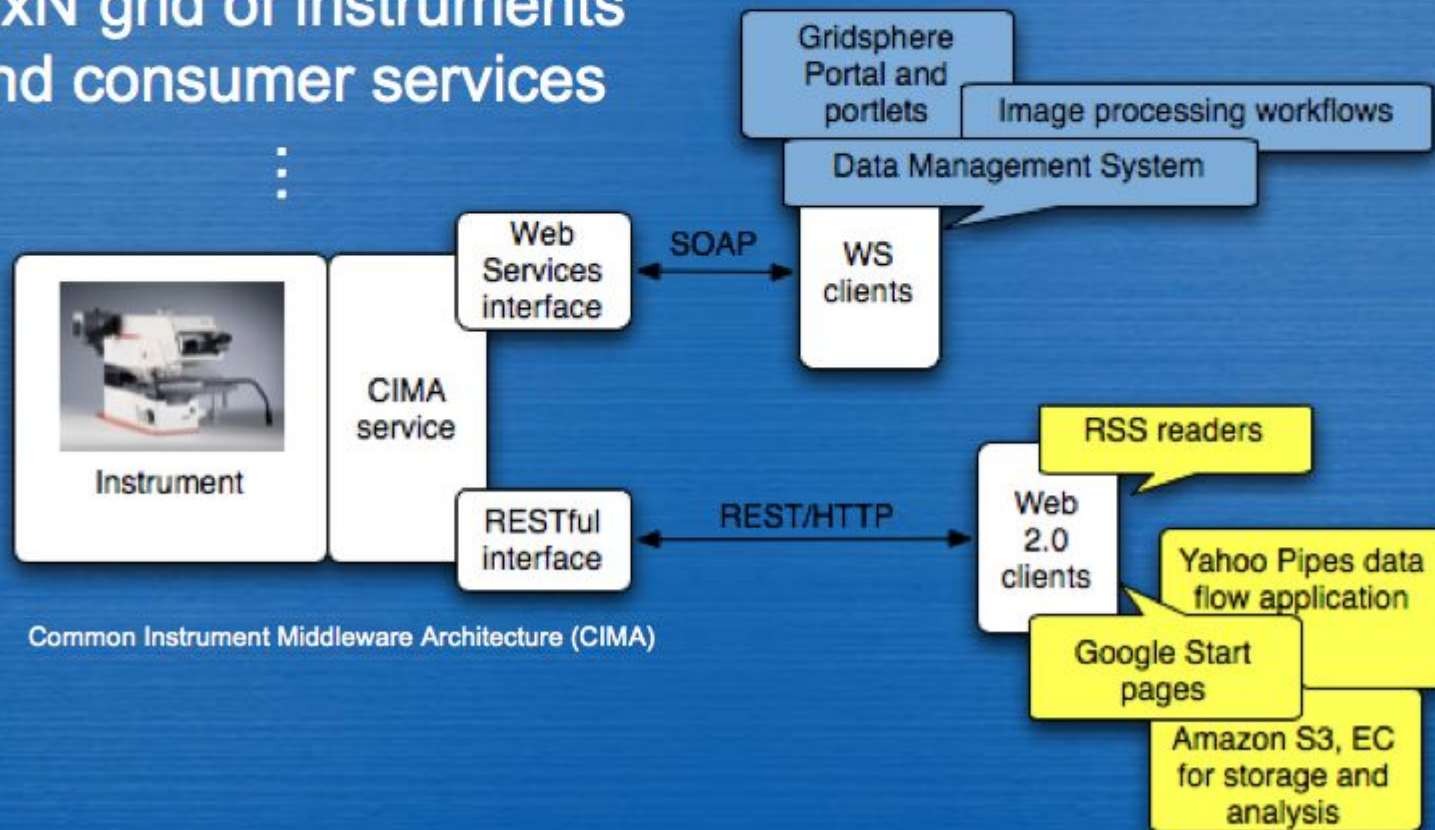
# Web 2.0 APIs as mediators for ubiquitous data

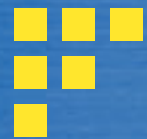
## The Web 2.0 Architecture of Participation: "People in the Machine Nurture the Cloud"



# Real World Web for e-Science

MxN grid of instruments  
and consumer services





# Moving CIMA from Web Services to Web 2.0

- WS SOA version is event driven with server making WS calls to client with data (and simple request-response)
- Use cases
  - Stateful control and real time status of hardware
  - Streaming data to a web page for monitoring (near real time)
  - Streaming data to a non-browser client (near real time)
  - Streaming data to storage
  - Data to a news feed for “digest” level awareness
- Web 2.0 approaches
  - RESTful interfaces
  - HTTP server on client to catch HTTP events
  - Polling (pushlets, etc.); Ajax, JSON, etc.
  - Publish/subscribe
  - RSS, Yahoo Pipes



# Web 2.0 approaches

- RESTful interfaces to map WS functionality
- Polling (pushlets, etc.); Ajax, JSON, etc.
  - Code on service to push data as web pages to client
  - 90% XML, 10% JS+HTML
- HTTP server on client to catch data as HTTP events
  - Analogous to WS-based architecture; use RESTful approach
  - 100% XML, easy extension since SOAP already needs one
- Publish/subscribe
  - jMaki
  - Pub/sub systems, e.g. NaradaBrokering, JMS, XMPP
- RSS
  - Good for low volume low frequency data
  - Yahoo Pipes for aggregation and fusion
- Payload
  - XML Parcel Schema document
  - Web page plus Microformats for searchability

# Example: Microsoft Robotics Studio RESTful interfaces for RT interactions in robotics

Jon Udell

July 25, 2007

## Henrik Frystyk Nielsen on the RESTful architecture of Microsoft Robotics Studio

Filed under: Uncategorized — Jon Udell @ 9:14 am

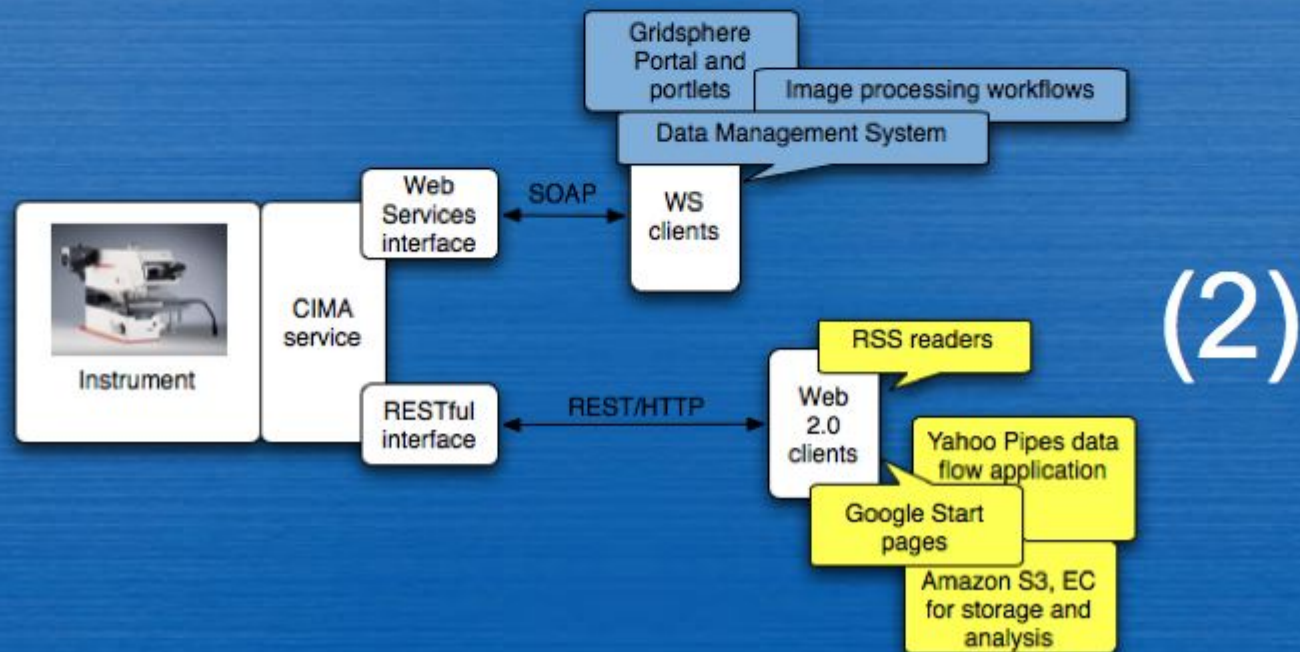
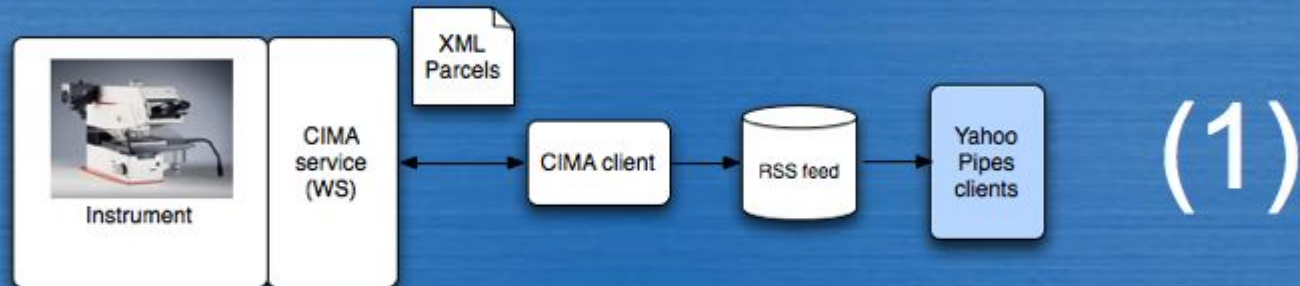
Henrik Frystyk Nielsen used to work for the World Wide Web Consortium on some key pieces of infrastructure including the HTTP specification and libwww. He left the W3C in 1999 and now works for Microsoft where his current project is Robotics Studio, whose tagline is: "A Windows-based environment for academic, hobbyist and commercial developers to easily create robotics applications across a wide variety of hardware." What that description doesn't tell you, but today's screencast shows, is that the Robotics Studio is based on a RESTful architecture, and that applications are built by composing lightweight services in ways that will be instantly familiar to every web developer.

To drive home that point, much of the action in this screencast occurs in a web browser, where you'll see Henrik explore a distributed directory of services and view XML snapshots of the current state of bumpers, cameras, and laser range finders.

From a read-only perspective it's all HTTP GET, and you can do things like subscribe to robotic sensors using RSS feeds. When you control a robot, SOAP is used to optimize fine-grained updates. But either way it's a loosely coupled and late bound system that leverages the fundamental flexibility of web architecture in a very different domain. In one compelling demonstration of that flexibility, you'll see a generic controller — which had been controlling the robot in Henrik's office with no prior knowledge of the device, purely by interface discovery — switch over to a simulated robot and drive it by means of the same kind of discovery.



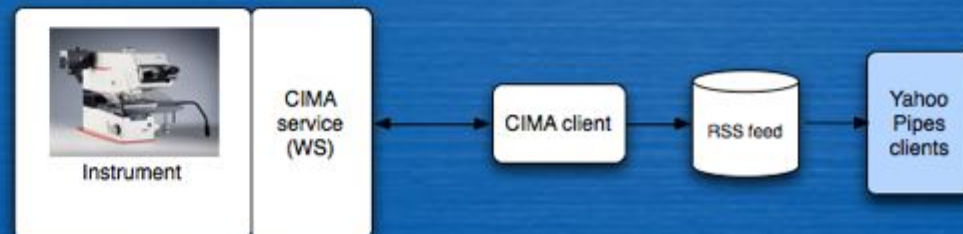
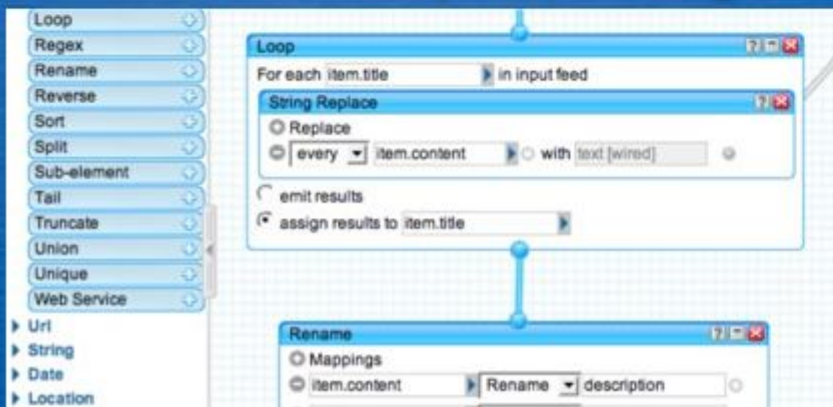
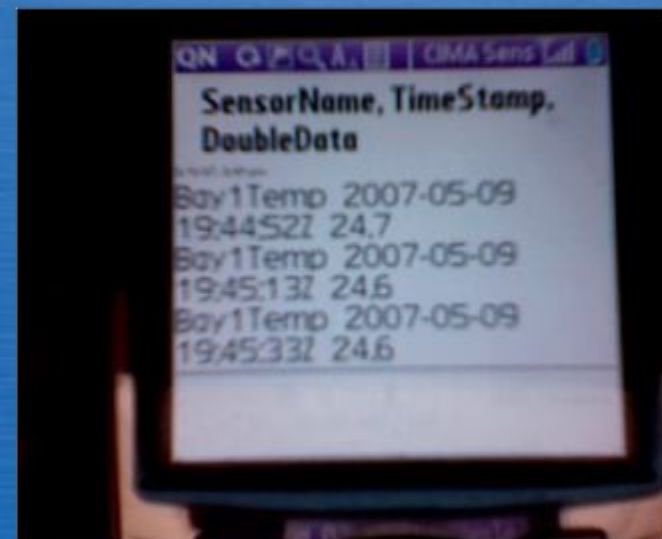
# CIMA -> Web 2.0, Takes 1 & 2



## Take 1:

# RSS/Atom feeds as an interface to instruments and sensors

- CIMA client “shim” registers for and receives events, output to RSS feed
- RSS feeds and event detection to handheld devices
- Use Yahoo Pipes for shared distributed processing



# A More RESTful interface



- Resource exposed is the CIMA service + source/sink
- Atomic CIMA operations translate OK as GET/PUT or POSTing Parcel XML and receiving a reply
- Primary problem: Mapping from event driven Web Services callbacks to [something else]
- Solution 1: Server push (e.g. pushlets)
  - Advantage: supports streaming of data from service
  - Disadvantages: long term instability of server connections, client support for parsing streaming data
- Solution 2: Asynchronous callback to a URL
  - Advantage: analogous to WS implementation
  - Disadvantage: requires a web server at client
- Solution 3: ?



# GET, PUT and POST

- Need to preserve idempotence of GET and PUT
  - Semantic issue about what this means to GET a time-varying signal; PUT should have no further side effects
- Resource exposed is the CIMA service or service+source/sink
- GET returns value of one source as a Parcel
- PUT sets value of one sink from value in URI
- Efficient but polling only, no streaming
- POST is used to duplicate Web Services model
  - Register operation may return multiple Parcels on one connection or may call client web server multiple times

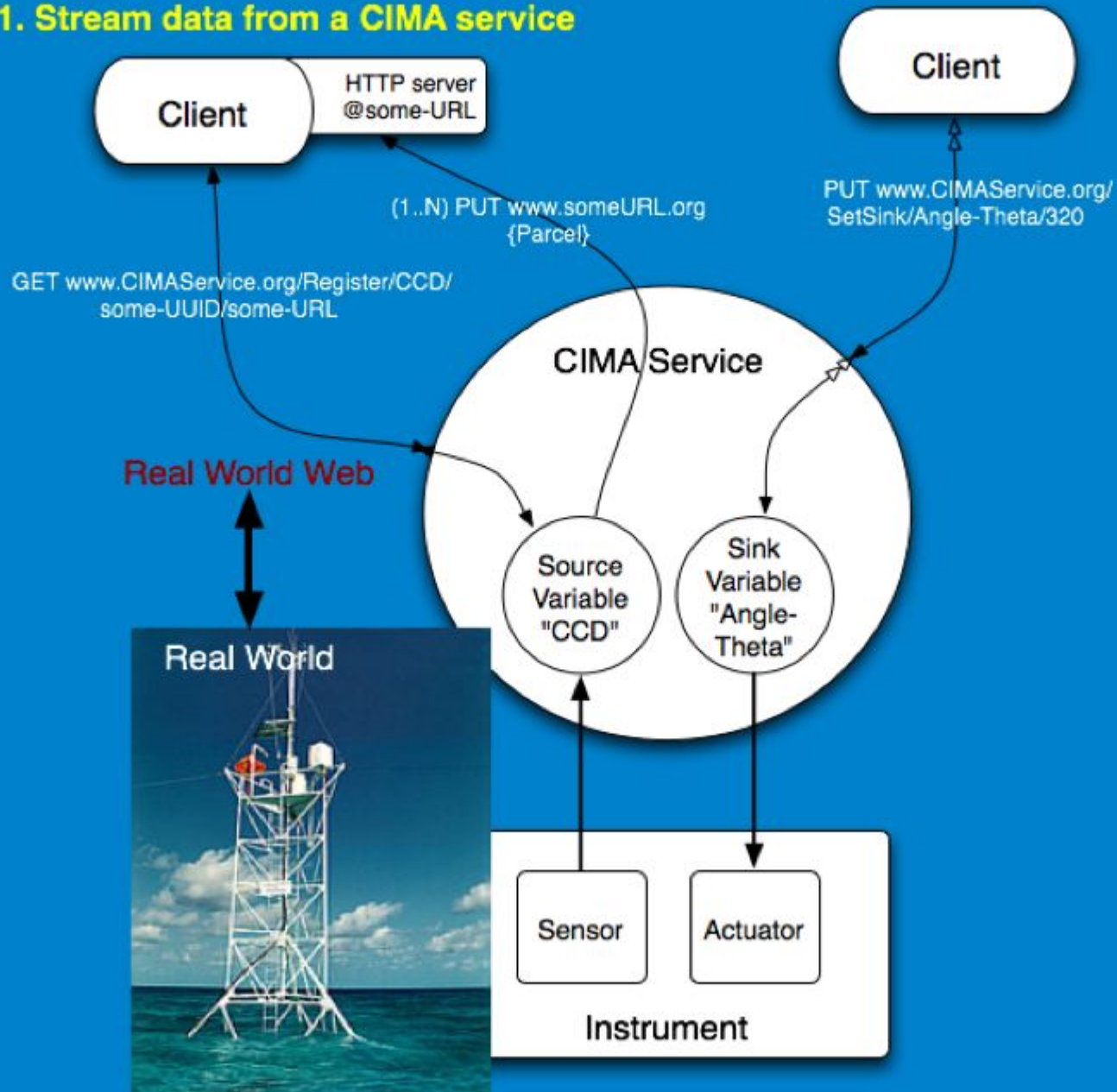
# CIMA RESTful interface

http://{service-url}/{CIMA-Function}/{source-sink-name}/{session\_key}/  
[/*HTML-escaped-value*] (for SetSink)  
[*callback\_spec\_URI*] (for Register);  
Credentials in HTTP hdr. for *Session*

HTTP OP CIMA Function	GET	PUT	POST
Register	OUT: Multiple Parcels or HTTP error code	X	IN: Parcel XML OUT: Multiple Parcels or HTTP error code
GetSource	OUT: XML Parcel containing value	X	IN: Parcel XML OUT: Parcel + HTTP status code
SetSink	X	OUT: HTTP status code	IN: Parcel XML OUT: Parcel or HTTP error code
Describe	OUT: XML Parcel w/description	X	IN: Parcel XML OUT: Parcel or HTTP error code
Session	OUT: XML Parcel w/session key	X	IN: Parcel XML OUT: Parcel or HTTP error code

## 2. Control an actuator through a CIMA service

## 1. Stream data from a CIMA service





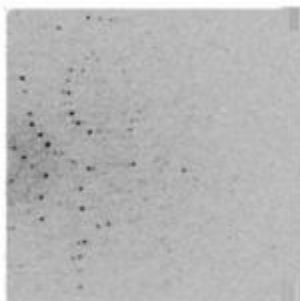
# GUIs for the Real World Web

- Need dashboards, cyberenvironments, lab portals, etc. to connect people to instruments and sensors making up the Real World Web in real time
- Gridsphere + JSR 168 portlets
  - Widely used solution in CI/CE/eScience communities
  - Data binding to Web Services via JSF or similar approach
  - User context part not transferable between JSR 168 containers
  - Identity management through Gridsphere and external user DB
- More Web 2.0-y (Webby 2.0?) solutions
  - Google/Yahoo/Netvibes start pages
  - Widgets and Gadgets
  - Identity management and authN are open questions
    - OpenID, Shibboleth, Multiple simultaneous solutions?
  - Extend formal ontology process to include real time tagging



## IUMSC Bay1

[Disable your browser's cache to get the live stream!](#)

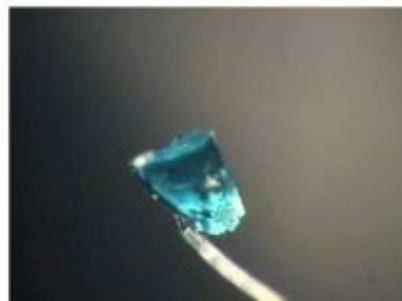


Latest CCD frame from Bruker SMART 6000 CCD detector

Frame: [0617411.384](#) [All frames for this sample](#)



Streaming video from IUMSC lab showing the Bruker instrument & controller PC



Streaming video from the crystal microscope on the Bruker diffractor

Local date/time:2006-10-31 09:15:14

These values are updated approx. every 60 sec. Times in UTC

Crystal Temperature	-148 C	2006-10-30 21:42:12 Z
Liquid Nitrogen level indicator	62.3 %	2006-10-30 21:42:08 Z
Bay1 Temp. & Humidity:	17.6 C Rel. Humid. 42.3 %	2006-10-30 21:42:00 Z
Diffractometer Enclosure Temp. & Humidity:	23.6 C Rel. Humid. 27.4 %	2006-10-30 21:42:35 Z
Campus Chill water IN:	10.5 C	2006-10-30 21:42:09 Z
Lab Chill water IN:	18.4 C	2006-10-30 21:42:08 Z
Lab Chill water OUT:	25.2 C	2006-10-30 21:42:08 Z

Gridsphere portal

JSR 168 portlets +  
Gridsphere Context +Java  
Server Faces (JSF) +  
Web services data  
bindings

Identity management  
through Gridsphere  
and Globus certs. (GAMA)

Developed over many  
months

iGoogle

http://www.google.com/ig

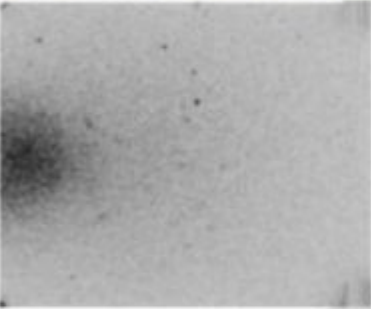
Latest Headlines Slashtmp Wikipedia Weather Travel Scaling Wiki ISDEC - BluWiki post to del.icio.us Agenda for Sensor In... Welcome to uExpress... NSF FastLane Home

**iGoogle** [Advanced Search](#) [Preferences](#) [Language Tools](#)

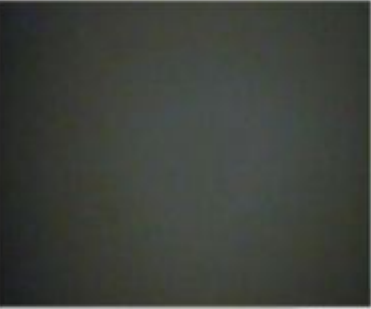
Google Search I'm Feeling Lucky

Home [Add a tab](#) [Select theme](#) [Add stuff](#)


IUMSC Bruker S6K current CCD image



IUMSC Bruker S6K Crystal Camera



IUMSC Bay 1 Overview



Exception Conditions IUMSC

- IUMSC Atom Feed - Abnormal Data

No abnormal data now.

[view link](#)

- IUMSC Atom Feed - Abnormal Data
- IUMSC Atom Feed - Abnormal Data
- IUMSC Atom Feed - Abnormal Data
- IUMSC Atom Feed - Abnormal Data
- IUMSC Atom Feed - Abnormal Data

**IUMSC Atom Feed**

Yu(Carol) Deng Aug 10, 2007 - [Show original item](#)

FrameBuffer\_F\_Usage 2007-08-09 23:10:10Z 89.8  
 FrameBuffer\_E\_Usage 2007-08-09 23:10:10Z 55.3  
 Bay1Temp 2007-08-09 23:10:12Z 16.7  
 CampCWinTemp 2007-08-09 23:10:25Z 13.5  
 LN2Levi 2007-08-09 23:10:15Z 41.5  
 LabCWinTemp 2007-08-09 23:10:25Z 16.4  
 LabCWOutTemp 2007-08-09 23:10:18Z -10000  
 CrystalTemp 2007-08-09 23:10:18Z -122  
 FrameBuffer\_D\_Usage 2007-08-09 23:10:13Z 84.4  
 DEHumid 2007-08-09 23:10:15Z 41.4  
 Bay1Humid 2007-08-09 23:10:18Z -10000  
 DETemp 2007-08-09 23:10:25Z 22.4

Bay1Temp IUMSC

- IUMSC Bay1Temp Atom Feed

sensor=Bay1Temp date=2007-08-14 time=04:49:31Z  
 value=16.1

Data & Time

Done

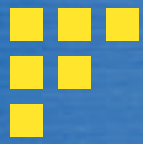
iGoogle start page

HTTP/REST

Identity management through Google.

Easy to build (~10 minutes)

Easy to maintain



# Summary

- Web 2.0 approaches have much to offer in developing eScience application systems and Cyberenvironments using instruments, sensors and other real time data sources
- Web Services and Web 2.0 can coexist and complement each other's strengths in systems/Grids for interacting with real time sources
- Let's build the Real World Web 2.0!



“Web 2.0, like success, has a  
hundred million parents...”



Thanks! Questions?

[mcmullen@indiana.edu](mailto:mcmullen@indiana.edu)